A Survey on Text Classification: From Traditional to Deep Learning

QIAN LI, Beihang University, China
HAO PENG, Beihang University, China
JIANXIN LI*, Beihang University, China
CONGYING XIA, University of Illinois at Chicago, USA
RENYU YANG, University of Leeds, UK
LICHAO SUN, Lehigh University, USA
PHILIP S. YU, University of Illinois at Chicago, USA
LIFANG HE, Lehigh University, USA

Text classification is the most fundamental and essential task in natural language processing. The last decade has seen a surge of research in this area due to the unprecedented success of deep learning. Numerous methods, datasets, and evaluation metrics have been proposed in the literature, raising the need for a comprehensive and updated survey. This paper fills the gap by reviewing the state-of-the-art approaches from 1961 to 2021, focusing on models from traditional models to deep learning. We create a taxonomy for text classification according to the text involved and the models used for feature extraction and classification. We then discuss each of these categories in detail, dealing with both the technical developments and benchmark datasets that support tests of predictions. A comprehensive comparison between different techniques, as well as identifying the pros and cons of various evaluation metrics are also provided in this survey. Finally, we conclude by summarizing key implications, future research directions, and the challenges facing the research area.

Additional Key Words and Phrases: deep learning, traditional models, text classification, evaluation metrics, challenges.

ACM Reference Format:
Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021. A Survey on Text Classification: From Traditional to Deep Learning. *ACM Trans. Intell. Syst. Technol.* 37, 4, Article 111 (April 2021), 39 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Text classification – the procedure of designating pre-defined labels for text – is an essential and significant task in many Natural Language Processing (NLP) applications, such as sentiment analysis [1] [2], topic labeling [3] [4], question answering [5] [6] and dialog act classification [7]. In the

*Corresponding author

Authors' addresses: Qian Li, Beihang University, Haidian, Beijing, China, liqian@act.buaa.edu.cn; Hao Peng, Beihang University, Haidian, Beijing, China, penghao@act.buaa.edu.cn; Jianxin Li, Beihang University, Haidian, Beijing, China, lijx@act.buaa.edu.cn; Congying Xia, University of Illinois at Chicago, Chicago, IL, USA, cxia8@uic.edu; Renyu Yang, University of Leeds, Leeds, England, UK, r.yang1@leeds.ac.uk; Lichao Sun, Lehigh University, Bethlehem, PA, USA, james.lichao.sun@gmail.com; Philip S. Yu, University of Illinois at Chicago, Chicago, IL, USA, psyu@uic.edu; Lifang He, Lehigh University, Bethlehem, PA, USA, lih319@lehigh.edu.

2157-6904/2021/4-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

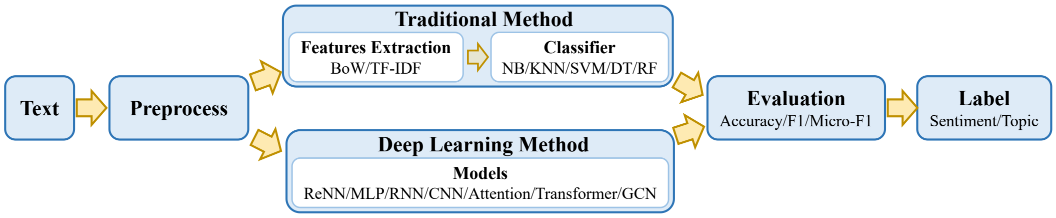


Fig. 1. Flowchart of the text classification with classic methods in each module. It is crucial to extract essential features for traditional methods, but features can be extracted automatically by deep learning methods.

era of information explosion, it is time-consuming and challenging to process and classify large amounts of text data manually. Besides, the accuracy of manual text classification can be easily influenced by human factors, such as fatigue and expertise. It is desirable to use machine learning methods to automate the text classification procedure to yield more reliable and less subjective results. Moreover, this can also help enhance information retrieval efficiency and alleviate the problem of information overload by locating the required information.

Fig. 1 illustrates a flowchart of the procedures involved in the text classification, under the light of traditional and deep analysis. Text data is different from numerical, image, or signal data. It requires NLP techniques to be processed carefully. The first important step is to preprocess text data for the model. Traditional models usually need to obtain good sample features by artificial methods and then classify them with classic machine learning algorithms. Therefore, the effectiveness of the method is largely restricted by feature extraction. However, different from traditional models, deep learning integrates feature engineering into the model fitting process by learning a set of nonlinear transformations that serve to map features directly to outputs.

From the 1960s until the 2010s, traditional text classification models dominated. Traditional methods mean statistics-based models, such as Naïve Bayes (NB) [8], K-Nearest Neighbor (KNN) [9], and Support Vector Machine (SVM) [10]. Comparing with the earlier rule-based methods, this method has obvious advantages in accuracy and stability. However, these approaches still need to do feature engineering, which is time-consuming and costly. Besides, they usually disregard the natural sequential structure or contextual information in textual data, making it challenging to learn the semantic information of the words. Since the 2010s, text classification has gradually changed from traditional models to deep learning models. Compared with the methods based on traditional, deep learning methods avoid designing rules and features by humans and automatically provide semantically meaningful representations for text mining. Therefore, most of the text classification research works are based on Deep Neural Networks (DNNs) [11], which are data-driven approaches with high computational complexity. Few works focus on traditional models to settle the limitations of computation and data.

1.1 Major Differences and Contributions

There have been several works reviewing text classification and its subproblems recently. Two of them are reviews of text classification. Kowsari et al. [12] surveyed different text feature extraction, dimensionality reduction methods, basic model structure for text classification, and evaluation methods. Minaee et al. [13] reviewed recent deep learning based text classification methods, benchmark datasets, and evaluation metrics. Unlike existing text classification reviews, we conclude existing models from traditional models to deep learning with works of recent years. Traditional models emphasize the feature extraction and classifier design. Once the text has well-designed characteristics, it can be quickly converged by training the classifier. DNNs can perform feature extraction automatically and learn well without domain knowledge. We then give the datasets and evaluation metrics for single-label and multi-label tasks and summarize future research challenges from data, models, and performance perspective. Moreover, we summarize various information in three tables, including the necessary information of classic deep learning models, primary information of main datasets, and a general benchmark of state-of-the-art methods under different applications. In summary, this study's main contributions are as follows:

- We introduce the process and development of text classification and present comprehensive analysis and research on primary models – from traditional to deep learning models – according to their model structures. We summarize the necessary information of deep learning models in terms of basic model structures in Table 1, including publishing years, methods, venues, applications, evaluation metrics, datasets and code links.
- We introduce the present datasets and give the formulation of main evaluation metrics with the comparison of metrics, including single-label and multi-label text classification tasks. We summarize the necessary information of primary datasets in Table 2, including the number of categories, average sentence length, the size of each dataset, related papers and data addresses.
- We summarize classification accuracy scores of models given in their articles, on benchmark datasets in Table 4 and conclude the survey by discussing the main challenges facing the text classification and key implications stemming from this study.

1.2 Organization of the Survey

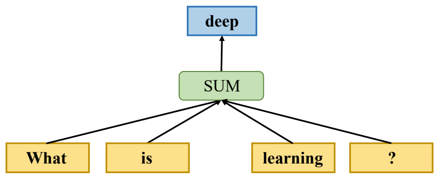
The rest of the survey is organized as follows. Section 2 summarizes the existing models related to text classification, including traditional and deep learning models, including a summary table. Section 3 introduces the primary datasets with a summary table and evaluation metrics on single-label and multi-label tasks. We then give quantitative results of the leading models in classic text classification datasets in Section 4. Finally, we summarize the main challenges for deep learning text classification in Section 5 before concluding the article in Section 6.

2 TEXT CLASSIFICATION METHODS

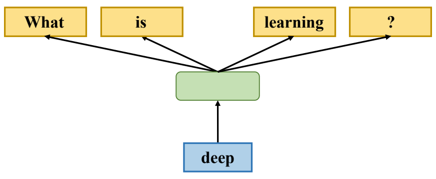
Text classification is referred to as extracting features from raw text data and predicting the categories of text data based on such features. Numerous models have been proposed in the past few decades for text classification. For traditional models, NB [8] is the first model used for the text classification task. Whereafter, generic classification models are proposed, such as KNN [9], SVM [10], and Random Forest (RF) [14], which are called classifiers, widely used for text classification. Recently, the eXtreme Gradient Boosting (XGBoost) [15] and the Light Gradient Boosting Machine (LightGBM) [16] have arguably the potential to provide excellent performance. For deep learning models, TextCNN [17] has the highest number of references in these models, wherein a Convolutional Neural Network (CNN) [18] model has been introduced to solve the text classification problem for the first time. While not specifically designed for handling text classification tasks, the Bidirectional Encoder Representation from Transformers (BERT) [19] has been widely employed when designing text classification models, considering its effectiveness on numerous text classification datasets.

2.1 Traditional Models

Traditional models accelerate text classification with improved accuracy and make the application scope of traditional expand. The first thing is to preprocess the raw input text for training traditional models, which generally consists of word segmentation, data cleaning, and statistics.



(a) CBOW.



(b) Skip-gram.

Fig. 2. The structure of word2vec, including CBOW and Skip-gram.

Then, text representation aims to express preprocessed text in a form that is much easier for computers and minimizes information loss, such as Bag-Of-Words (BOW) [20], N-gram [21], Term Frequency-Inverse Document Frequency (TF-IDF) [22], word2vec [23] and Global Vectors for word representation (GloVe) [24]. BOW means that all the words in the corpus are formed into a mapping array. According to the mapping array, a sentence can be represented as a vector. The i-th element in the vector represents the frequency of the i-th word in the mapping array of the sentence. The vector is the BOW of the sentence. At the core of the BOW is representing each text with a dictionary-sized vector. The individual value of the vector denotes the word frequency corresponding to its inherent position in the text. Compared to BOW, N-gram considers the information of adjacent words and builds a dictionary by considering the adjacent words. It is used to calculate the probability model of a sentence. The probability of a sentence is expressed as the joint probability of each word in the sentence. The probability of a sentence can be calculated by predicting the probability of the N-th word, given the sequence of the $(N-1)$-th words. To simplify the calculation, the N-gram model adopts the Markov hypothesis [21]. A word appears only concerning the words that preceded it. Therefore, the N-gram model performs a sliding window with size N. By counting and recording the occurrence frequency of all fragments, the probability of a sentence can be calculated using the frequency of relevant fragments in the record. TF-IDF [22] uses the word frequency and inverses the document frequency to model the text. TF is the word frequency of a word in a specific article, and IDF is the reciprocal of the proportion of the articles containing this word to the total number of articles in the corpus. TF-IDF is the multiplication of the two. TF-IDF assesses the importance of a word to one document in a set of files or a corpus. The importance of a word increases proportionally with the number of times it appears in a document. However, it decreases inversely with its frequency in the corpus as a whole. The word2vec [23] employs local context information to obtain word vectors, as shown in Fig. **??**. Word vector refers to a fixed-length real value vector specified as the word vector for any word in the corpus. The word2vec uses two essential models: CBOW and Skip-gram. The former is to predict the current word on the premise that the context of the current word is known. The latter is to predict the context when the current word is known. The GloVe [24] – with both the local context and global statistical features – trains on the nonzero elements in a word-word co-occurrence matrix, as shown in Fig. **??**. It enables word vectors to contain as much semantic and grammatical information as possible. The construction method of word vector is: firstly, the co-occurrence matrix of words is constructed based on the corpus, and then the word vector is learned based on the co-occurrence matrix and GloVe model. Finally, the represented text is fed into the classifier according to selected features. Here, we discuss some representative classifiers in detail:

2.1.1 PGM-based methods. Probabilistic Graphical Models (PGMs) express the conditional dependencies among features in graphs, such as the Bayesian network [25]. It is combinations of probability theory and graph theory.

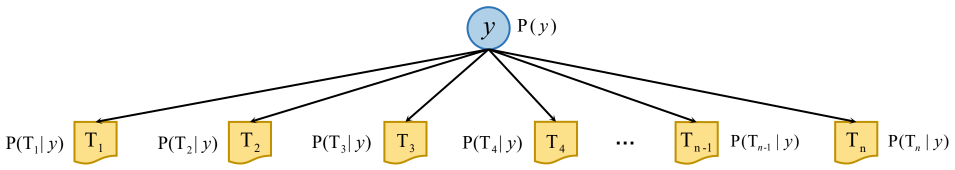


Fig. 3. The structure of Naïve Bayes.

Naïve Bayes (NB) [8] is the simplest and most broadly used model based on applying Bayes' theorem. The NB algorithm has an independent assumption: when the target value has been given, the conditions between text $T = [T_1, T_2, \cdots, T_n]$ are independent (see Fig. 3). The NB algorithm primarily uses the prior probability to calculate the posterior probability $\mathrm{P}(y \mid \mathrm{T}_1, T_2, \cdots, T_n) = \frac{p(y)\prod_{j=1}^{n} p(T_j|y)}{\prod_{j=1}^{n} p(T_j)}$. Due to its simple structure, NB is broadly used for text classification tasks. Although the assumption that the features are independent is sometimes not actual, it substantially simplifies the calculation process and performs better. To improve the performance on smaller categories, Schneider [26] proposes a feature selection score method through calculating KL-divergence [27] between the training set and corresponding categories for multinomial NB text classification. Dai et al. [28] propose a transfer learning method named Naive Bayes Transfer Classification (NBTC) to settle the different distribution between the training set and the target set. It uses the EM algorithm [29] to obtain a locally optimal posterior hypothesis on the target set. NB classifier is also used for fake news detection [30], and sentiment analysis [31], which can be seen as a text classification task. Bernoulli NB, Gaussian NB and Multinomial NB are three popular approaches of NB text classification [32]. Multinomial NB performs slightly better than Bernoulli NB on few labeled dataset[33]. Bayesian NB classifier with Gaussian event model [32] has been proven to be superior to NB with multinomial event model on 20 Newsgroups (20NG) [34] and WebKB [35] datasets.

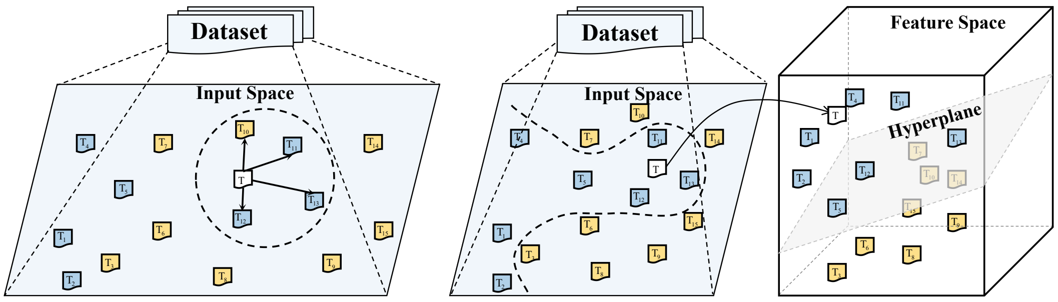


Fig. 4. The structure of KNN where $k = 4$ (left) and the structure of SVM (right). Each node represents a text and nodes with different contours represent different categories.

2.1.2 *KNN-based Methods.*

At the core of the K-Nearest Neighbors (KNN) algorithm [9] is to classify an unlabeled sample by finding the category with most samples on the k nearest samples. It is a simple classifier without building the model and can decrease complexity through the fasting process of getting k nearest neighbors. Fig. 4 showcases the structure of KNN. We can find k training texts approaching a specific text to be classified through estimating the in-between distance. Hence, the text can be divided into the most common categories found in k training set texts. The improvement of KNN algorithm mainly includes feature similarity [36], K value [37] and index optimization [38]. However, due to the positive correlation between model time/space complexity and the amount of data, the KNN algorithm takes an unusually long time on the large-scale datasets [39]. To decrease the number of selected features, Soucy et al. [40] propose a KNN algorithm without feature weighting. It manages to find relevant features, building the inter-dependencies of words by using a feature selection. When the data is extremely unevenly distributed, KNN tends to classify samples with more data. The Neighbor-Weighted K-Nearest Neighbor (NWKNN) [41] is proposed to improve classification performance on the unbalanced corpora. It casts a significant weight for neighbors in a small category and a small weight for neighbors in a broad class.

2.1.3 SVM-based Methods. Cortes and Vapnik [42] propose Support Vector Machine (SVM) to tackle the binary classification of pattern recognition. Joachims [10], for the first time, uses the SVM method for text classification representing each text as a vector. As illustrated in Fig. 4, SVM-based approaches turn text classification tasks into multiple binary classification tasks. In this context, SVM constructs an optimal hyperplane in the one-dimensional input space or feature space, maximizing the distance between the hyperplane and the two categories of training sets, thereby achieving the best generalization ability. The goal is to make the distance of the category boundary along the direction perpendicular to the hyperplane is the largest. Equivalently, this will result in the lowest error rate of classification. Constructing an optimal hyperplane can be transformed into a quadratic programming problem to obtain a globally optimal solution. Choosing the appropriate kernel function [43] and feature selection [44] are of the utmost importance to ensure SVM can deal with nonlinear problems and become a robust nonlinear classifier. Furthermore, active learning [45] and adaptive learning [46] method are used for text classification to reduce the labeling effort based on the supervised learning algorithm SVM. To analyze what the SVM algorithms learn and what tasks are suitable, Joachims [47] proposes a theoretical learning model combining the statistical traits with the generalization performance of an SVM, analyzing the features and benefits using a quantitative approach. Transductive Support Vector Machine (TSVM) [48] is proposed to lessen misclassifications of the particular test collections with a general decision function considering a specific test set. It uses prior knowledge to establish a more suitable structure and study faster.

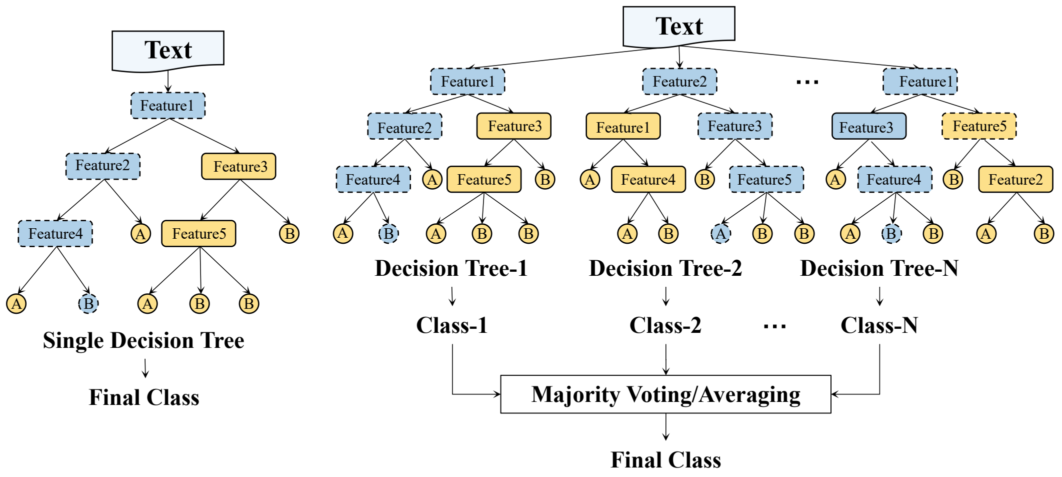


Fig. 5. An example of DT (left) and the structure of RF (right). The nodes with the dotted outline represent the nodes of the decision route. It has five features to predict whether each text belongs to category A or B.

2.1.4 DT-based Methods. Decision Trees (DT) [49] is a supervised tree structure learning method – reflective of the idea of divide-and-conquer – and is constructed recursively. It learns disjunctive expressions and has robustness for the text with noise. As shown in Fig. 5, decision trees can be generally divided into two distinct stages: tree construction and tree pruning. It starts at the root node and tests the data samples (composed of instance sets, which have several attributes), and divides the dataset into diverse subsets according to different results. A subset of datasets constitutes a child node, and every leaf node in the decision tree represents a category. Constructing the decision tree is to determine the correlation between classes and attributes, further exploited to predict the record categories of unknown forthcoming types. The classification rules generated by the decision tree algorithm are straight-forward, and the pruning strategy [50] can also help reduce the influence of noise. Its limitation, however, mainly derives from inefficiency in coping with explosively increasing data size. More specifically, the Iterative Dichotomiser 3 (ID3) [51] algorithm uses information gain as the attribute selection criterion in the selection of each node – It is used to select the attribute of each branch node, and then select the attribute having the maximum information gain value to become the discriminant attribute of the current node. Based on ID3, C4.5 [52] learns to obtain a map from attributes to classes, which effectively classifies entities unknown to new categories. DT based algorithms usually need to train for each dataset, which is low efficiency [53]. Thus, Johnson et al. [54] propose a DT-based symbolic rule system. The method represents each text as a vector calculated by the frequency of each word in the text, and induces rules from the training data. The learning rules are used for classifying the other data, being similar to the training data. Furthermore, to reduce the computational costs of DT algorithms, Fast Decision-Tree (FDT) [55] uses a two-pronged strategy: pre-selecting a feature set and training multiple DTs on different data subsets. Results from multiple DTs are combined through a data-fusion technique to resolve the cases of imbalanced classes.

2.1.5 Integration-based Methods. Integrated algorithms aim to aggregate the results of multiple algorithms for better performance and interpretation. Conventional integrated algorithms are bootstrap aggregation, such as RF [14], boosting such as the Adaptive Boosting (AdaBoost) [56], and XGBoost [15] and stacking. The bootstrap aggregation method trains multiple classifiers without strong dependencies and then aggregates their results. For instance, RF [14] consists of multiple tree classifiers wherein all trees depend on the value of the random vector sampled independently (depicted in Fig. 5). It is worth noting that each tree within the RF shares the same distribution. The generalization error of an RF relies on the strength of each tree and the relationship among trees, and will converge to a limit with the increment of tree number in the forest. In boosting based algorithms, all labeled data are trained with the same weight to initially obtain a weaker classifier [57]. The weights of the data will then be adjusted according to the former result of the classifier. The training procedure will continue by repeating such steps until the termination condition is reached. Unlike bootstrap and boosting algorithms, stacking based algorithms break down the data into n parts and use n classifiers to calculate the input data in a cascade manner – Result from upstream classifier will feed into the downstream classifier as input. The training will terminate once a pre-defined iteration number is targeted. The integrated method can capture more features from multiple trees. However, it helps little for short text. Motivated by this, Bouaziz et al. [58] combine data enrichment – with semantics in RFs for short text classification – to overcome the deficiency of sparseness and insufficiency of contextual information. In integrated algorithms, not all classifiers learn well. It is necessary to give different weights for each classifier. To differentiate contributions of trees in a forest, Islam et al. [59] exploit the Semantics Aware Random Forest (SARF) classifier, choosing features similar to the features of the same class, for extracting features and producing the prediction values.

Summary. The parameters of NB are more diminutive, less sensitive to missing data, and the algorithm is simple. However, it assumes that features are independent of each other. When the number of features is large, or the correlation between features is significant, the performance of NB decreases. SVM can solve high-dimensional and nonlinear problems. It has a high generalization ability, but it is sensitive to missing data. KNN mainly depends on the surrounding finite adjacent samples, rather than discriminating class domain to determine the category. Thus, for the dataset to be divided with more crossover or overlap of the class domain, it is more suitable than other methods. DT is easy to understand and interpret. Given an observed model, it is easy to deduce the corresponding logical expression according to the generated decision tree. The traditional method is a type of machine learning. It learns from data, which are pre-defined features that are important to the performance of prediction values. However, feature engineering is tough work. Before training the classifier, we need to collect knowledge or experience to extract features from the original text. The traditional methods train the initial classifier based on various textual features extracted from the raw text. Toward small datasets, traditional models usually present better performance than deep learning models under the limitation of computational complexity. Therefore, some researchers have studied the design of traditional models for specific domains with fewer data.

2.2 Deep Learning Models

The DNNs consist of artificial neural networks that simulate the human brain to automatically learn high-level features from data, getting better results than traditional models in speech recognition, image processing, and text understanding. Input datasets should be analyzed to classify the data, such as a single-label, multi-label, unsupervised, unbalanced dataset. According to the trait of the dataset, the input word vectors are sent into the DNN for training until the termination condition is reached. The performance of the training model is verified by the downstream task, such as sentiment classification, question answering, and event prediction. We show some DNNs over the years in Table 1, including designs that are different from the corresponding basic models, evaluation metrics, and experimental datasets.

Numerous deep learning models have been proposed in the past few decades for text classification, as shown in Table 1. We tabulate primary information – including publication years, venues, applications, code links, evaluation metrics, and experiment datasets – of main deep learning models for text classification. The applications in this table include Sentiment Analysis (SA), Topic Labeling (TL), News Classification (NC), Question Answering (QA), Dialog Act Classification (DAC), Natural Language Inference (NLI) and Relation Classification (RC). The multilayer perceptron [172] and the recursive neural network [173] are the first two deep learning approaches used for the text classification task, which improve performance compared with traditional models. Then, CNNs, Recurrent Neural Networks (RNNs), and attention mechanisms are used for text classification [101] [174] [175]. Many researchers advance text classification performance for different tasks by improving CNN, RNN, and attention, or model fusion and multi-task methods. The appearance of BERT [19], which can generate contextualized word vectors, is a significant turning point in the development of text classification and other NLP technologies. Many researchers [176] [142] have studied text classification models based on BERT, which achieves better performance than the above models in multiple NLP tasks, including text classification. Besides, some researchers study text classification technology based on Graph Neural Network (GNN) [155] [177] to capture structural information in the text, which cannot be replaced by other methods. Here, we classify DNNs by structure and discuss some representative models in detail:

2.2.1 ReNN-based Methods. Traditional models cost lots of time on design features for each task. Furthermore, in the case of deep learning, the meaning of "word vectors" is different: each input word is associated with a fixed-length vector whose values are either drawn at random or derived from a previous traditional process, thus forming a matrix L called word embedding matrix

Table 1. Basic information based on different models. Trans: Transformer. Time: training time.

Model	Year	Method	Venue	Applications	Code Link	Metrics	Datasets
ReNN	2011	RAE [60]	EMNLP	SA, QA	[61]	Accuracy	MPQA, MR, EP
	2012	MV-RNN [62]	EMNLP	SA	[63]	Accuracy, F1	MR
	2013	RNTN [64]	EMNLP	SA	[65]	Accuracy	SST
	2014	DeepRNN [66]	NIPS	SA;QA	-	Accuracy	SST-1;SST-2
MLP	2014	Paragraph-Vec [67]	ICML	SA, QA	[68]	Error Rate	SST, IMDB
	2015	DAN [69]	ACL	SA, QA	[70]	Accuracy, Time	RT, SST, IMDB
RNN	2015	Tree-LSTM [1]	ACL	SA	[71]	Accuracy	SST-1, SST-2
	2015	S-LSTM [2]	ICML	SA	-	Accuracy	SST
	2015	TextRCNN [72]	AAAI	SA, TL	[73]	Macro-F1, etc.	20NG, Fudan, ACL, SST-2
	2015	MT-LSTM [6]	EMNLP	SA,QA	[74]	Accuracy	SST-1, SST-2, QC, IMDB
	2016	oh-2LSTMp [75]	ICML	SA, TL	[76]	Error Rate	IMDB, Elec, RCV1, 20NG
	2016	BLSTM-2DCNN [77]	COLING	SA, QA, TL	[78]	Accuracy	SST-1, Subj, TREC, etc.
	2016	Multi-Task [79]	IJCAI	SA	[80]	Accuracy	SST-1, SST-2, Subj, IMDB
	2017	DeepMoji [81]	EMNLP	SA	[82]	Accuracy	SS-Twitter, SE1604, etc.
	2017	TopicRNN [83]	ICML	SA	[84]	Error Rate	IMDB
	2017	Miyato et al. [85]	ICLR	SA	[86]	Error Rate	IMDB, DBpedia, etc.
	2018	RNN-Capsule [87]	TheWebConf	SA	[88]	Accuracy	MR, SST-1, etc.
	2019	HM-DenseRNNs [89]	IJCAI	SA, TL	[90]	Accuracy	IMDB, SST-5, AG
CNN	2014	TextCNN [17]	EMNLP	SA, QA	[91]	Accuracy	MR, SST-2, Subj, etc.
	2014	DCNN [5]	ACL	SA, QA	[92]	Accuracy	MR, TREC, Twitter
	2015	CharCNN [93]	NeurIPS	SA, QA, TL	[94]	Error Rate	AG, Yelp P, DBPedia, etc.
	2016	SeqTextRCNN [7]	NAACL	Dialog act	[95]	Accuracy	DSTC 4, MRDA, SwDA
	2017	XML-CNN [96]	SIGIR	NC, TL, SA	[97]	NDCG@K, etc.	EUR-Lex, Wiki-30K, etc.
	2017	DPCNN [98]	ACL	SA, TL	[99]	Error Rate	AG, DBPedia, Yelp.P, etc.
	2017	KPCNN [100]	IJCAI	SA, QA, TL	-	Accuracy	Twitter, AG, Bing, etc.
	2018	TextCapsule [101]	EMNLP	SA, QA, TL	[102]	Accuracy	Subj, TREC, Reuters, etc.
	2018	HFT-CNN [103]	EMNLP	TL	[104]	Micro-F1, etc.	RCV1, Amazon670K
	2019	CCRCNN [105]	AAAI	TL	-	Accuracy	TREC, MR, AG
	2020	Bao et al. [106]	ICLR	TL	[107]	Accuracy	20NG, Reuters-2157, etc.
Attention	2016	HAN [108]	NAACL	SA, TL	[109]	Accuracy	Yelp.F, YahooA, etc.
	2016	BI-Attention [110]	NAACL	SA	-	Accuracy	NLP&CC 2013 [111]
	2016	LSTMN [112]	EMNLP	SA	[113]	Accuracy	SST-1
	2017	Lin et al. [114]	ICLR	SA	[115]	Accuracy	Yelp, SNLI Age
	2018	SGM [116]	COLING	TL	[117]	HL, Micro-F1	RCV1-V2, AAPD
	2018	ELMo [118]	NAACL	SA, QA, NLI	[119]	Accuracy	SQuAD, SNLI, SST-5
	2018	BiBloSA [120]	ICLR	SA	[121]	Accuracy, Time	CR, MPQA, SUBJ, etc.
	2019	AttentionXML [122]	NeurIPS	TL	[123]	P@k, N@k, etc.	EUR-Lex, etc.
	2019	HAPN [124]	EMNLP	RC	-	Accuracy	FewRel, CSID
	2019	Proto-HATT [125]	AAAI	RC	[126]	Accuracy	FewRel
	2019	STCKA [127]	AAAI	SA, TL	[128]	Accuracy	Weibo, Product Review, etc.
	2020	HyperGAT [129]	EMNLP	TL, NC	[130]	Accuracy	20NG, Ohsumed, MR, etc.
	2020	MSMSA [131]	AAAI	ST, QA, NLI	-	Accuracy, F1	IMDB, MR, SST, SNLI, etc.
	2020	Choi [132]	EMNLP	SA, TL	-	Accuracy	SST2, IMDB, 20NG
Trans	2019	BERT [19]	NAACL	SA, QA	[133]	Accuracy	SST-2, QQP, QNLI, CoLA
	2019	BERT-BASE [134]	ACL	TL	[135]	P@K, R@K, etc.	EUR-LEX
	2019	Sun et al. [136]	CCL	SA, QA, TL	[137]	Error Rate	TREC, DBPedia, etc.
	2019	XLNet [138]	NeurIPS	SA, QA, NC	[139]	EM, F1, etc.	Yelp-2, AG, MNLI, etc.
	2019	RoBERTa [140]	arXiv	SA, QA	[141]	F1, Accuracy	SQuAD, MNLI-m, SST-2
	2020	GAN-BERT [142]	ACL	SA, NLI	[143]	F1, Accuracy	SST-5, MNLI
	2020	BAE [144]	EMNLP	SA, QA	[145]	Accuracy	Amazon, Yelp, MR, MPQA
	2020	ALBERT [146]	ICLR	SA, QA	[147]	F1, Accuracy	SST, MNLI, SQuAD
	2020	TG-Transformer [148]	EMNLP	SA, TL	-	Accuracy, Time	R8, R52, Ohsumed, etc.
	2020	X-Transformer [149]	KDD	SA, TL	[150]	P@K, R@K	Eurlex-4K, Wiki10-31K, etc.
	2021	LightXML [151]	arXiv	TL, ML, NLI	[152]	P@K, Time	AmazonCat-13K, etc.
GNN	2018	DGCNN [153]	TheWebConf	TL	[154]	Macro-F1, etc.	RCV1, NYTimes
	2019	TextGCN [155]	AAAI	SA, TL	[156]	Accuracy	20NG, Ohsumed, R52, etc.
	2019	SGC[157]	ICML	NC, TL, SA	[158]	Accuracy, Time	20NG, R8, Ohsumed, etc.
	2019	Huang et al. [159]	EMNLP	NC, TL	[160]	Accuracy	R8, R52, Ohsumed
	2019	Peng et al. [161]	arXiv	NC, TL	-	Micro-F1, etc.	RCV1, EUR-Lex, etc.
	2020	TextING [162]	ACL	SA, NC, TL	[163]	Accuracy	MR, R8, R52, Ohsumed
	2020	TensorGCN [164]	AAAI	SA, NC, TL	[165]	Accuracy	20NG, R8, R52, Ohsumed, MR
	2020	MAGNET [166]	ICAART	TL	[167]	Micro-F1, HL	Reuters, RCV1-V2, etc.
Others	2017	Miyato et al. [85]	ICLR	SA, NC	[168]	Error Rate	IMDB, RCV1, et al.
	2018	TMN [169]	EMNLP	TL	-	Accuracy, F1	Snippets, Twitter, et al.
	2019	Zhang et al. [170]	NAACL	TL, NC	[171]	Accuracy	DBpedia, 20NG.

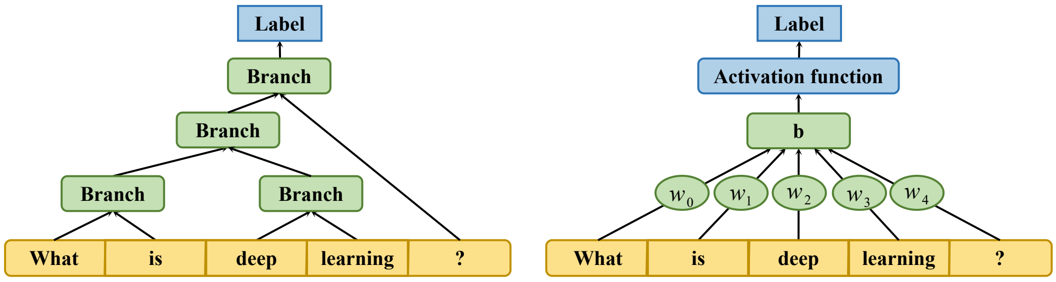


Fig. 6. The architecture of ReNN (left) and the architecture of MLP (right).

which represents the vocabulary words in a small latent semantic space, of generally 50 to 300 dimensions. The Recursive Neural Network (ReNN) [173] can automatically learn the semantics of text recursively and the syntax tree structure without feature design, as shown in Fig. 6. We give an example of ReNN based models. First, each word of input text is taken as the leaf node of the model structure. Then all nodes are combined into parent nodes using a weight matrix. The weight matrix is shared across the whole model. Each parent node has the same dimension with all leaf nodes. Finally, all nodes are recursively aggregated into a parent node to represent the input text to predict the label.

ReNN-based models improve performance compared with traditional models and save on labor costs due to excluding feature designs used for different text classification tasks. The Recursive AutoEncoder (RAE) [60] is used to predict the distribution of sentiment labels for each input sentence and learn the representations of multi-word phrases. To learn compositional vector representations for each input text, the Matrix-Vector Recursive Neural Network (MV-RNN) [62] introduces a ReNN model to learn the representation of phrases and sentences. It allows that the length and type of input texts are inconsistent. MV-RNN allocates a matrix and a vector for each node on the constructed parse tree. Furthermore, the Recursive Neural Tensor Network (RNTN) [64] is proposed with a tree structure to capture the semantics of sentences. It inputs phrases with different length and represents the phrases by parse trees and word vectors. The vectors of higher nodes on the parse tree are estimated by the equal tensor-based composition function. For RNTN, the time complexity of building the textual tree is high, and expressing the relationship between documents is complicated within a tree structure. The performance is usually improved, with the depth being increased for DNNs. Therefore, Irsoy et al. [66] propose a Deep Recursive Neural Network (DeepReNN), which stacks multiple recursive layers. It is built by binary parse trees and learns distinct perspectives of compositionality in language.

2.2.2 MLP-based Methods. A MultiLayer Perceptron (MLP) [172], sometimes colloquially called "vanilla" neural network, is a simple neural network structure that is used for capturing features automatically. As shown in Fig. 6, we show a three-layer MLP model. It contains an input layer, a hidden layer with an activation function in all nodes, and an output layer. Each node connects with a certain weight w_i. It treats each input text as a bag of words and achieves high performance on many text classification benchmarks comparing with traditional models.

There are some MLP-based methods proposed by some research groups for text classification tasks. The Paragraph Vector (Paragraph-Vec) [67] is the most popular and widely used method, which is similar to the Continuous Bag-Of-Words (CBOW) [23]. It gets fixed-length feature representations of texts with various input lengths by employing unsupervised algorithms. Comparing with CBOW, it adds a paragraph token mapped to the paragraph vector by a matrix. The model predicts the fourth word by the connection or average of this vector to the three contexts of the word. Paragraph vectors can be used as a memory for paragraph themes and are used as a paragraph function and inserted into the prediction classifier.

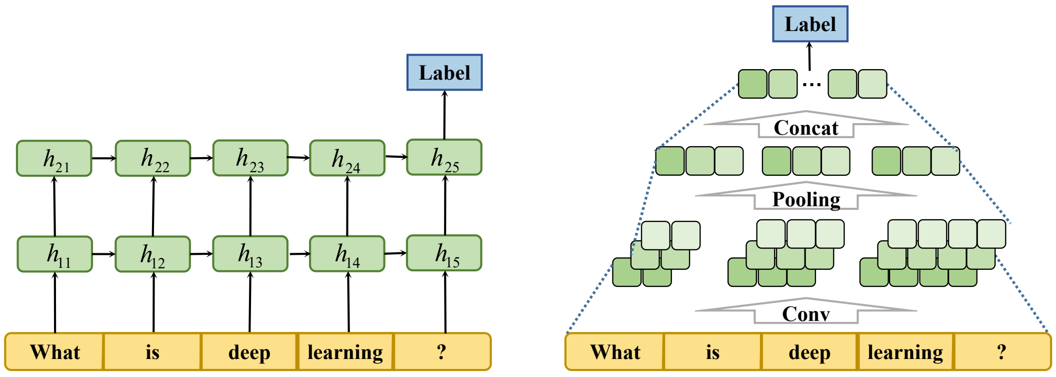


Fig. 7. The RNN based model (left) and the CNN based model (right).

2.2.3 *RNN-based Methods.* The Recurrent Neural Network (RNN) [173] is broadly used for capturing long-range dependency through recurrent computation. The RNN language model learns historical information, considering the location information among all words suitable for text classification tasks. We show an RNN model for text classification with a simple sample, as shown in Fig. 7. Firstly, each input word is represented by a specific vector using a word embedding technology. Then, the embedding word vectors are fed into RNN cells one by one. The output of RNN cells are the same dimension with the input vector and are fed into the next hidden layer. The RNN shares parameters across different parts of the model and has the same weights of each input word. Finally, the label of input text can be predicted by the last output of the hidden layer.

To diminish the time complexity of the model and capture contextual information, Liu et al. [79] introduce a model for catching the semantics of long texts. It is a biased model that parsed the text one by one, making the following inputs profit over the former and decreasing the semantic efficiency of capturing the whole text. For modeling topic labeling tasks with long input sequences, TopicRNN [83] is proposed. It captures the dependencies of words in a document via latent topics and uses RNNs to capture local dependencies and latent topic models for capturing global semantic dependencies. Virtual Adversarial Training (VAT) [178] is a useful regularization method applicable to semi-supervised learning tasks. Miyato et al. [85] apply adversarial and virtual adversarial training text and employ the perturbation into word embedding rather than the original input text. The model improves the quality of the word embedding and is not easy to overfit during training. Capsule network [179] captures the relationships between features using dynamic routing between capsules comprised of a group of neurons in a layer. Wang et al. [87] propose an RNN-Capsule model with a simple capsule structure for the sentiment classification task.

In the backpropagation process of RNN, the weights are adjusted by gradients, calculated by continuous multiplications of derivatives. If the derivatives are extremely small, it may cause a gradient vanishing problem by continuous multiplications. Long Short-Term Memory (LSTM) [180], the improvement of RNN, effectively alleviates the gradient vanishing problem. It is composed of a cell to remember values on arbitrary time intervals and three gate structures to control information flow. The gate structures include input gates, forget gates, and output gates. The LSTM classification method can better capture the connection among context feature words, and use the forgotten gate structure to filter useless information, which is conducive to improving the total capturing ability of the classifier. Tree-LSTM [1] extends the sequence of LSTM models to the tree structure. The whole subtree with little influence on the result can be forgotten through the LSTM forgetting gate mechanism for the Tree-LSTM model.

Natural Language Inference (NLI) [181] predicts whether one text's meaning can be deduced from another by measuring the semantic similarity between each pair of sentences. To consider other granular matchings and matchings in the reverse direction, Wang et al. [182] propose a model for the NLI task named Bilateral Multi-Perspective Matching (BiMPM). It encodes input sentences by the BiLSTM encoder. Then, the encoded sentences are matched in two directions. The results are aggregated in a fixed-length matching vector by another BiLSTM layer. Finally, the result is evaluated by a fully connected layer.

2.2.4 CNN-based Methods.

Convolutional Neural Networks (CNNs) [18] are proposed for image classification with convolving filters that can extract features of pictures. Unlike RNN, CNN can simultaneously apply convolutions defined by different kernels to multiple chunks of a sequence. Therefore, CNNs are used for many NLP tasks, including text classification. For text classification, the text requires being represented as a vector similar to the image representation, and text features can be filtered from multiple angles, as shown in Fig. 7. Firstly, the word vectors of the input text are spliced into a matrix. The matrix is then fed into the convolutional layer, which contains several filters with different dimensions. Finally, the result of the convolutional layer goes through the pooling layer and concatenates the pooling result to obtain the final vector representation of the text. The category is predicted by the final vector.

To try using CNN for the text classification task, an unbiased model of convolutional neural networks is introduced by Kim, called TextCNN [17]. It can better determine discriminative phrases in the max-pooling layer with one layer of convolution and learn hyperparameters except for word vectors by keeping word vectors static. Training only on labeled data is not enough for data-driven deep models. Therefore, some researchers consider utilizing unlabeled data. Johnson et al. [183] propose a CNN model based on two-view semi-supervised learning for text classification, which first uses unlabeled data to train the embedding of text regions and then labeled data. DNNs usually have better performance, but it increases the computational complexity. Motivated by this, a Deep Pyramid Convolutional Neural Network (DPCNN) [98] is proposed, with a little more computational accuracy, increasing by raising the network depth. The DPCNN is more specific than Residual Network (ResNet) [184], as all the shortcuts are exactly simple identity mappings without any complication for dimension matching.

According to the minimum embedding unit of text, embedding methods are divided into character-level, word-level, and sentence-level embedding. Character-level embeddings can settle Out-Of-Vocabulary (OOV) [185] words. Word-level embeddings learn the syntax and semantics of the words. Moreover, sentence-level embedding can capture relationships among sentences. Motivated by these, Nguyen et al. [186] propose a deep learning method based on a dictionary, increasing information for word-level embeddings through constructing semantic rules and deep CNN for character-level embeddings. Adams et al. [187] propose a character-level CNN model, called MGTC, to classify multi-lingual texts written. TransCap [188] is proposed to encapsulate the sentence-level semantic representations into semantic capsules and transfer document-level knowledge.

RNN based models capture the sequential information to learn the dependency among input words, and CNN based models extract the relevant features from the convolution kernels. Thus some works study the fusion of the two methods. BLSTM-2DCNN [77] integrates a Bidirectional LSTM (BiLSTM) with two-dimensional max pooling. It uses a 2D convolution to sample more meaningful information of the matrix and understands the context better through BiLSTM. Moreover, Xue et al. [189] propose MTNA, a combination of BiLSTM and CNN layers, to solve aspect category classification and aspect term extraction tasks.

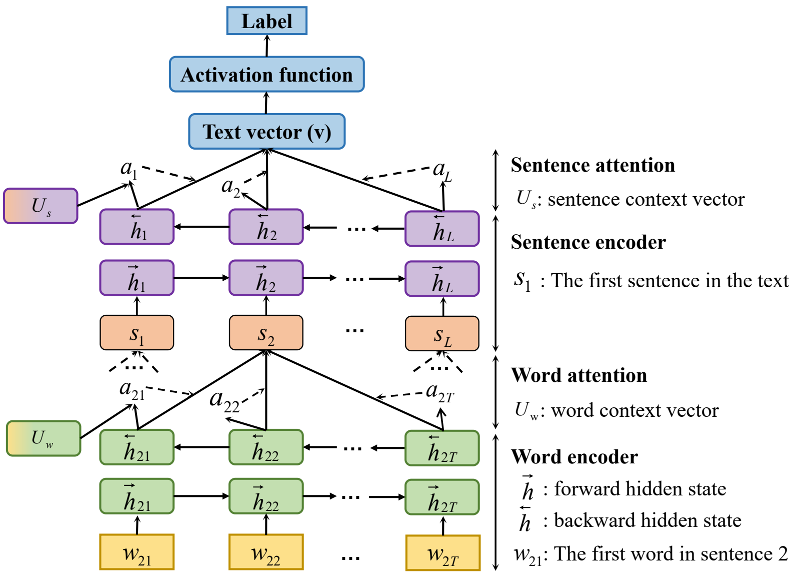

Label
Activation function
Text vector (v)
Sentence attention
U_s: sentence context vector
Sentence encoder
s_1 : The first sentence in the text
Word attention
U_w: word context vector
Word encoder
$\vec{h}$: forward hidden state
$\overleftarrow{h}$: backward hidden state
w_{21}: The first word in sentence 2

Fig. 8. The architecture of hierarchical attention network (HAN) [108].

2.2.5 *Attention-based Methods.* CNN and RNN provide excellent results on tasks related to text classification. However, these models are not intuitive enough for poor interpretability, especially in classification errors, which cannot be explained due to the non-readability of hidden data. The attention-based methods are successfully used in the text classification. Bahdanau et al. [190] first propose an attention mechanism that can be used in machine translation. Motivated by this, Yang et al. [108] introduce the Hierarchical Attention Network (HAN) to gain better visualization by employing the extremely informational components of a text, as shown in Fig. 8. HAN includes two encoders and two levels of attention layers. The attention mechanism lets the model pay different attention to specific inputs. It aggregates essential words into sentence vectors firstly and then aggregates vital sentence vectors into text vectors. It can learn how much contribution of each word and sentence for the classification judgment, which is beneficial for applications and analysis through the two levels of attention.

The attention mechanism can improve the performance with interpretability for text classification, which makes it popular. There are some other works based on attention. LSTMN [112] is proposed to process text step by step from left to right and does superficial reasoning through memory and attention. BI-Attention [110] is designed for cross-lingual text classification to catch bilingual long-distance dependencies. Hu et al. [191] propose an attention mechanism based on category attributes for solving the imbalance of the number of various charges which contain few-shot charges. HAPN [124] is presented for few-shot text classification.

Self-attention [192] captures the weight distribution of words in sentences by constructing K, Q and V matrices among sentences that can capture long-range dependencies on text classification. We give an example for self-attention, as shown in Fig. 9. Each input word vector a_i can be represented as three n-dimensional vectors, including q_i, k_i and v_i. After self-attention, the output vector b_i can be represented as $\sum_j softmax(a_{ij})v_j$ and $a_{ij} = q_i \cdot k_j/\sqrt{n}$. All output vectors can be parallelly computed. Lin et al. [114] used source token self-attention to explore the weight of every token to the entire sentence in the sentence representation task. To capture long-range dependencies, Bi-directional Block Self-Attention Network (Bi-BloSAN) [120] uses an intra-block Self-Attention Network (SAN) to every block split by sequence and an inter-block SAN to the outputs.

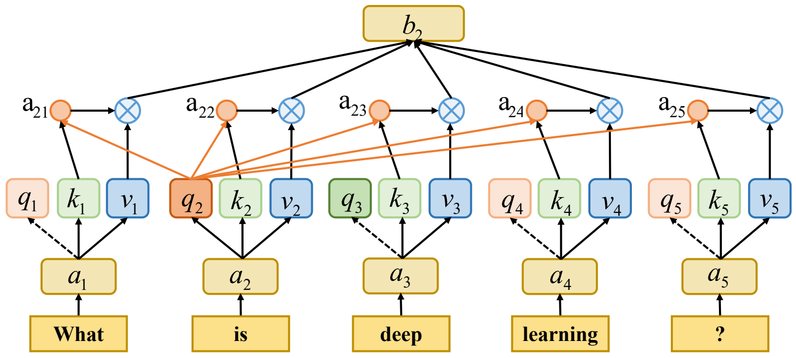


Fig. 9. An example of self-attention for calculating output vector b_2.

Aspect-Based Sentiment Analysis (ABSA) [31, 193] breaks down a text into multiple aspects and allocates each aspect a sentiment polarity. The sentiment polarity can be divided into three types: positive, neutral and negative. Some attention-based models are proposed to identify the fine-grained opinion polarity towards a specific aspect for aspect-based sentiment tasks. ATAE-LSTM [194] can concentrate on different parts of each sentence according to the input through the attention mechanisms. MGAN [195] presents a fine-grained attention mechanism with a coarse-grained attention mechanism to learn the word-level interaction between context and aspect.

To catch the complicated semantic relationship among each question and candidate answers for the QA task, Tan et al. [196] introduce CNN and RNN and generate answer embeddings by using a simple one-way attention mechanism affected through the question context. The attention captures the dependence among the embeddings of questions and answers. Extractive QA can be seen as the text classification task. It inputs a question and multiple candidates answers and classifies every candidate answer to recognize the correct answer. Furthermore, AP-BILSTM [197] with a two-way attention mechanism can learn the weights between the question and each candidate answer to obtain the importance of each candidate answer to the question.

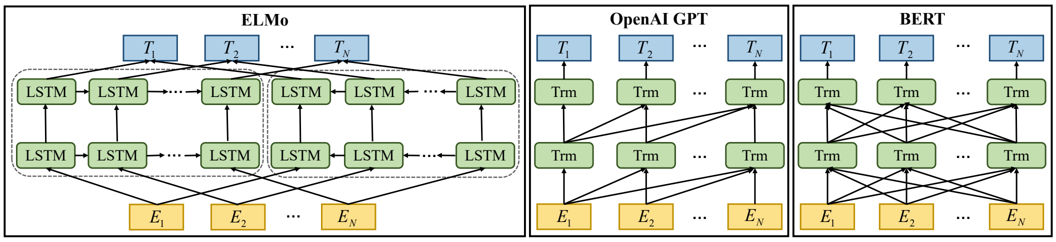


Fig. 10. Differences in pre-trained model architectures [19], including BERT, OpenAI GPT and ELMo. E_i represents embedding of i th input. Trm represents the transformer block. T_i represents predicted tag of i th input.

2.2.6 Pre-trained Methods. Pre-trained language models [198] effectively learn global semantic representation and significantly boost NLP tasks, including text classification. It generally uses unsupervised methods to mine semantic knowledge automatically and then construct pre-training targets so that machines can learn to understand semantics.

As shown in Fig. 10, we give differences in the model architectures among the Embedding from Language Model (ELMo) [118], OpenAI GPT [199], and BERT [19]. ELMo [118] is a deep contextualized word representation model, which is readily integrated into models. It can model complicated characteristics of words and learn different representations for various linguistic contexts. It learns each word embedding according to the context words with the bi-directional LSTM. GPT [199] employs supervised fine-tuning and unsupervised pre-training to learn general representations that transfer with limited adaptation to many NLP tasks. Furthermore, the domain of the target dataset does not need to be similar to the domain of unlabeled datasets. The training procedure of the GPT algorithm usually includes two stages. Firstly, the initial parameters of a neural network model are learned by a modeling objective on the unlabeled dataset. We can then employ the corresponding supervised objective to accommodate these parameters for the target task. To pre-train deep bidirectional representations from the unlabeled text through joint conditioning on both left and right context in every layer, BERT model [19], proposed by Google, significantly improves performance on NLP tasks, including text classification. BERT applies the bi-directional encoder designed to pre-train the bi-directional representation of depth by jointly adjusting the context in all layers. It can utilize contextual information when predicting which words are masked. It is fine-tuned by adding just an additional output layer to construct models for multiple NLP tasks, such as SA, QA, and machine translation. Comparing with these three models, ELMo is a feature-based method using LSTM, and BERT and OpenAI GPT are fine-tuning approaches using Transformer. Furthermore, ELMo and BERT are bidirectional training models and OpenAI GPT is training from left to right. Therefore, BERT gets a better result, which combines the advantages of ELMo and OpenAI GPT.

Transformer-based models can parallelize computation without considering the sequential information suitable for large scale datasets, making it popular for NLP tasks. Thus, some other works are used for text classification tasks and get excellent performance. RoBERTa [140], is an improved version of BERT, adopts the dynamic masking method that generates the masking pattern every time with a sequence to be fed into the model. It uses more data for longer pre-training and estimates the influence of various essential hyperparameters and the size of training data. To be specific: 1) The training time is longer (a total of nearly 200,000 training, nearly 1.6 billion training data have been seen), the batch size (8K) is larger, and the training data is more (30G Chinese training, including 300 million sentences and 10 billion words); 2) It removes the next sentence prediction (NSP) task; 3) It employs more extended training sequence; 4) It dynamically adjusts the masking mechanism and use the full word mask.

XLNet [138] is a generalized autoregressive pre-training approach. Unlike BERT, the denoising autoencoder with the mask is not used in the first stage, but the autoregressive LM is used. It maximizes the expected likelihood across the whole factorization order permutations to learn the bidirectional context. Furthermore, it can overcome the weaknesses of BERT by an autoregressive formulation and integrate ideas from Transformer-XL [200] into pre-training.

BERT model has many parameters. In order to reduce the parameters, ALBERT [146] uses two-parameter simplification schemes. It reduces the fragmentation vector's length and shares parameters with all encoders. It also replaces the next sentence matching task with the next sentence order task and continuously blocks fragmentation. When the ALBERT model is pre-trained on a massive Chinese corpus, the parameters are less and better performance. In general, these methods adopt unsupervised objective functions for pre-training, including the next sentence prediction, masking technology, and permutation. These target functions based on the word prediction demonstrate a strong ability to learn the word dependence and semantic structure [201].

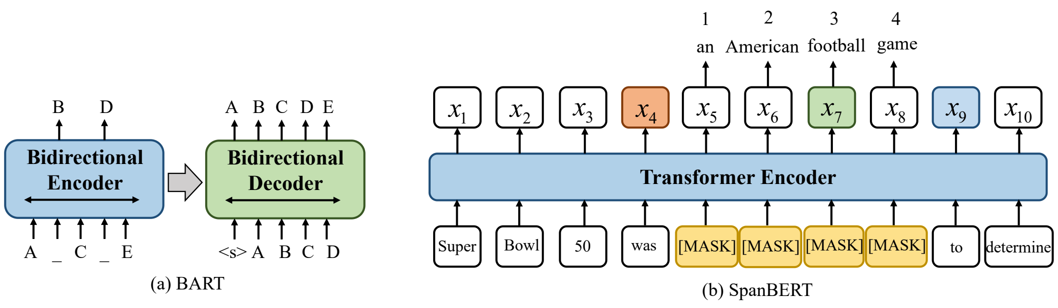

B D
Bidirectional Encoder
A _ C _ E
A B C D E
Bidirectional Decoder
<s> A B C D
(a) BART
1 2 3 4
an American football game
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
Transformer Encoder
Super Bowl 50 was [MASK] [MASK] [MASK] [MASK] to determine
(b) SpanBERT

Fig. 11. The architecture of BART [202] and SpanBERT [203].

BART [202] is a denoising autoencoder based on the Seq2Seq model, as shown in Fig. 11 (a). The pre-training of BART consists of two steps. Firstly, it uses a noise function to destroy the text. Secondly, the Seq2Seq model is used to reconstruct the original text. In various noise methods, by randomly shuffling the order of the original sentence and then using the first new text filling method to obtain optimal performance. The new text filling method is replacing the text fragment with a single mask token. It uses only a specific masked token to indicate that a token is masked.

SpanBERT [203] is specially designed to better represent and predict spans of text, as shown in Fig. 11 (b). It optimizes BERT from three aspects and achieves good results in multiple tasks such as QA. The specific optimization is embodied in three aspects. Firstly, the span mask scheme is proposed to mask a continuous paragraph of text randomly. Secondly, Span Boundary Objective (SBO) is added to predict span by the token next to the span boundary to get the better performance to finetune stage. Thirdly, the NSP pre-training task is removed.

ERNIE [204] is based on the method of knowledge enhancement. It learns the semantic relations in the real world by modeling the prior semantic knowledge such as entity concepts in massive datasets. Specifically, ERNIE enables the model to learn the semantic representation of complete concepts by masking semantic units such as words and entities. It mainly consists of a Transformer encoder and task embedding. In the Transformer encoder, the context information of each token is captured by the self-attention mechanism, and the context representation is generated for embedding. Task embedding is used for tasks with different characteristics.

2.2.7 GNN-based Methods. The DNN models like CNN get great performance on regular structure, not for arbitrarily structured graphs. Some researchers study how to expand on arbitrarily structured graphs [205] [206] [207]. With the increasing attention of Graph Neural Networks (GNNs), GNN-based models obtain excellent performance by encoding syntactic structure of sentences on semantic role labeling task [208], relation classification task [209] and machine translation task [210]. It turns text classification into a graph node classification task. We show a GCN model for text classification with four input texts, as shown in Fig. 12. Firstly, the four input texts $T = [T_1, T_2, T_3, T_4]$ and the words $X = [x_1, x_2, x_3, x_4, x_5, x_6]$ in the text, defined as nodes, are constructed into the graph structures. The graph nodes are connected by bold black edges, which indicates document-word edges and word-word edges. The weight of each word-word edge usually means their co-occurrence frequency in the corpus. Then, the words and texts are represented through the hidden layer. Finally, the label of all input texts can be predicted by the graph.

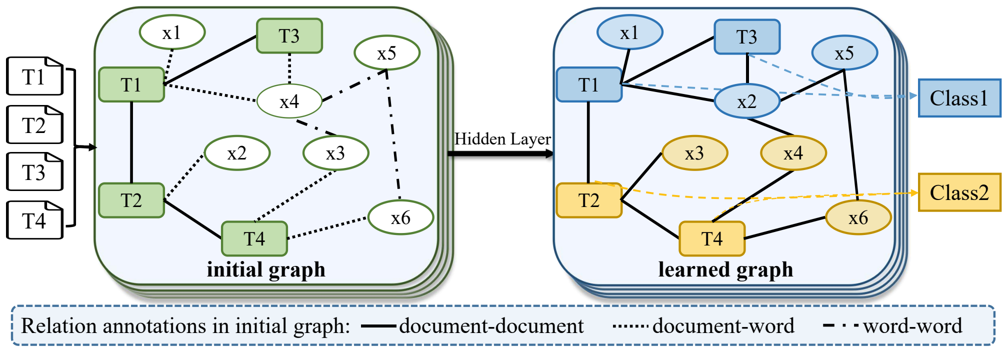


Fig. 12. The GNN-based model. The initial graph differently depending on how the graph is designed. We give an example to establish edges between documents and documents, documents and sentences, and words to words.

The GNN-based models can learn the syntactic structure of sentences, making some researchers study using GNN for text classification. DGCNN [153] is a graph-CNN converting text to graph-of-words, having the advantage of learning different levels of semantics with CNN models. Yao et al. [155] propose the Text Graph Convolutional Network (TextGCN), which builds a heterogeneous word text graph for a whole dataset and captures global word co-occurrence information. To enable GNN-based models to underpin online testing, Huang et al. [159] build graphs for each text with global parameter sharing, not a corpus-level graph structure, to help preserve global information and reduce the burden. TextING [162] builds individual graphs for each document and learns text-level word interactions by GNN to effectively produce embeddings for obscure words in the new text.

Graph ATtention network (GAT) [211] employs masked self-attention layers by attending over its neighbors. Thus, some GAT-based models are proposed to compute the hidden representations of each node. The Heterogeneous Graph ATtention networks (HGAT) [212] with a dual-level attention mechanism learns the importance of different neighboring nodes and node types in the current node. The model propagates information on the graph and captures the relations to address the semantic sparsity for semi-supervised short text classification. MAGNET [166] is proposed to capture the correlation among the labels based on GATs, which learns the crucial dependencies between the labels and generates classifiers by a feature matrix and a correlation matrix.

Event Prediction (EP) can be divided into generated event prediction and selective event prediction (also known as script event prediction). EP, referring to scripted event prediction in this review, infers the subsequent event according to the existing event context. Unlike other text classification tasks, texts in EP are composed of a series of sequential subevents. Extracting features of the relationship among such subevents is of critical importance. SGNN [213] is proposed to model event interactions and learn better event representations by constructing an event graph to utilize the event network information better. The model makes full use of dense event connections for the EP task.

2.2.8 Others. In addition to all the above models, there are some other individual models. Here we introduce some exciting models.

Siamese Neural Network. The siamese neural network [214] is also called a twin neural network (Twin NN). It utilizes equal weights while working in tandem using two distinct input vectors to calculate comparable output vectors. Mueller et al. [215] present a siamese adaptation of the LSTM network comprised of couples of variable-length sequences. The model is employed to estimate the semantic similarity among texts, exceeding carefully handcrafted features and proposed neural network models of higher complexity. The model further represents text employing neural networks whose inputs are word vectors learned separately from a vast dataset. To settle unbalanced data classification in the medical domain, Jayadeva et al. [216] use a Twin NN model to learn from enormous unbalanced corpora. The objective functions achieve the Twin SVM approach with non-parallel decision boundaries for the corresponding classes, and decrease the Twin NN complexity, optimizing the feature map to better discriminate among classes.

Virtual Adversarial Training (VAT). Deep learning methods require many extra hyperparameters, which increase the computational complexity. VAT [217], regularization based on local distributional smoothness can be used in semi-supervised tasks, requires only some hyperparameters, and can be interpreted directly as robust optimization. Miyato et al. [85] use VAT to effectively improve the robustness and generalization ability of the model and word embedding performance.

Reinforcement Learning (RL). RL learns the best action in a given environment through maximizing cumulative rewards. Zhang et al. [218] offer an RL approach to establish structured sentence representations via learning the structures related to tasks. The model has Information Distilled LSTM (ID-LSTM) and Hierarchical Structured LSTM (HS-LSTM) representation models. The ID-LSTM learns the sentence representation by choosing essential words relevant to tasks, and the HS-LSTM is a two-level LSTM for modeling sentence representation.

Memory Networks. Memory networks [219] learn to combine the inference components and the long-term memory component. Li et al. [220] use two LSTMs with extended memories and neural memory operations for jointly handling the extraction tasks of aspects and opinions via memory interactions. Topic Memory Networks (TMN) [169] is an end-to-end model that encodes latent topic representations indicative of class labels.

QA Style for Sentiment Classification Task. It is an interesting attempt to treat the sentiment classification task as a QA task. Shen et al. [221] create a high-quality annotated corpus. A three-stage hierarchical matching network was proposed to consider the matching information between questions and answers.

External Commonsense Knowledge. Due to the insufficient information of the event itself to distinguish the event for the EP task, Ding et al. [222] consider that the event extracted from the original text lacked common knowledge, such as the intention and emotion of the event participants. The model improves the effect of stock prediction, EP, and so on.

Quantum Language Model. In the quantum language model, the words and dependencies among words are represented through fundamental quantum events. Zhang et al. [223] design a quantum-inspired sentiment representation method to learn both the semantic and the sentiment information of subjective text. By inputting density matrices to the embedding layer, the performance of the model improves.

Summary. RNN computes sequentially and cannot be calculated in parallel. The shortcoming of RNN makes it more challenging to become mainstream in the current trend that models tend to have deeper and more parameters. CNN extracts features from text vectors through the convolution kernel. The number of features captured by the convolution kernel is related to its size. CNN is deep enough that, in theory, it can capture features at long distances. Due to insufficient optimization methods for parameters of the deep network and the loss of location information due to the pooling layer, the deeper layer does not bring significant improvement. Compared with RNN, CNN has parallel computing capability and can effectively retain location information for the improved version of CNN. Still, it has weak feature capture capability for long-distance. GNN builds a graph for text. When a valid graph structure is designed, the learned representation can better capture the structural information. Transformer treats the input text as a fully connected graph, with attention score weights on the edges. It is capable of parallel computing and is highly efficient in extracting features between different words by self-attention, solving short-term memory problems. However, the attention mechanism in Transformer is computation-heavy, especially when dealing with long sequences. Some improved models [224] [146] [225] for computing complexity in Transformer have recently been proposed. Overall, Transformer is a better choice for text classification. Deep Learning consists of multiple hidden layers in a neural network with a higher level of complexity and can be trained on unstructured data. Deep learning can learn language features and master higher level and more abstract language features based on words and vectors. Deep learning architecture can learn feature representations directly from the input without too many manual interventions and prior knowledge. However, deep learning technology is a data-driven method that requires enormous data to achieve high performance. Although self-attention based models can bring some interpretability among words for DNNs, it is not enough comparing with traditional models to explain why and how it works well.

3 DATASETS AND EVALUATION METRICS

3.1 Datasets

The availability of labeled datasets for text classification has become the main driving force behind the fast advancement of this research field. In this section, we summarize the characteristics of these datasets in terms of domains and give an overview in Table 2, including the number of categories, average sentence length, the size of each dataset, related papers, data sources to access and applications.

3.1.1 Sentiment Analysis (SA). SA is the process of analyzing and reasoning the subjective text within emotional color. It is crucial to get information on whether it supports a particular point of view from the text that is distinct from the traditional text classification that analyzes the objective content of the text. SA can be binary or multi-class. Binary SA is to divide the text into two categories, including positive and negative. Multi-class SA classifies text to multi-level or fine-grained labels. The SA datasets include Movie Review (MR) [257] [226], Stanford Sentiment Treebank (SST) [227], Multi-Perspective Question Answering (MPQA) [258] [229], IMDB [230], Yelp [231], Amazon Reviews (AM) [93], NLP&CC 2013 [111], Subj [250], CR [251], SS-Twitter [259], SS-Youtube [259], SE1604 [260] and so on. Here we detail several datasets.

MR. The MR is a movie review dataset, each of which corresponds to a sentence. The corpus has 5,331 positive data and 5,331 negative data. 10-fold cross-validation by random splitting is commonly used to test MR.

SST. The SST is an extension of MR. It has two categories. SST-1 with fine-grained labels with five classes. It has 8,544 training texts and 2,210 test texts, respectively. Furthermore, SST-2 has 9,613 texts with binary labels being partitioned into 6,920 training texts, 872 development texts, and 1,821 testing texts.

MPQA. The MPQA is an opinion dataset. It has two class labels and also an MPQA dataset of opinion polarity detection sub-tasks. MPQA includes 10,606 sentences extracted from news articles from various news sources. It should be noted that it contains 3,311 positive texts and 7,293 negative texts without labels of each text.

IMDB reviews. The IMDB review is developed for binary sentiment classification of film reviews with the same amount in each class. It can be separated into training and test groups on average, by 25,000 comments per group.

Table 2. Summary statistics for the datasets. C: Number of target classes. L: Average sentence length. N: Dataset size.

Datasets	#C	#L	#N	Language	Related Papers	Sources	Applications
MR	2	20	10,662	English	[17] [5] [101] [155]	[226]	SA
SST-1	5	18	11,855	English	[64] [17] [1] [2][112]	[227]	SA
SST-2	2	19	9,613	English	[64] [17] [6] [79] [19]	[228]	SA
MPQA	2	3	10,606	English	[60] [17] [120]	[229]	SA
IMDB	2	294	50,000	English	[69][108] [6] [79] [85]	[230]	SA
Yelp.P	2	153	598,000	English	[93] [98]	[231]	SA
Yelp.F	5	155	700,000	English	[93] [108] [98]	[231]	SA
Amz.P	2	91	4,000,000	English	[122] [93]	[232]	SA
Amz.F	5	93	3,650,000	English	[93] [108] [122]	[232]	SA
Twitter	3	19	11,209	English	[5][100]	[233]	SA
NLP&CC 2013	2	-	115,606	Multi-language	[110]	[111]	SA
20NG	20	221	18,846	English	[72] [75] [106] [155] [157]	[34]	NC
AG News	4	45/7	127,600	English	[98] [100] [101] [138]	[234]	NC
R8	8	66	7,674	English	[155] [157] [159]	[235]	NC
R52	52	70	9,100	English	[155] [157] [159]	[235]	NC
Sogou	6	578	510,000	Chinese	[93]	[236]	NC
Newsgroup	20	-	18,846	English	[237]	[237]	NC
DBPedia	14	55	630,000	English	[93] [98] [85] [136]	[238]	TL
Ohsumed	23	136	7,400	English	[155] [157] [159]	[239]	TL
YahooA	10	112	1,460,000	English	[93] [108]	[93]	TL
EUR-Lex	3,956	1,239	19,314	English	[96] [134] [161] [134]	[240]	TL
Amazon670K	670	244	643,474	English	[103] [122]	[241]	TL
Google news	152	6	11,109	English	[242] [3] [243]	[242]	TL
TweetSet 2011-2012	89	-	2,472	English	[242] [243]	[242]	TL
TweetSet 2011-2015	269	8	30,322	English	[4] [3]	[4]	TL
Bing	4	20	34,871	English	[100]	[244]	TL
Fudan	20	2981	18,655	Chinese	[72]	[245]	TL
SQuAD	-	5,000	5,570	English	[118] [118] [140] [146]	[246]	QA
TREC-QA	-	1,162	68	English	[197]	[247]	QA
TREC	6	10	5,952	English	[17] [5] [6] [100]	[248]	QA
WikiQA	-	873	243	English	[249] [197]	[249]	QA
Subj	2	23	10,000	English	[17] [79] [101]	[250]	QA
CR	2	19	3,775	English	[17] [101]	[251]	QA
Reuters	90	168	10,788	English	[101] [166]	[252]	ML
Reuters10	10	168	9,979	English	[253]	[254]	ML
RCV1	103	240	807,595	English	[75] [103] [153] [134]	[255]	ML
RCV1-V2	103	124	804,414	English	[116] [166]	[256]	ML
AAPD	54	163	55,840	English	[116] [166]	[117]	ML

Yelp reviews. The Yelp review is summarized from the Yelp Dataset Challenges in 2013, 2014, and 2015. This dataset has two categories. Yelp-2 of these were used for negative and positive emotion classification tasks, including 560,000 training texts and 38,000 test texts. Yelp-5 is used to detect fine-grained affective labels with 650,000 training and 50,000 test texts in all classes.

AM. The AM is a popular corpus formed by collecting Amazon website product reviews [232]. This dataset has two categories. The Amazon-2 with two classes includes 3,600,000 training sets and 400,000 testing sets. Amazon-5, with five classes, includes 3,000,000 and 650,000 comments for training and testing.

3.1.2 News Classification (NC). News content is one of the most crucial information sources which has a critical influence on people. The NC system facilitates users to get vital knowledge in real-time. News classification applications mainly encompass: recognizing news topics and recommending related news according to user interest. The news classification datasets include 20 Newsgroups (20NG) [34], AG News (AG) [93] [234], R8 [235], R52 [235], Sogou News (Sogou) [136] and so on. Here we detail several datasets.

20NG. The 20NG is a newsgroup text dataset. It has 20 categories with the same number of each category and includes 18,846 texts.

AG. The AG News is a search engine for news from academia, choosing the four largest classes. It uses the title and description fields of each news. AG contains 120,000 texts for training and 7,600 texts for testing.

R8 and R52. R8 and R52 are two subsets which are the subset of Reuters [252]. R8 has 8 categories, divided into 2,189 test files and 5,485 training courses. R52 has 52 categories, split into 6,532 training files and 2,568 test files.

Sogou. The Sogou combines two datasets, including SogouCA and SogouCS news sets. The label of each text is the domain names in the URL.

3.1.3 Topic Labeling (TL). The topic analysis attempts to get the meaning of the text by defining the sophisticated text theme. The topic labeling is one of the essential components of the topic analysis technique, intending to assign one or more subjects for each document to simplify the topic analysis. The topic labeling datasets include DBPedia [238], Ohsumed [239], Yahoo answers (YahooA) [93], EUR-Lex [240], Amazon670K [241], Bing [244], Fudan [245], and PubMed [261]. Here we detail several datasets.

DBpedia. The DBpedia is a large-scale multi-lingual knowledge base generated using Wikipedia's most ordinarily used infoboxes. It publishes DBpedia each month, adding or deleting classes and properties in every version. DBpedia's most prevalent version has 14 classes and is divided into 560,000 training data and 70,000 test data.

Ohsumed. The Ohsumed belongs to the MEDLINE database. It includes 7,400 texts and has 23 cardiovascular disease categories. All texts are medical abstracts and are labeled into one or more classes.

YahooA. The YahooA is a topic labeling task with 10 classes. It includes 140,000 training data and 5,000 test data. All text contains three elements, being question titles, question contexts, and best answers, respectively.

3.1.4 Question Answering (QA). The QA task can be divided into two types: the extractive QA and the generative QA. The extractive QA gives multiple candidate answers for each question to choose which one is the right answer. Thus, the text classification models can be used for the extractive QA task. The QA discussed in this paper is all extractive QA. The QA system can apply the text classification model to recognize the correct answer and set others as candidates. The question answering datasets include Stanford Question Answering Dataset (SQuAD) [246], TREC-QA [248], WikiQA [249], Subj [250], CR [251], MS MARCO [262], and Quora [263]. Here we detail several datasets.

SQuAD. The SQuAD is a set of question and answer pairs obtained from Wikipedia articles. The SQuAD has two categories. SQuAD1.1 contains 536 pairs of 107,785 Q&A items. SQuAD2.0 combines 100,000 questions in SQuAD1.1 with more than 50,000 unanswerable questions that crowd workers face in a form similar to answerable questions [264].

TREC-QA. The TREC-QA includes 5,452 training texts and 500 testing texts. It has two versions. TREC-6 contains 6 categories, and TREC-50 has 50 categories.

WikiQA. The WikiQA dataset includes questions with no correct answer, which needs to evaluate the answer.

MS MARCO. The MS MARCO contains questions and answers. The questions and part of the answers are sampled from actual web texts by the Bing search engine. Others are generative. It is used for developing generative QA systems released by Microsoft.

3.1.5 *Natural Language Inference (NLI).*

NLI is used to predict whether the meaning of one text can be deduced from another. Paraphrasing is a generalized form of NLI. It uses the task of measuring the semantic similarity of sentence pairs to decide whether one sentence is the interpretation of another. The NLI datasets include Stanford Natural Language Inference (SNLI) [181], Multi-Genre Natural Language Inference (MNLI) [265], Sentences Involving Compositional Knowledge (SICK) [266], Microsoft Research Paraphrase (MSRP) [267], Semantic Textual Similarity (STS) [268], Recognising Textual Entailment (RTE) [269], SciTail [270], etc. Here we detail several of the primary datasets.

SNLI. The SNLI is generally applied to NLI tasks. It contains 570,152 human-annotated sentence pairs, including training, development, and test sets, which are annotated with three categories: neutral, entailment, and contradiction.

MNLI. The MNLI is an expansion of SNLI, embracing a broader scope of written and spoken text genres. It includes 433,000 sentence pairs annotated by textual entailment labels.

SICK. The SICK contains almost 10,000 English sentence pairs. It consists of neutral, entailment and contradictory labels.

MSRP. The MSRP consists of sentence pairs, usually for the text-similarity task. Each pair is annotated by a binary label to discriminate whether they are paraphrases. It respectively includes 1,725 training and 4,076 test sets.

3.1.6 *Multi-Label (ML) datasets.*

In multi-label classification, an instance has multiple labels, and each label can only take one of the multiple classes. There are many datasets based on multi-label text classification. It includes Reuters [252], Reuters Corpus Volume I (RCV1) [255], RCV1-2K [255], Arxiv Academic Paper Dataset (AAPD) [117], Patent, Web of Science (WOS-11967) [271], AmazonCat-13K [272], BlurbGenreCollection (BGC) [273], etc. Here we detail several datasets.

Reuters. The Reuters is a popularly used dataset for text classification from Reuters financial news services. It has 90 training classes, 7,769 training texts, and 3,019 testing texts, containing multiple labels and single labels. There are also some Reuters sub-sets of data, such as R8, BR52, RCV1, and RCV1-v2.

RCV1 and RCV1-2K. The RCV1 is collected from Reuters News articles from 1996-1997, which is human-labeled with 103 categories. It consists of 23,149 training and 784,446 testing texts, respectively. The RCV1-2K dataset has the same features as the RCV1. However, the label set of RCV1-2K has been expanded with some new labels. It contains 2456 labels.

AAPD. The AAPD is a large dataset in the computer science field for the multi-label text classification from website [1]. It has 55,840 papers, including the abstract and the corresponding subjects with 54 labels in total. The aim is to predict the corresponding subjects of each paper according to the abstract.

Patent Dataset. The Patent Dataset is obtained from USPTO [2], which is a patent system grating U.S. patents containing textual details such title and abstract. It contains 100,000 US patents awarded in the real-world with multiple hierarchical categories.

WOS-11967. The WOS-11967 is crawled from the Web of Science, consisting of abstracts of published papers with two labels for each example. It is shallower, but significantly broader, with fewer classes in total.

[1]https://arxiv.org/
[2]https://www.uspto.gov/

Table 3. The notations used in evaluation metrics.

Notations	Descriptions
TP	true positive
FP	false positive
TN	true negative
FN	false negative
TP_t	true positive of the t th label on a text
FP_t	false positive of the t th label on a text
TN_t	true negative of the t th label on a text
FN_t	false negative of the t th label on a text
$\mathcal{S}$	label set of all samples
Q	the number of predicted labels on each text

3.1.7 Others. There are some datasets for other applications, such as SemEval-2010 Task 8 [274], ACE 2003-2004 [275], TACRED [276], and NYT-10 [277], FewRel [278], Dialog State Tracking Challenge 4 (DSTC 4) [279], ICSI Meeting Recorder Dialog Act (MRDA) [280], and Switchboard Dialog Act (SwDA) [281], and so on.

3.2 Evaluation Metrics

In terms of evaluating text classification models, accuracy and F1 score are the most used to assess the text classification methods. Later, with the increasing difficulty of classification tasks or the existence of some particular tasks, the evaluation metrics are improved. For example, evaluation metrics such as $P@K$ and $Micro\text{–}F1$ are used to evaluate multi-label text classification performance, and MRR is usually used to estimate the performance of QA tasks. In Table 3, we give the notations used in evaluation metrics.

3.2.1 Single-label metrics. Single-label text classification divides the text into one of the most likely categories applied in NLP tasks such as QA, SA, and dialogue systems [7]. For single-label text classification, one text belongs to just one catalog, making it possible not to consider the relations among labels. Here, we introduce some evaluation metrics used for single-label text classification tasks.

Accuracy and ErrorRate. The Accuracy and ErrorRate are the fundamental metrics for a text classification model. The *Accuracy* and *ErrorRate* are respectively defined as

$$Accuracy = \frac{(TP + TN)}{N}, \tag{1}$$

$$ErrorRate = 1 - Accuracy = \frac{(FP + FN)}{N}. \tag{2}$$

Precision, Recall and F1. These are vital metrics utilized for unbalanced test sets, regardless of the standard type and error rate. For example, most of the test samples have a class label. $F1$ is the harmonic average of *Precision* and *Recall*. *Precision*, *Recall*, and $F1$ as defined

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \tag{3}$$

$$F1 = \frac{2Precision \times Recall}{Precision + Recall}. \tag{4}$$

The desired results will be obtained when the accuracy, $F1$ and $Recall$ value reach 1. On the contrary, when the values become 0, the worst result is obtained. For the multi-class classification problem, the precision and recall value of each class can be calculated separately, and then the performance of the individual and whole can be analyzed.

Exact Match (EM). The EM [29] is a metric for QA tasks, measuring the prediction that matches all the ground-truth answers precisely. It is the primary metric utilized on the SQuAD dataset.

Mean Reciprocal Rank (MRR). The MRR [282] is usually applied for assessing the performance of ranking algorithms on QA and Information Retrieval (IR) tasks. MRR is defined as

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{rank(i)}, \tag{5}$$

where $rank(i)$ is the ranking of the ground-truth answer at answer i-th.

Hamming-Loss (HL). The HL [57] assesses the score of misclassified instance-label pairs where a related label is omitted or an unrelated is predicted.

Among these single-label evaluation metrics, the Accuracy is the earliest metric that calculates the proportion of the sample size that is predicted correctly and is not considered whether the predicted sample is a positive or a negative sample. $Precision$ calculates how many of the positive samples are actually positive, and the $Recall$ calculates how many of the positive examples in the sample are predicted correctly. Furthermore, $F1$ is the harmonic average of them, which is the most commonly used evaluation metrics.

3.2.2 Multi-label metrics. Compared with single-label text classification, multi-label text classification divides the text into multiple category labels, and the number of category labels is variable. These metrics are designed for single label text classification, which are not suitable for multi-label tasks. Thus, there are some metrics designed for multi-label text classification.

Micro–F1. The $Micro–F1$ [283] is a measure that considers the overall accuracy and recall of all labels. The $Micro–F1$ is defined as:

$$Micro–F1 = \frac{2P_t \times R_t}{P + R}, \tag{6}$$

where:

$$P = \frac{\sum_{t \in \mathcal{S}} TP_t}{\sum_{t \in S} TP_t + FP_t}, \quad R = \frac{\sum_{t \in S} TP_t}{\sum_{t \in \mathcal{S}} TP_t + FN_t}. \tag{7}$$

Macro–F1. The $Macro–F1$ [283] calculates the average $F1$ of all labels. Unlike $Micro–F1$, which sets even weight to every example, $Macro–F1$ sets the same weight to all labels in the average process. Formally, $Macro–F1$ is defined as:

$$Macro–F1 = \frac{1}{\mathcal{S}} \sum_{t \in \mathcal{S}} \frac{2P_t \times R_t}{P_t + R_t}, \tag{8}$$

where:

$$P_t = \frac{TP_t}{TP_t + FP_t}, \quad R_t = \frac{TP_t}{TP_t + FN_t}. \tag{9}$$

In addition to the above evaluation metrics, there are some rank-based evaluation metrics for extreme multi-label classification tasks, including $P@K$ and $NDCG@K$.

Precision at Top K (P@K). The $P@K$ [96] is the precision at the top k. For $P@K$, each text has a set of $\mathcal{L}$ ground truth labels $L_t = \{l_0, l_1, l_2 \ldots, l_{\mathcal{L}-1}\}$, in order of decreasing probability $P_t =$

$[p_0, p_1, p_2 \ldots, p_{Q-1}]$. The precision at k is

$$P@K = \frac{1}{k} \sum_{j=0}^{\min(\mathcal{L},k)-1} rel_{L_i}\left(P_t(j)\right), \tag{10}$$

$$\mathrm{rel}_L(p) = \begin{cases} 1 & \text{if } p \in L \\ 0 & \text{otherwise} \end{cases}, \tag{11}$$

where $\mathcal{L}$ is the number of ground truth labels or possible answers on each text and k is the number of selected labels on extreme multi-label text classification.

Normalized Discounted Cummulated Gains (NDCG@K). The $NDCG@K$ [96] is

$$NDCG@K = \frac{1}{IDCG\left(L_i, k\right)} \sum_{j=0}^{n-1} \frac{rel_{L_i}\left(P_t(j)\right)}{\ln(j+1)}, \tag{12}$$

where $IDCG$ is ideal discounted cumulative gain and the particular rank position n is

$$n = \min\left(\max\left(|P_i|, |L_i|\right), k\right). \tag{13}$$

Among these multi-label evaluation metrics, $Micro{-}F1$ considers the number of categories, which makes it suitable for the unbalanced data distribution. $Macro{-}F1$ does not take into account the amount of data that treats each class equally. Thus, it is easily affected by the classes with high Recall and Precision. When the number of categories is large or extremely large, either P@K or NDCG@K is used.

4 QUANTITATIVE RESULTS

There are many differences between sentiment analysis, news classification, topic labeling and natural language inference tasks, which can not be simplified modeled as a text classification task. In this section, we tabulate the performance of the main models given in their articles on classic datasets evaluated by classification accuracy, as shown in Table 4, including MR, SST-2, IMDB, Yelp.P, Yelp.F, Amazon.F, 20NG, AG, DBpedia, and SNLI.

We give the performance of NB and SVM algorithms from RNTN [64] due to the less traditional text classification model has been an experiment on datasets in Table 4. The accuracy of NB and SVM are 81.8% and 79.4% on SST-2, respectively. We can see that, in the SST-2 data set with only two categories, the accuracy of NB is better than that of SVM. It may be because NB has relatively stable classification efficiency on new data sets. The performance is also stable on small data sets. Compared with the deep learning model, the performance of NB is lower. NB has the advantage of lower computational complexity than deep models. However, it requires manual classification features, making it difficult to migrate the model directly to other data sets.

For deep learning models, pre-trained models get better results on most datasets. It means that if you need to implement a text classification task, you can preferentially try pre-trained models, such as BERT, RoBERTa, and XLNET, etc., except MR and 20NG, which have not been experimented on BERT based models. Pre-trained models are essential to NLP. It uses a deep model to learn a better feature of the text. It also demonstrates that the accuracy of NLP tasks can be significantly improved by a profound model that can be pre-trained from unlabeled datasets. For the MR dataset, the accuracy of RNN-Capsule [87] is 83.8%, obtaining the best result. It suggests that RNN-Capsule builds a capsule in each category for sentiment analysis. It can output words including sentiment trends indicating attributes of capsules with no applying linguistic knowledge. For 20NG dataset, BLSTM-2DCNN [77] gets 96.5% score with the best accuracy score. It may demonstrate the

Table 4. Accuracy of text classification models on primary datasets evaluated by classification accuracy (in terms of publication year). Bold is the most accurate.

Model	**Sentiment**						**News**		**Topic**	**NLI**
	MR	SST-2	IMDB	Yelp.P	Yelp.F	Amz.F	20NG	AG	DBpedia	SNLI
NB [8]	-	81.80	-	-	-	-	-	-	-	-
SVM [42]	-	79.40	-	-	-	-	-	-	-	-
Tree-CRF [284]	77.30	-	-	-	-	-	-	-	-	-
RAE [60]	77.70	82.40	-	-	-	-	-	-	-	-
MV-RNN [62]	79.00	82.90	-	-	-	-	-	-	-	-
RNTN [64]	75.90	85.40	-	-	-	-	-	-	-	-
DCNN [5]		86.80	89.40	-	-	-	-	-	-	-
Paragraph-Vec [67]		87.80	92.58	-	-	-	-	-	-	-
TextCNN[17]	81.50	88.10	-	-	-	-	-	-	-	-
TextRCNN [72]	-	-	-	-	-	-	96.49	-	-	-
DAN [69]	-	86.30	89.40	-	-	-	-	-	-	-
Tree-LSTM [1]		88.00	-	-	-	-	-	-	-	-
CharCNN [93]	-	-	-	95.12	62.05	-	-	90.49	98.45	-
HAN [108]	-	-	49.40	-	-	63.60	-	-	-	-
SeqTextRCNN [7]	-	-	-	-	-	-	-	-	-	-
oh-2LSTMp [75]	-	-	94.10	97.10	67.61	-	86.68	93.43	99.16	-
LSTMN [112]	-	87.30	-	-	-	-	-	-	-	-
Multi-Task [79]	-	87.90	91.30	-	-	-	-	-	-	-
BLSTM-2DCNN [77]	82.30	89.50	-	-	-	-	**96.50**	-	-	-
TopicRNN [83]	-	-	93.72	-	-	-	-	-	-	-
DPCNN [98]	-	-	-	97.36	69.42	65.19	-	93.13	99.12	-
KPCNN [100]	83.25	-	-	-	-	-	-	88.36	-	-
RNN-Capsule [87]	**83.80**		-	-	-	-	-	-	-	-
ULMFiT [285]	-	-	95.40	97.84	71.02	-	-	94.99	99.20	-
LEAM[286]	76.95	-	-	95.31	64.09	-	81.91	92.45	99.02	-
TextCapsule [101]	82.30	86.80	-	-	-	-	-	92.60	-	-
TextGCN [155]	76.74	-	-	-	-	-	86.34	67.61	-	-
BERT-base [19]	-	93.50	95.63	98.08	70.58	61.60	-	-	-	91.00
BERT-large [19]	-	94.90	95.79	98.19	71.38	62.20	-	-	-	91.70
MT-DNN[287]	-	95.60	83.20	-	-	-	-	-	-	91.50
XLNet-Large [138]	-	96.80	**96.21**	**98.45**	**72.20**	**67.74**	-	-	-	-
XLNet [138]	-	**97.00**	-	-	-	-	-	**95.51**	**99.38**	-
RoBERTa [140]	-	96.40	-	-	-	-	-	-	-	**92.60**

effectiveness of applying the 2D max-pooling operation to obtain a fixed-length representation of the text and utilize 2D convolution to sample more meaningful matrix information.

5 FUTURE RESEARCH CHALLENGES

Text classification – as efficient information retrieval and mining technology – plays a vital role in managing text data. It uses NLP, data mining, machine learning, and other techniques to automatically classify and discover different text types. Text classification takes multiple types of text as input, and the text is represented as a vector by the pre-training model. Then the vector is fed into the DNN for training until the termination condition is reached, and finally, the performance of the training model is verified by the downstream task. Existing models have already shown their usefulness in text classification, but there are still many possible improvements to explore.

Although some new text classification models repeatedly brush up the accuracy index of most classification tasks, it cannot indicate whether the model "understands" the text from the semantic level like human beings. Moreover, with the emergence of the noise sample, the small sample noise may cause the decision confidence to change substantially or even lead to decision reversal. Therefore, the semantic representation ability and robustness of the model need to be proved in practice. Besides, the pre-trained semantic representation model represented by word vectors can often improve the performance of downstream NLP tasks. The existing research on the transfer strategy of context-free word vectors is still relatively preliminary. Thus, we conclude from data, models, and performance perspective, text classification mainly faces the following challenges.

5.1 Challenges from Data Perspective

For a text classification task, data is essential to model performance, whether it is traditional or deep learning method. The text data mainly studied includes multi-chapter, short text, cross-language, multi-label, less sample text. For the characteristics of these data, the existing technical challenges are as follows:

Zero-shot/Few-shot learning. Zero-shot or few-shot learning for text classification aim to classify text having no or few same labeled class data. However, the current models are too dependent on numerous labeled data. The performance of these models is significantly affected by zero-shot or few-shot learning. Thus, some works focus on tackling these problems. The main idea is to infer the features through learning kinds of semantic knowledge, such as learning relationship among classes [288] and incorporating class descriptions [170]. Furthermore, latent features generation [289] meta-Learning [290] [291] [106] and dynamic memory mechanism [292] are also efficient methods. Nevertheless, with the limitation of little unseen class data and different data distribution between seen class and unseen class, there is still a long way to go to reach the learning ability comparable to that of humans.

The external knowledge. As we all know, the more beneficial information is input into a DNN, its better performance. For example, a question answering system incorporating a common-sense knowledge base can answer questions about the real world and help solve problems with incomplete information. Therefore, adding external knowledge (knowledge base or knowledge graph) [293] [294] is an efficient way to promote the model's performance. The existing knowledge includes conceptual information [100] [127] [204], commonsense knowledge [222], knowledge base information [295] [296], general knowledge graph [170] and so on, which enhances the semantic representation of texts. Nevertheless, with the limitation of input scale, how and what to add for different tasks is still a challenge.

Special domain with many terminologies. Most of the existing models are supervised models, which over-rely on numerous labeled data. When the sample size is too small, or zero samples occur, the performance of the model will be significantly affected. New data set annotation takes a lot of time. Therefore, unsupervised learning and semi-supervised learning have great potential for text classification. Furthermore, texts in a particular field [297] [298], such as financial and medical texts, contain many specific words or domain experts intelligible slang, abbreviations, etc., which make the existing pre-trained word vectors challenging to work on.

The multi-label text classification task. Multi-label text classification requires full consideration of the semantic relationship among labels, and the embedding and encoding of the model is a process of lossy compression [299] [300]. Therefore, how to reduce the loss of hierarchical semantics and retain rich and complex document semantic information during training is still a problem to be solved.

5.2 Challenges from Model Perspective

Most existing structures of traditional and deep learning models are tried for text classification, including integration methods. BERT learns a language representation that can be used to fine-tune for many NLP tasks. The primary method is to increase data, improve computation power, and design training procedures for getting better results [301] [302] [303]. How the tradeoff between data and compute resources and prediction performance is worth studying.

Text representation. The text representation method based on the vector space model is simple and effective in the text preprocessing stage. However, it will lose the semantic information of the text, so the application performance based on this method is limited. The proposed semantically based text representation method is too time-consuming. Therefore, the efficient semantically based text representation method still needs further research. In the text representation of text classification based on deep learning, word embedding is the main concept, while the representation unit is described differently in different languages. Then, a word is represented in the form of a vector by learning mapping rules through the model. Therefore, how to design adaptive data representation methods is more conducive to the combination of deep learning and specific classification tasks.

Model integration. Most structures of traditional and deep learning models are tried for text classification, including integration methods. RNN requires recursive step by step to get global information. CNN can obtain local information, and the sensing field can be increased through the multi-layer stack to capture more comprehensive contextual information. Attention mechanisms learn global dependency among words in a sentence. The transformer model is dependent on attention mechanisms to establish the depth of the global dependency relationship between the input and output. Therefore, designing an integrated model is worth trying to take advantage of these models.

Model efficiency. Although text classification models based on deep learning are highly effective, such as CNNs, RNNs, and GNNs. However, there are many technical limitations, such as the depth of the network layer, regularization problem, network learning rate, etc. Therefore, there is still more broad space for development to optimize the algorithm and improve the speed of model training.

5.3 Challenges from Performance Perspective

The traditional model and the deep model can achieve good performance in most text classification tasks, but the anti-interference ability of their results needs to be improved [304] [176] [305]. How to realize the interpretation of the deep model is also a technical challenge.

The semantic robustness of the model. In recent years, researchers have designed many models to enhance the accuracy of text classification models. However, when there are some adversarial samples in the datasets, the model's performance decreases significantly. Adversarial training is a crucial method to improve the robustness of the pre-training model. For example, a popular approach is converting attack into defense and using the adversarial sample training model. Consequently, how to improve the robustness of models is a current research hotspot and challenge.

The interpretability of the model. DNNs have unique advantages in feature extraction and semantic mining and have achieved excellent text classification tasks. Only a better understanding of the theories behind these models can accurately design better models for various applications. However, deep learning is a black-box model, the training process is challenging to reproduce, and the implicit semantics and output interpretability are poor. It makes the improvement and optimization of the model, losing clear guidelines. Why does one model outperform another on one data set but underperform on others? What does the deep learning model learn? Furthermore, we cannot accurately explain why the model improves performance.

6 CONCLUSION

This paper principally introduces the existing models for text classification tasks from traditional models to deep learning. Firstly, we introduce some primary traditional models and deep learning models with a summary table. The traditional model improves text classification performance mainly by improving the feature extraction scheme and classifier design. In contrast, the deep learning model enhances performance by improving the presentation learning method, model structure, and additional data and knowledge. Then, we introduce the datasets with a summary table and evaluation metrics for single-label and multi-label tasks. Furthermore, we give the quantitative results of the leading models in a summary table under different applications for classic text classification datasets. Finally, we summarize the possible future research challenges of text classification.

ACKNOWLEDGMENTS

The authors of this paper were supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grants (No.U20B2053 and 61872022), State Key Laboratory of Software Development Environment (SKLSDE-2020ZX-12). Philip S. Yu was supported by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941. Lifang He was supported by NSF ONR N00014-18-1-2009 and Lehigh's accelerator grant S00010293. This work was also sponsored by CAAI-Huawei MindSpore Open Fund. Thanks for computing infrastructure provided by Huawei MindSpore platform.

REFERENCES

[1] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. ACL, 2015*, pp. 1556–1566, 2015.
[2] X. Zhu, P. Sobhani, and H. Guo, "Long short-term memory over recursive structures," in *Proc. ICML, 2015*, pp. 1604–1612, 2015.
[3] J. Chen, Z. Gong, and W. Liu, "A dirichlet process biterm-based mixture model for short text stream clustering," *Appl. Intell.*, vol. 50, no. 5, pp. 1609–1619, 2020.
[4] J. Chen, Z. Gong, and W. Liu, "A nonparametric model for online topic discovery with word embeddings," *Inf. Sci.*, vol. 504, pp. 32–47, 2019.
[5] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. ACL, 2014*, pp. 655–665, 2014.
[6] P. Liu, X. Qiu, X. Chen, S. Wu, and X. Huang, "Multi-timescale long short-term memory neural network for modelling sentences and documents," in *Proc. EMNLP, 2015*, pp. 2326–2335, 2015.
[7] J. Y. Lee and F. Dernoncourt, "Sequential short-text classification with recurrent and convolutional neural networks," in *Proc. NAACL, 2016*, pp. 515–520, 2016.
[8] M. E. Maron, "Automatic indexing: An experimental inquiry," *J. ACM*, vol. 8, no. 3, pp. 404–417, 1961.
[9] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[10] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. ECML, 1998*, pp. 137–142, 1998.
[11] R. Aly, S. Remus, and C. Biemann, "Hierarchical multi-label classification of text with capsule networks," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2: Student Research Workshop*, pp. 323–330, 2019.
[12] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.
[13] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning based text classification: A comprehensive review," *CoRR*, vol. abs/2004.03705, 2020.
[14] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. ACM SIGKDD, 2016*, pp. 785–794, 2016.

[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proc. NeurIPS, 2017*, pp. 3146–3154, 2017.

[17] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP, 2014*, pp. 1746–1751, 2014.

[18] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Ieee, 2017.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL, 2019*, pp. 4171–4186, 2019.

[20] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[21] W. B. Cavnar, J. M. Trenkle, *et al.*, "N-gram-based text categorization," in *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, vol. 161175, Citeseer, 1994.

[22] "Term frequency by inverse document frequency," in *Encyclopedia of Database Systems*, p. 3035, 2009.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. ICLR, 2013*, 2013.

[24] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. EMNLP, 2014*, pp. 1532–1543, 2014.

[25] M. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. ACM SIGKDD, 2010*, pp. 999–1008, 2010.

[26] K. Schneider, "A new feature selection score for multinomial naive bayes text classification based on kl-divergence," in *Proc. ACL, 2004*, 2004.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[28] W. Dai, G. Xue, Q. Yang, and Y. Yu, "Transferring naive bayes classifiers for text classification," in *Proc. AAAI, 2007*, pp. 540–545, 2007.

[29] A., P., Dempster, N., M., Laird, D., B., and Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, 1977.

[30] M. Granik and V. Mesyura, "Fake news detection using naive bayes classifier," in *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pp. 900–903, 2017.

[31] M. S. Mubarok, K. Adiwijaya, and M. Aldhi, "Aspect-based sentiment analysis to review products using naïve bayes," vol. 1867, p. 020060, 08 2017.

[32] S. Xu, "Bayesian naïve bayes classifiers to text classification," *J. Inf. Sci.*, vol. 44, no. 1, pp. 48–59, 2018.

[33] G. Singh, B. Kumar, L. Gaur, and A. Tyagi, "Comparison between multinomial and bernoulli naïve bayes for text classification," in *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 593–596, 2019.

[34] "20NG Corpus." http://ana.cachopo.org/datasets-for-single-label-text-categorization, 2007.

[35] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery, "Learning to extract symbolic knowledge from the world wide web," in *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA*, pp. 509–516, 1998.

[36] T. Jo, "Using k nearest neighbors for text segmentation with feature similarity," in *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pp. 1–5, 2017.

[37] L. Baoli, L. Qin, and Y. Shiwen, "An adaptive <i>k</i>-nearest neighbor text categorization strategy," *ACM Transactions on Asian Language Information Processing*, vol. 3, p. 215–226, Dec. 2004.

[38] S. Chen, "K-nearest neighbor algorithm optimization in text categorization," *IOP Conference Series: Earth and Environmental Science*, vol. 108, p. 052074, jan 2018.

[39] S. Jiang, G. Pang, M. Wu, and L. Kuang, "An improved k-nearest-neighbor algorithm for text categorization," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1503–1509, 2012.

[40] P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," in *Proc. ICDM, 2001*, pp. 647–648, 2001.

[41] S. Tan, "Neighbor-weighted k-nearest neighbor for unbalanced text corpus," *Expert Syst. Appl.*, vol. 28, no. 4, pp. 667–671, 2005.

[42] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[43] C. Leslie, E. Eskin, and W. S. Noble, "The spectrum kernel: A string kernel for svm protein classification," in *Biocomputing 2002*, pp. 564–575, World Scientific, 2001.

[44] H. Taira and M. Haruno, "Feature selection in svm text categorization," in *AAAI/IAAI*, pp. 480–486, 1999.

[45] X. Li and Y. Guo, "Active learning with multi-label svm classification.," in *IjCAI*, pp. 1479–1485, Citeseer, 2013.

[46] T. Peng, W. Zuo, and F. He, "Svm based adaptive learning method for text classification from positive and unlabeled documents," *Knowledge and Information Systems*, vol. 16, no. 3, pp. 281–301, 2008.

[47] T. Joachims, "A statistical learning model of text classification for support vector machines," in *Proc. SIGIR, 2001*, pp. 128–136, 2001.

[48] T. JOACHIMS, "Transductive inference for text classification using support vector macines," in *International Conference on Machine Learning*, 1999.

[49] T. M. Mitchell, *Machine learning.* McGraw Hill series in computer science, McGraw-Hill, 1997.

[50] R. Rastogi and K. Shim, "PUBLIC: A decision tree classifier that integrates building and pruning," *Data Min. Knowl. Discov.*, vol. 4, no. 4, pp. 315–344, 2000.

[51] R. J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[52] J. R. Quinlan, *C4.5: Programs for Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[53] M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han, "Generalization and decision tree induction: efficient classification in data mining," in *Proceedings Seventh International Workshop on Research Issues in Data Engineering. High Performance Database Management for Large-Scale Applications*, pp. 111–120, IEEE, 1997.

[54] D. E. Johnson, F. J. Oles, T. Zhang, and T. Götz, "A decision-tree-based symbolic rule induction system for text categorization," *IBM Syst. J.*, vol. 41, no. 3, pp. 428–437, 2002.

[55] P. Vateekul and M. Kubat, "Fast induction of multiple decision trees in text categorization from large scale, imbalanced, and multi-label data," in *Proc. ICDM Workshops, 2009*, pp. 320–325, 2009.

[56] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. EuroCOLT, 1995*, pp. 23–37, 1995.

[57] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, 1999.

[58] A. Bouaziz, C. Dartigues-Pallez, C. da Costa Pereira, F. Precioso, and P. Lloret, "Short text classification using semantic random forest," in *Proc. DAWAK, 2014*, pp. 288–299, 2014.

[59] M. Z. Islam, J. Liu, J. Li, L. Liu, and W. Kang, "A semantics aware random forest for text classification," in *Proc. CIKM, 2019*, pp. 1061–1070, 2019.

[60] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proc. EMNLP, 2011*, pp. 151–161, 2011.

[61] "A MATLAB implementation of RAE." https://github.com/vin00/Semi-Supervised-Recursive-Autoencoders-for-Predicting-Sentiment-Distributions, 2011.

[62] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proc. EMNLP, 2012*, pp. 1201–1211, 2012.

[63] "A Tensorflow implementation of MV_RNN." https://github.com/github-pengge/MV_RNN, 2012.

[64] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP, 2013*, pp. 1631–1642, 2013.

[65] "A MATLAB implementation of RNTN." https://github.com/pondruska/DeepSentiment, 2013.

[66] O. Irsoy and C. Cardie, "Deep recursive neural networks for compositionality in language," in *Proc. NIPS, 2014*, pp. 2096–2104, 2014.

[67] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. ICML, 2014*, pp. 1188–1196, 2014.

[68] "A PyTorch implementation of Paragraph Vectors (doc2vec)." https://github.com/inejc/paragraph-vectors, 2014.

[69] M. Iyyer, V. Manjunatha, J. L. Boyd-Graber, and H. D. III, "Deep unordered composition rivals syntactic methods for text classification," in *Proc. ACL, 2015*, pp. 1681–1691, 2015.

[70] "An implementation of DAN." https://github.com/miyyer/dan, 2015.

[71] "A PyTorch implementation of Tree-LSTM." https://github.com/stanfordnlp/treelstm, 2015.

[72] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," AAAI'15, p. 2267–2273, AAAI Press, 2015.

[73] "A Tensorflow implementation of TextRCNN." https://github.com/roomylee/rcnn-text-classification, 2015.

[74] "An implementation of MT-LSTM." https://github.com/AlexAntn/MTLSTM, 2015.

[75] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," in *Proc. ICML, 2016*, pp. 526–534, 2016.

[76] "An implementation of oh-2LSTMp." http://riejohnson.com/cnn_20download.html, 2015.

[77] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling," in *Proc. COLING, 2016*, pp. 3485–3495, 2016.

[78] "An implementation of BLSTM-2DCNN." https://github.com/ManuelVs/NNForTextClassification, 2016.

[79] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," in *Proc. IJCAI, 2016*, pp. 2873–2879, 2016.

[80] "A PyTorch implementation of Multi-Task." https://github.com/baixl/text_classification, 2016.

[81] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm," in *Proc. EMNLP, 2017*, pp. 1615–1625, 2017.
[82] "A Keras implementation of DeepMoji." https://github.com/bfelbo/DeepMoji, 2018.
[83] A. B. Dieng, C. Wang, J. Gao, and J. W. Paisley, "Topicrnn: A recurrent neural network with long-range semantic dependency," in *Proc. ICLR, 2017*, 2017.
[84] "A PyTorch implementation of TopicRNN." https://github.com/dangitstam/topic-rnn, 2017.
[85] T. Miyato, A. M. Dai, and I. J. Goodfellow, "Adversarial training methods for semi-supervised text classification," in *Proc. ICLR, 2017*, 2017.
[86] "A Tensorflow implementation of Virtual adversarial training." https://github.com/tensorflow/models/tree/master/adversarial_text, 2017.
[87] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, "Sentiment analysis by capsules," in *Proc. WWW, 2018*, pp. 1165–1174, 2018.
[88] "A PyTorch implementation of RNN-Capsule." https://github.com/wangjiosw/Sentiment-Analysis-by-Capsules, 2018.
[89] Y. Zhao, Y. Shen, and J. Yao, "Recurrent neural network for text classification with hierarchical multiscale dense connections," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 5450–5456, 2019.
[90] "An implementation of HM-DenseRNNs." https://github.com/zhaoyizhaoyi/hm-densernns, 2019.
[91] "A Keras implementation of TextCNN." https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras, 2014.
[92] "A Tensorflow implementation of DCNN." https://github.com/kinimod23/ATS_Project, 2014.
[93] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. NeurIPS, 2015*, pp. 649–657, 2015.
[94] "A Tensorflow implementation of CharCNN." https://github.com/mhjabreel/CharCNN, 2015.
[95] "A Keras implementation of SeqTextRCNN." https://github.com/ilimugur/short-text-classification, 2016.
[96] J. Liu, W. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *Proc. ACM SIGIR, 2017*, pp. 115–124, 2017.
[97] "A Pytorch implementation of XML-CNN." https://github.com/siddsax/XML-CNN, 2017.
[98] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," in *Proc. ACL, 2017*, pp. 562–570, 2017.
[99] "A PyTorch implementation of DPCNN." https://github.com/Cheneng/DPCNN, 2017.
[100] J. Wang, Z. Wang, D. Zhang, and J. Yan, "Combining knowledge with deep convolutional neural networks for short text classification," in *Proc. IJCAI, 2017*, pp. 2915–2921, 2017.
[101] M. Yang, W. Zhao, J. Ye, Z. Lei, Z. Zhao, and S. Zhang, "Investigating capsule networks with dynamic routing for text classification," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 3110–3119, 2018.
[102] "A Tensorflow implementation of TextCapsule." https://github.com/andyweizhao/capsule_text_classification, 2018.
[103] K. Shimura, J. Li, and F. Fukumoto, "HFT-CNN: learning hierarchical category structure for multi-label short text categorization," in *Proc. EMNLP, 2018*, pp. 811–816, 2018.
[104] "An implementation of HFT-CNN." https://github.com/ShimShim46/HFT-CNN, 2018.
[105] J. Xu and Y. Cai, "Incorporating context-relevant knowledge into convolutional neural networks for short text classification," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 10067–10068, 2019.
[106] Y. Bao, M. Wu, S. Chang, and R. Barzilay, "Few-shot text classification with distributional signatures," in *Proc. ICLR, 2020*, 2020.
[107] "A PyTorch implementation of few-shot text classification with distributional signatures." https://github.com/YujiaBao/Distributional-Signatures, 2020.
[108] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical attention networks for document classification," in *Proc. NAACL, 2016*, pp. 1480–1489, 2016.
[109] "A Keras implementation of TextCNN." https://github.com/richliao/textClassifier, 2014.
[110] X. Zhou, X. Wan, and J. Xiao, "Attention-based LSTM network for cross-lingual sentiment classification," in *Proc. EMNLP, 2016*, pp. 247–256, 2016.
[111] "NLP&CC Corpus." http://tcci.ccf.org.cn/conference/2013/index.html, 2013.
[112] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proc. EMNLP, 2016*, pp. 551–561, 2016.
[113] "A Tensorflow implementation of LSTMN." https://github.com/JRC1995/Abstractive-Summarization, 2016.

[114] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” in *Proc. ICLR, 2017*, 2017.
[115] “A PyTorch implementation of Structured-Self-Attention.” https://github.com/kaushalshetty/Structured-Self-Attention, 2017.
[116] P. Yang, X. Sun, W. Li, S. Ma, W. Wu, and H. Wang, “SGM: sequence generation model for multi-label classification,” in *Proc. COLING, 2018*, pp. 3915–3926, 2018.
[117] “A PyTorch implementation of SGM.” https://github.com/lancopku/SGM, 2018.
[118] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proc. NAACL, 2018*, pp. 2227–2237, 2018.
[119] “A PyTorch implementation of ELMo.” https://github.com/flairNLP/flair, 2018.
[120] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang, “Bi-directional block self-attention for fast and memory-efficient sequence modeling,” in *Proc. ICLR, 2018*, 2018.
[121] “A PyTorch implementation of BiBloSA.” https://github.com/galsang/BiBloSA-pytorch, 2018.
[122] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu, “Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification,” in *Proc. NeurIPS, 2019*, pp. 5812–5822, 2019.
[123] “A PyTorch implementation of AttentionXML.” https://github.com/yourh/AttentionXML, 2019.
[124] S. Sun, Q. Sun, K. Zhou, and T. Lv, “Hierarchical attention prototypical networks for few-shot text classification,” in *Proc. EMNLP, 2019*, pp. 476–485, 2019.
[125] T. Gao, X. Han, Z. Liu, and M. Sun, “Hybrid attention-based prototypical networks for noisy few-shot relation classification,” in *Proc. AAAI, 2019*, pp. 6407–6414, 2019.
[126] “A PyTorch implementation of HATT-Proto.” https://github.com/thunlp/HATT-Proto, 2019.
[127] J. Chen, Y. Hu, J. Liu, Y. Xiao, and H. Jiang, “Deep short text classification with knowledge powered attention,” in *Proc. AAAI, 2019*, pp. 6252–6259, 2019.
[128] “A PyTorch implementation of STCKA.” https://github.com/AIRobotZhang/STCKA, 2019.
[129] K. Ding, J. Wang, J. Li, D. Li, and H. Liu, “Be more with less: Hypergraph attention networks for inductive text classification,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 4927–4936, 2020.
[130] “A pytorch implementation of HyperGAT.” https://github.com/kaize0409/HyperGAT, 2020.
[131] Q. Guo, X. Qiu, P. Liu, X. Xue, and Z. Zhang, “Multi-scale self-attention for text classification,” in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7847–7854, 2020.
[132] S. Choi, H. Park, J. Yeo, and S. Hwang, “Less is more: Attention supervision with counterfactuals for text classification,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6695–6704, 2020.
[133] “A Tensorflow implementation of BERT.” https://github.com/google-research/bert, 2019.
[134] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos, “Large-scale multi-label text classification on EU legislation,” in *Proc. ACL, 2019*, pp. 6314–6322, 2019.
[135] “A Tensorflow implementation of BERT-BASE.” https://github.com/iliaschalkidis/lmtc-eurlex57k, 2019.
[136] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune BERT for text classification?,” in *Proc. CCL, 2019*, pp. 194–206, 2019.
[137] “A Tensorflow implementation of BERT4doc-Classification.” https://github.com/xuyige/BERT4doc-Classification, 2019.
[138] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *Proc. NeurIPS, 2019*, pp. 5754–5764, 2019.
[139] “A Tensorflow implementation of XLNet.” https://github.com/zihangdai/xlnet, 2019.
[140] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized BERT pretraining approach,” *CoRR*, vol. abs/1907.11692, 2019.
[141] “A PyTorch implementation of RoBERTa.” https://github.com/pytorch/fairseq, 2019.
[142] D. Croce, G. Castellucci, and R. Basili, “GAN-BERT: generative adversarial learning for robust text classification with a bunch of labeled examples,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 2114–2119, 2020.
[143] “A pytorch implementation of GAN-BERT.” https://github.com/crux82/ganbert, 2020.
[144] S. Garg and G. Ramakrishnan, “BAE: bert-based adversarial examples for text classification,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 6174–6181, 2020.

[145] "An implementation of BAE." https://github.com/QData/TextAttack/blob/master/textattack/attack_recipes/bae_garg_2019.py, 2020.

[146] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *Proc. ICLR, 2020*, 2020.

[147] "A Tensorflow implementation of ALBERT." https://github.com/google-research/ALBERT, 2020.

[148] H. Zhang and J. Zhang, "Text graph transformer for document classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 8322–8327, 2020.

[149] W. Chang, H. Yu, K. Zhong, Y. Yang, and I. S. Dhillon, "Taming pretrained transformers for extreme multi-label text classification," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 3163–3171, 2020.

[150] "An implementation of X-Transformer." https://github.com/OctoberChang/X-Transformer, 2020.

[151] T. Jiang, D. Wang, L. Sun, H. Yang, Z. Zhao, and F. Zhuang, "Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification," *CoRR*, vol. abs/2101.03305, 2021.

[152] "An implementation of LightXML." https://github.com/kongds/LightXML, 2021.

[153] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proc. WWW, 2018*, pp. 1063–1072, 2018.

[154] "A Tensorflow implementation of DeepGraphCNNforTexts." https://github.com/HKUST-KnowComp/DeepGraphCNNforTexts, 2018.

[155] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI, 2019*, pp. 7370–7377, 2019.

[156] "A Tensorflow implementation of TextGCN." https://github.com/yao8839836/text_gcn, 2019.

[157] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. ICML, 2019*, pp. 6861–6871, 2019.

[158] "An implementation of SGC." https://github.com/Tiiiger/SGC, 2019.

[159] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, "Text level graph neural network for text classification," in *Proc. EMNLP, 2019*, pp. 3442–3448, 2019.

[160] "An implementation of TextLevelGNN." https://github.com/LindgeW/TextLevelGNN, 2019.

[161] H. Peng, J. Li, S. Wang, L. Wang, Q. Gong, R. Yang, B. Li, P. Yu, and L. He, "Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[162] Y. Zhang, X. Yu, Z. Cui, S. Wu, Z. Wen, and L. Wang, "Every document owns its structure: Inductive text classification via graph neural networks," in *Proc. ACL, 2020*, pp. 334–339, 2020.

[163] "A Tensorflow implementation of TextING." https://github.com/CRIPAC-DIG/TextING, 2019.

[164] X. Liu, X. You, X. Zhang, J. Wu, and P. Lv, "Tensor graph convolutional networks for text classification," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8409–8416, 2020.

[165] "A Tensorflow implementation of TensorGCN." https://github.com/THUMLP/TensorGCN, 2019.

[166] A. Pal, M. Selvakumar, and M. Sankarasubbu, "MAGNET: multi-label text classification using attention-based graph neural network," in *Proc. ICAART, 2020*, pp. 494–505, 2020.

[167] "A repository of MAGNET." https://github.com/monk1337/MAGnet, 2020.

[168] "A Tensorflow implementation of Miyato et al.." https://github.com/TobiasLee/Text-Classification, 2017.

[169] J. Zeng, J. Li, Y. Song, C. Gao, M. R. Lyu, and I. King, "Topic memory networks for short text classification," in *Proc. EMNLP, 2018*, pp. 3120–3131, 2018.

[170] J. Zhang, P. Lertvittayakumjorn, and Y. Guo, "Integrating semantic knowledge to tackle zero-shot text classification," in *Proc. NAACL, 2019*, pp. 1031–1040, 2019.

[171] "A Tensorflow implementation of KG4ZeroShotText." https://github.com/JingqingZ/KG4ZeroShotText, 2019.

[172] M. k. Alsmadi, K. B. Omar, S. A. Noah, and I. Almarashdah, "Performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks," in *2009 IEEE International Advance Computing Conference*, pp. 296–299, 2009.

[173] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. E. P. Reyes, M. Shyu, S. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 92:1–92:36, 2019.

[174] L. Qin, W. Che, Y. Li, M. Ni, and T. Liu, "Dcr-net: A deep co-interactive relation network for joint dialog act recognition and sentiment classification," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8665–8672, 2020.

[175] Z. Deng, H. Peng, D. He, J. Li, and P. S. Yu, "Htcinfomax: A global model for hierarchical text classification via information maximization," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021* (K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, eds.), pp. 3259–3265, Association for Computational Linguistics, 2021.

[176] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT really robust? A strong baseline for natural language attack on text classification and entailment," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8018–8025, 2020.

[177] R. Ragesh, S. Sellamanickam, A. Iyer, R. Bairi, and V. Lingam, "Hetegcn: heterogeneous graph convolutional networks for text classification," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 860–868, 2021.

[178] T. Miyato, S. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *CoRR*, vol. abs/1704.03976, 2017.

[179] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. ICANN, 2011* (T. Honkela, W. Duch, M. Girolami, and S. Kaski, eds.), (Berlin, Heidelberg), pp. 44–51, Springer Berlin Heidelberg, 2011.

[180] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[181] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proc. EMNLP, 2015*, pp. 632–642, 2015.

[182] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," in *Proc. IJCAI, 2017*, pp. 4144–4150, 2017.

[183] R. Johnson and T. Zhang, "Semi-supervised convolutional neural networks for text categorization via region embedding," in *Proc. NeurIPS, 2015*, pp. 919–927, 2015.

[184] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. ECCV, 2016*, pp. 630–645, 2016.

[185] I. Bazzi, *Modelling out-of-vocabulary words for robust speech recognition.* PhD thesis, Massachusetts Institute of Technology, 2002.

[186] H. Nguyen and M. Nguyen, "A deep neural architecture for sentence-level sentiment classification in twitter social networking," in *Proc. PACLING, 2017*, pp. 15–27, 2017.

[187] B. Adams and G. McKenzie, "Crowdsourcing the character of a place: Character-level convolutional networks for multilingual geographic text classification," *Trans. GIS*, vol. 22, no. 2, pp. 394–408, 2018.

[188] Z. Chen and T. Qian, "Transfer capsule network for aspect level sentiment classification," in *Proc. ACL, 2019*, pp. 547–556, 2019.

[189] W. Xue, W. Zhou, T. Li, and Q. Wang, "MTNA: A neural multi-task model for aspect category classification and aspect term extraction on restaurant reviews," in *Proc. IJCNLP, 2017*, pp. 151–156, 2017.

[190] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR, 2015*, 2015.

[191] Z. Hu, X. Li, C. Tu, Z. Liu, and M. Sun, "Few-shot charge prediction with discriminative legal attributes," in *Proc. COLING, 2018*, pp. 487–498, 2018.

[192] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS, 2017*, pp. 5998–6008, 2017.

[193] Y. Ma, H. Peng, and E. Cambria, "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM," in *Proc. AAAI, 2018*, pp. 5876–5883, 2018.

[194] Y. Wang, M. Huang, X. Zhu, and L. Zhao, "Attention-based LSTM for aspect-level sentiment classification," in *Proc. EMNLP, 2016*, pp. 606–615, 2016.

[195] F. Fan, Y. Feng, and D. Zhao, "Multi-grained attention network for aspect-level sentiment classification," in *Proc. EMNLP, 2018*, pp. 3433–3442, 2018.

[196] M. Tan, C. dos Santos, B. Xiang, and B. Zhou, "Improved representation learning for question answer matching," in *Proc. ACL, 2016*, (Berlin, Germany), pp. 464–473, Association for Computational Linguistics, Aug. 2016.

[197] C. N. dos Santos, M. Tan, B. Xiang, and B. Zhou, "Attentive pooling networks," *CoRR*, vol. abs/1602.03609, 2016.

[198] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *CoRR*, vol. abs/2003.08271, 2020.

[199] A. Radford, "Improving language understanding by generative pre-training," 2018.

[200] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proc. ACL, 2019*, pp. 2978–2988, 2019.

[201] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?," in *Proc. ACL, 2019*, pp. 3651–3657, 2019.

[202] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 7871–7880, 2020.

[203] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 64–77, 2020.

[204] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, "ERNIE: enhanced representation through knowledge integration," *CoRR*, vol. abs/1904.09223, 2019.

[205] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *CoRR*, vol. abs/1506.05163, 2015.

[206] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NeurIPS, 2016*, pp. 3837–3845, 2016.

[207] C. Li, X. Peng, H. Peng, J. Li, and L. Wang, "Textgtl: Graph-based transductive learning for semi-supervised textclassification via structure-sensitive interpolation," in *IJCAI 2021*, ijcai.org, 2021.

[208] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. EMNLP, 2017*, pp. 1506–1515, 2017.

[209] Y. Li, R. Jin, and Y. Luo, "Classifying relations in clinical narratives using segment graph convolutional and recurrent neural networks (seg-gcrns)," *JAMIA*, vol. 26, no. 3, pp. 262–268, 2019.

[210] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," in *Proc. EMNLP, 2017*, pp. 1957–1967, 2017.

[211] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. ICLR, 2018*, 2018.

[212] L. Hu, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *Proc. EMNLP, 2019*, pp. 4820–4829, 2019.

[213] Z. Li, X. Ding, and T. Liu, "Constructing narrative event evolutionary graph for script event prediction," in *Proc. IJCAI, 2018*, pp. 4201–4207, 2018.

[214] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," in *Proc. NeurIPS, 1993]*, pp. 737–744, 1993.

[215] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. AAAI, 2016*, pp. 2786–2792, 2016.

[216] Jayadeva, H. Pant, M. Sharma, and S. Soman, "Twin neural networks for the classification of large unbalanced datasets," *Neurocomputing*, vol. 343, pp. 34 – 49, 2019. Learning in the Presence of Class Imbalance and Concept Drift.

[217] T. Miyato, S. ichi Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," 2015.

[218] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *Proc. AAAI, 2018*, pp. 6053–6060, 2018.

[219] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2015.

[220] X. Li and W. Lam, "Deep multi-task learning for aspect term extraction with memory interaction," in *Proc. EMNLP, 2017*, pp. 2886–2892, 2017.

[221] C. Shen, C. Sun, J. Wang, Y. Kang, S. Li, X. Liu, L. Si, M. Zhang, and G. Zhou, "Sentiment classification towards question-answering with hierarchical matching network," in *Proc. EMNLP, 2018*, pp. 3654–3663, 2018.

[222] X. Ding, K. Liao, T. Liu, Z. Li, and J. Duan, "Event representation learning enhanced with external commonsense knowledge," in *Proc. EMNLP, 2019*, pp. 4893–4902, 2019.

[223] Y. Zhang, D. Song, P. Zhang, X. Li, and P. Wang, "A quantum-inspired sentiment representation model for twitter sentiment analysis," *Applied Intelligence*, 2019.

[224] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019.

[225] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8BERT: quantized 8bit BERT," in *Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition, EMC2@NeurIPS 2019, Vancouver, Canada, December 13, 2019*, pp. 36–39, 2019.

[226] "MR Corpus." http://www.cs.cornell.edu/people/pabo/movie-review-data/, 2002.

[227] "SST Corpus." http://nlp.stanford.edu/sentiment, 2013.

[228] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP, 2013*, (Seattle, Washington, USA), pp. 1631–1642, Association for Computational Linguistics, Oct. 2013.

[229] "MPQA Corpus." http://www.cs.pitt.edu/mpqa/, 2005.

[230] Q. Diao, M. Qiu, C. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proc. ACM SIGKDD, 2014*, pp. 193–202, 2014.
[231] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. EMNLP, 2015*, pp. 1422–1432, 2015.
[232] "Amazon review Corpus." https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products, 2015.
[233] "Twitter Corpus." https://www.cs.york.ac.uk/semeval-2013/task2/, 2013.
[234] "AG Corpus." http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, 2004.
[235] "Reuters Corpus." https://www.cs.umb.edu/~smimarog/textmining/datasets/, 2007.
[236] C. Wang, M. Zhang, S. Ma, and L. Ru, "Automatic online news issue construction in web environment," in *Proc. WWW, 2008*, pp. 457–466, 2008.
[237] X. Li, C. Li, J. Chi, J. Ouyang, and C. Li, "Dataless text classification: A topic modeling approach with document manifold," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pp. 973–982, 2018.
[238] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
[239] "Ohsumed Corpus." http://davis.wpi.edu/xmdv/datasets/ohsumed.html, 2015.
[240] "EUR-Lex Corpus." http://www.ke.tu-darmstadt.de/resources/eurlex/eurlex.html, 2019.
[241] "Amazon670K Corpus." http://manikvarma.org/downloads/XC/XMLRepository.html, 2016.
[242] J. Yin and J. Wang, "A dirichlet multinomial mixture model-based approach for short text clustering," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 233–242, 2014.
[243] J. Chen, Z. Gong, W. Wang, W. Liu, M. Yang, and C. Wang, "Tam: Targeted analysis model with reinforcement learning on short texts," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2020.
[244] F. Wang, Z. Wang, Z. Li, and J. Wen, "Concept-based short text classification and ranking," in *Proc. CIKM, 2014*, pp. 1069–1078, 2014.
[245] "Fudan Corpus." www.datatang.com/data/44139and43543, 2015.
[246] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100, 000+ questions for machine comprehension of text," in *Proc. EMNLP, 2016*, pp. 2383–2392, 2016.
[247] X. Yao, B. V. Durme, C. Callison-Burch, and P. Clark, "Answer extraction as sequence tagging with tree edit distance," in *Proc. NAACL, 2013*, pp. 858–867, 2013.
[248] "TREC Corpus." https://cogcomp.seas.upenn.edu/Data/QA/QC/, 2002.
[249] Y. Yang, W. Yih, and C. Meek, "Wikiqa: A challenge dataset for open-domain question answering," in *Proc. EMNLP, 2015*, pp. 2013–2018, 2015.
[250] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. ACL, 2004*, (Barcelona, Spain), pp. 271–278, July 2004.
[251] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. ACM SIGKDD, 2004*, pp. 168–177, 2004.
[252] "Reuters Corpus." https://martin-thoma.com/nlp-reuters, 2017.
[253] J. Kim, S. Jang, E. L. Park, and S. Choi, "Text classification using capsules," *Neurocomputing*, vol. 376, pp. 214–221, 2020.
[254] "Reuters10 Corpus." http://www.nltk.org/book/ch02.html, 2020.
[255] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, 2004.
[256] "RCV1-V2 Corpus." http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm, 2004.
[257] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. ACL, 2005*, pp. 115–124, 2005.
[258] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language Resources and Evaluation*, vol. 39, no. 2-3, pp. 165–210, 2005.
[259] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *J. Assoc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, 2012.
[260] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, "Semeval-2016 task 4: Sentiment analysis in twitter," in *Proc. SemEval, 2016)*, 2016.
[261] Z. Lu, "Pubmed and beyond: a survey of web tools for searching biomedical literature," *Database*, vol. 2011, 2011.
[262] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "MS MARCO: A human generated machine reading comprehension dataset," in *Proc. NeurIPS, 2016*, 2016.
[263] https://data.quora.com/First-Quora-Dataset-Release-QuestionPairs.

[264] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in *Proc. ACL, 2018*, pp. 784–789, 2018.
[265] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proc. NAACL, 2018*, pp. 1112–1122, 2018.
[266] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment," in *Proc. SemEval, 2014*, pp. 1–8, 2014.
[267] B. Dolan, C. Quirk, and C. Brockett, "Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources," in *Proc. COLING, 2004*, 2004.
[268] D. M. Cer, M. T. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation," *CoRR*, vol. abs/1708.00055, 2017.
[269] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL recognising textual entailment challenge," in *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, pp. 177–190, 2005.
[270] T. Khot, A. Sabharwal, and P. Clark, "Scitail: A textual entailment dataset from science question answering," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 5189–5197, 2018.
[271] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *Proc. ICMLA, 2017*, pp. 364–371, 2017.
[272] "AmazonCat-13K Corpus." https://drive.google.com/open?id=1VwHAbri6y6oh8lkpZ6sSY_b1FRNnCLFL, 2018.
[273] "BlurbGenreCollection-EN Corpus." https://www.inf.uni-hamburg.de/en/inst/ab/lt/resources/data/blurb-genre-collection.html, 2017.
[274] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proc. NAACL, 2009*, pp. 94–99, 2009.
[275] S. M. Strassel, M. A. Przybocki, K. Peterson, Z. Song, and K. Maeda, "Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction," in *Proc. LREC, 2008*, 2008.
[276] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proc. EMNLP, 2017*, pp. 35–45, 2017.
[277] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Proc. ECML PKDD, 2010*, pp. 148–163, 2010.
[278] "FewRel Corpus." https://github.com/thunlp/FewRel, 2019.
[279] S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, and M. Henderson, "The fourth dialog state tracking challenge," in *Proc. IWSDS, 2016*, pp. 435–449, 2016.
[280] J. Ang, Y. Liu, and E. Shriberg, "Automatic dialog act segmentation and classification in multiparty meetings," in *Proc. ICASSP, 2005*, pp. 1061–1064, 2005.
[281] D. Jurafsky and E. Shriberg, "Switchboard swbd-damsl shallow-discourse-function annotation coders manual," 01 1997.
[282] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015* (R. Baeza-Yates, M. Lalmas, A. Moffat, and B. A. Ribeiro-Neto, eds.), pp. 373–382, ACM, 2015.
[283] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval.* Cambridge University Press, 2008.
[284] T. Nakagawa, K. Inui, and S. Kurohashi, "Dependency tree-based sentiment classification using crfs with hidden variables," in *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pp. 786–794, 2010.
[285] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. ACL, 2018*, pp. 328–339, 2018.
[286] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint embedding of words and labels for text classification," in *Proc. ACL, 2018*, pp. 2321–2331, 2018.
[287] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," in *Proc. ACL, 2019*, pp. 4487–4496, 2019.
[288] P. K. Pushp and M. M. Srivastava, "Train once, test anywhere: Zero-shot learning for text classification," *CoRR*, vol. abs/1712.05972, 2017.

[289] C. Song, S. Zhang, N. Sadoughi, P. Xie, and E. P. Xing, "Generalized zero-shot text classification for ICD coding," in *Proc. IJCAI, 2020*, pp. 4018–4024, 2020.

[290] R. Geng, B. Li, Y. Li, X. Zhu, P. Jian, and J. Sun, "Induction networks for few-shot text classification," in *Proc. EMNLP, 2019*, pp. 3902–3911, 2019.

[291] S. Deng, N. Zhang, Z. Sun, J. Chen, and H. Chen, "When low resource NLP meets unsupervised language model: Meta-pretraining then meta-learning for few-shot text classification (student abstract)," in *Proc. AAAI, 2020*, pp. 13773–13774, 2020.

[292] R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu, "Dynamic memory induction networks for few-shot text classification," in *Proc. ACL, 2020*, pp. 1087–1094, 2020.

[293] K. R. Rojas, G. Bustamante, A. Oncevay, and M. A. S. Cabezudo, "Efficient strategies for hierarchical text classification: External knowledge and auxiliary tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 2252–2257, 2020.

[294] N. Shanavas, H. Wang, Z. Lin, and G. I. Hawe, "Knowledge-driven graph similarity for text classification," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 4, pp. 1067–1081, 2021.

[295] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proc. ACL, 2017*, pp. 221–231, 2017.

[296] R. Türker, L. Zhang, M. Koutraki, and H. Sack, "TECNE: knowledge based text classification using network embeddings," in *Proc. EKAW, 2018*, pp. 53–56, 2018.

[297] X. Liang, D. Cheng, F. Yang, Y. Luo, W. Qian, and A. Zhou, "F-HMTC: detecting financial events for investment decisions based on neural hierarchical multi-label text classification," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 4490–4496, 2020.

[298] S. P. B., S. Modi, K. S. Hareesha, and S. Kumar, "Classification and comparison of malignancy detection of cervical cells based on nucleus and textural features in microscopic images of uterine cervix," *Int. J. Medical Eng. Informatics*, vol. 13, no. 1, pp. 1–13, 2021.

[299] T. Wang, L. Liu, N. Liu, H. Zhang, L. Zhang, and S. Feng, "A multi-label text classification method via dynamic semantic representation model and deep neural network," *Appl. Intell.*, vol. 50, no. 8, pp. 2339–2351, 2020.

[300] B. Wang, X. Hu, P. Li, and P. S. Yu, "Cognitive structure learning model for hierarchical multi-label text classification," *Knowl. Based Syst.*, vol. 218, p. 106876, 2021.

[301] J. Du, Y. Huang, and K. Moilanen, "Pointing to select: A fast pointer-lstm for long text classification," in *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pp. 6184–6193, 2020.

[302] J. Du, C. Vong, and C. L. P. Chen, "Novel efficient RNN and lstm-like architectures: Recurrent and gated broad learning systems and their applications for text classification," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1586–1597, 2021.

[303] T. Kanchinadam, Q. You, K. Westpfahl, J. Kim, S. Gunda, S. Seith, and G. Fung, "A simple yet brisk and efficient active learning platform for text classification," *CoRR*, vol. abs/2102.00426, 2021.

[304] Y. Zhou, J. Jiang, K. Chang, and W. Wang, "Learning to discriminate perturbations for blocking adversarial attacks in text classification," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 4903–4912, 2019.

[305] A. Azizi, I. A. Tahmid, A. Waheed, N. Mangaokar, J. Pu, M. Javed, C. K. Reddy, and B. Viswanath, "T-miner: A generative approach to defend against trojan attacks on dnn-based text classification," *CoRR*, vol. abs/2103.04264, 2021.