Deep Learning–based Text Classification: A Comprehensive Review

SHERVIN MINAEE, Snapchat Inc
NAL KALCHBRENNER, Google Brain, Amsterdam
ERIK CAMBRIA, Nanyang Technological University, Singapore
NARJES NIKZAD and MEYSAM CHENAGHLU, University of Tabriz
JIANFENG GAO, Microsoft Research, Redmond

Deep learning–based models have surpassed classical machine learning–based approaches in various text classification tasks, including sentiment analysis, news categorization, question answering, and natural language inference. In this article, we provide a comprehensive review of more than 150 deep learning–based models for text classification developed in recent years, and we discuss their technical contributions, similarities, and strengths. We also provide a summary of more than 40 popular datasets widely used for text classification. Finally, we provide a quantitative analysis of the performance of different deep learning models on popular benchmarks, and we discuss future research directions.

CCS Concepts: • **General and reference** → *Surveys and overviews*; • **Computing methodologies** → *Natural language processing*; • **Computer systems organization** → *Neural networks*;

Additional Key Words and Phrases: Text classification, sentiment analysis, question answering, news categorization, deep learning, natural language inference, topic classification

ACM Reference format:
Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep Learning–based Text Classification: A Comprehensive Review. *ACM Comput. Surv.* 54, 3, Article 62 (April 2021), 40 pages.
https://doi.org/10.1145/3439726

1 INTRODUCTION

Text classification, also known as text categorization, is a classical problem in **natural language processing (NLP)**, which aims to assign labels or tags to textual units such as sentences, queries, paragraphs, and documents. It has a wide range of applications including question answering, spam detection, sentiment analysis, news categorization, user intent classification, content moderation, and so on. Text data can come from different sources, including web data, emails, chats, social media, tickets, insurance claims, user reviews, and questions and answers from customer

Authors' addresses: S. Minaee, Snapchat Inc., 2025 1st Ave #500, Seattle, WA 98121; email: sminaee@snapchat.com; N. Kalchbrenner, Google Brain, Claude Debussylaan 34, 1082 MD Amsterdam, Netherlands; email: nalk@google.com; E. Cambria, Nanyang Technological University, 50 Nanyang Ave, Singapore 639798; email: cambria@ntu.edu.sg; N. Nikzad and M. Chenaghlu, East Azerbaijan Province, Tabriz, 29 Bahman Boulevard, Iran; emails: {narjes.nikzad, m.asgari.c}@tabrizu.ac.ir; J. Gao, Microsoft Research, 14820 NE 36th St, Redmond, WA 98052; email: jfgao@microsoft.com.

0360-0300/2021/04-ART62 $15.00
https://doi.org/10.1145/3439726

services, to name a few. Text is an extremely rich source of information. But extracting insights from text can be challenging and time-consuming, due to its unstructured nature.

Text classification can be performed either through manual annotation or by automatic labeling. With the growing scale of text data in industrial applications, automatic text classification is becoming increasingly important. Approaches to automatic text classification can be grouped into two categories:

- Rule-based methods
- Machine learning (data-driven)–based methods

Rule-based methods classify text into different categories using a set of pre-defined rules, and require a deep domain knowledge. However, machine learning–based approaches learn to classify text based on observations of data. Using pre-labeled examples as training data, a machine learning algorithm learns inherent associations between texts and their labels.

Machine learning models have drawn lots of attention in recent years. Most classical machine learning–based models follow the two-step procedure. In the first step, some hand-crafted features are extracted from the documents (or any other textual unit). In the second step, those features are fed to a classifier to make a prediction. Popular hand-crafted features include **bag of words (BoW)** and their extensions. Popular choices of classification algorithms include Naïve Bayes, **support vector machines (SVM)**, **hidden Markov model (HMM)**, gradient boosting trees, and random forests. The two-step approach has several limitations. For example, reliance on the hand-crafted features requires tedious feature engineering and analysis to obtain good performance. In addition, the strong dependence on domain knowledge for designing features makes the method difficult to generalize to new tasks. Finally, these models cannot take full advantage of large amounts of training data, because the features (or feature templates) are pre-defined.

Neural approaches have been explored to address the limitations due to the use of hand-craft features. The core component of these approaches is a machine-learned embedding model that maps text into a low-dimensional continuous feature vector, thus no hand-crafted features is needed. One of earliest embedding models is **latent semantic analysis (LSA)** developed by Dumais et al. [1] in 1989. LSA is a linear model with less than 1 million parameters, trained on 200K words. In 2001, Bengio et al. [2] proposed the first neural language model based on a feed-forward neural network trained on 14 million words. However, these early embedding models underperform classical models using hand-crafted features, and thus are not widely adopted. A paradigm shift starts when much larger embedding models are developed using much larger amounts of training data. In 2013, Google developed a series of word2vec models [3] that are trained on 6 billion words and immediately become popular for many NLP tasks. In 2017, the teams from AI2 and University of Washington developed a contextual embedding model based on a 3-layer bidirectional LSTM with 93M parameters trained on 1B words. The model, called ELMo [4], works much better than word2vec, because they capture contextual information. In 2018, OpenAI started building embedding models using Transformer [5], a new NN architecture developed by Google. Transformer is solely based on attention that substantially improves the efficiency of large-scale model training on TPU. Their first model is called GPT [6], which is now widely used for text generation tasks. The same year, Google developed BERT [7] based on bidirectional transformer. BERT consists of 340M parameters, trained on 3.3 billion words, and is the current state of the art embedding model. The trend of using larger models and more training data continues. By the time this article is published, OpenAI's latest GPT-3 model [8] contains 170 billion parameters, and Google's GShard [9] contains 600 billion parameters.

Although these gigantic models show very impressive performance on various NLP tasks, some researchers argue that they do not really understand language and are not robust enough for

many mission-critical domains [10–14]. Recently, there is an growing interest in exploring neuro-symbolic hybrid models (e.g., References [15–18]) to address some of the fundamental limitations of neural models, such as lack of grounding, being unable to perform symbolic reasoning, not interpretable. These works, although important, are beyond the scope of this article.

While there are many good reviews and text books on text classification methods and applications, in general, e.g., References [19–21], this survey is unique in that it presents a comprehensive review on more than 150 **deep learning (DL)** models developed for various text classification tasks, including sentiment analysis, news categorization, topic classification, **question answering (QA)**, and **natural language inference (NLI)**, over the course of the past six years. In particular, we group these works into several categories based on their neural network architectures, including **recurrent neural networks (RNNs)**, **convolutional neural networks (CNNs)**, attention, Transformers, Capsule Nets, and so on. The contributions of this article can be summarized as follows:

- We present a detailed overview of more than 150 DL models proposed for text classification.
- We review more than 40 popular text classification datasets.
- We provide a quantitative analysis of the performance of a selected set of DL models on 16 popular benchmarks.
- We discuss remaining challenges and future directions.

1.1 Text Classification Tasks

Text Classification (TC) is the process of categorizing texts (e.g., tweets, news articles, customer reviews) into organized groups. Typical TC tasks include sentiment analysis, news categorization and topic classification. Recently, researchers show that it is effective to cast many natural language understanding (NLU) tasks (e.g., extractive question answering, natural language inference) as TC by allowing DL-based text classifiers to take a pair of texts as input (e.g., References [7, 22, 23]). This section introduces five TC tasks discussed in this article, including three typical TC tasks and two NLU tasks that are commonly cast as TC in many recent DL studies.

Sentiment Analysis. This is the task of analyzing people's opinions in textual data (e.g., product reviews, movie reviews, or tweets), and extracting their polarity and viewpoint. The task can be cast as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities.

News Categorization. News contents are among the most important information sources. A news classification system helps users obtain information of interest in real-time by, e.g., identifying emerging news topics or recommending relevant news based on user interests.

Topic Analysis. The task, also known as *topic classification*, aims to identify the theme or topics of a text (e.g., whether a product review is about "customer support" or "ease of use").

Question Answering (QA). There are two types of QA tasks: extractive and generative. Extractive QA is a TC task: Given a question and a set of candidate answers (e.g., text spans in a document in SQuAD [24]), a system classifies each candidate answer as correct or not. Generative QA is a text generation task, since it requires generating answers on the fly. This article only discusses extractive QA.

Natural language inference (NLI). NLI, also known as *recognizing textual entailment* **(RTE)**, predicts whether the meaning of one text can be inferred from another. An NLI system needs to assign to a pair of text units a label such as entailment, contradiction, and neutral [25].

Paraphrasing is a generalized form of NLI, also known as *text pair comparison*, the task of measuring the semantic similarity of a sentence pair indicating how likely one sentence is a paraphrase of the other.

1.2 Article Structure

The rest of the article is structured as follows: Section 2 presents a comprehensive review of more than 150 DL-based text classification models. Section 3 presents a recipe of building text classifiers using DL models. Section 4 reviews some of the most popular TC datasets. Section 5 presents a quantitative performance analysis of a selected set of DL models on 16 benchmarks. Section 6 discusses the main challenges and future directions for DL-based TC methods. Section 7 concludes the article.

2 DEEP LEARNING MODELS FOR TEXT CLASSIFICATION

This section reviews more than 150 DL models proposed for various TC tasks. For clarify, we group these models into several categories based on their model architectures[1]:

- Feed-forward networks view text as a bag of words (Section 2.1).
- RNN-based models view text as a sequence of words, and are intended to capture word dependencies and text structures (Section 2.2).
- CNN-based models are trained to recognize patterns in text, such as key phrases, for TC (Section 2.3).
- Capsule networks address the information loss problem suffered by the pooling operations of CNNs, and recently have been applied to TC (Section 2.4).
- The attention mechanism is effective to identify correlated words in text, and has become a useful tool in developing DL models (Section 2.5).
- Memory-augmented networks combine neural networks with a form of external memory, which the models can read from and write to (Section 2.6).
- Graph neural networks are designed to capture internal graph structures of natural language, such as syntactic and semantic parse trees (Section 2.7).
- Siamese Neural Networks are designed for text matching, a special case of TC (Section 2.8).
- Hybrid models combine attention, RNNs, CNNs, and so on, to capture local and global features of sentences and documents (Section 2.9).
- Transformers allow for much more parallelization than RNNs, making it possible to efficiently (pre-)train very big language models using GPUs (Section 2.10).
- Finally, in Section 2.11, we review modeling technologies that are beyond supervised learning, including unsupervised learning using autoencoder and adversarial training, and reinforcement learning.

Readers are expected to be reasonably familiar with basic DL models to comprehend the content of this section. Readers are referred to the DL textbook by Goodfellow et al. [26] for more details.

2.1 Feed-Forward Neural Networks

Feed-forward networks are among the simplest DL models for text representation. Yet, they have achieved high accuracy on many TC benchmarks. These models view text as a bag of words. For each word, they learn a vector representation using an embedding model such as word2vec [27]

[1]These categories are introduced mainly for a pedagogical purpose. They are by no means exclusive to each other. For example, the Transformer uses a composite structure consisting of feed-forward layers and the attention mechanism, and memory-augment networks also involve the attention mechanism.

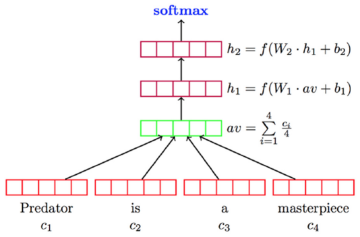

Fig. 1. The architecture of the Deep Average Network (DAN) [29].

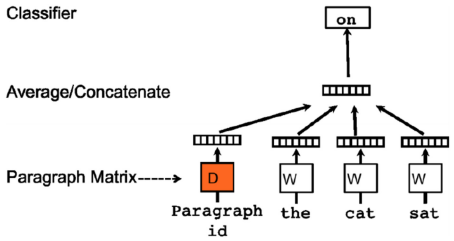

Fig. 2. The doc2vec model [32].

or Glove [28], take the vector sum or average of the embeddings as the representation of the text, pass it through one or more feed-forward layers, known as **Multi-Layer Perceptrons (MLPs)**, and then perform classification on the final layer's representation using a classifier such as logistic regression, Naïve Bayes, or SVM [29]. An example of these models is the **Deep Average Network (DAN)** [29], whose architecture is shown in Figure 1. Despite its simplicity, DAN outperforms other more sophisticated models that are designed to explicitly learn the compositionality of texts. For example, DAN outperforms syntactic models on datasets with high syntactic variance. Joulin et al. [30] propose a simple and efficient text classifier called fastText. Like DAN, fastText views text as a bag of words. Unlike DAN, fastText uses a bag of n-grams as additional features to capture local word order information. This turns out to be very efficient in practice, achieving comparable results to the methods that explicitly use the word order [31].

Le and Mikolov [32] propose doc2vec, which uses an unsupervised algorithm to learn fixed-length feature representations of variable-length pieces of texts, such as sentences, paragraphs, and documents. As shown in Figure 2, the architecture of doc2vec is similar to that of the **Continuous Bag of Words (CBOW)** model [3, 27]. The only difference is the additional paragraph token that is mapped to a paragraph vector via matrix D. In doc2vec, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph. After being trained, the paragraph vector is used as features for the paragraph (e.g., in lieu of or in addition to BoW), and fed to a classifier for prediction. Doc2vec achieves new state of the art results on several TC tasks when it is published.

2.2 RNN-based Models

RNN-based models view text as a sequence of words, and are intended to capture word dependencies and text structures for TC. However, vanilla RNN models do not perform well, and often underperform feed-forward neural networks. Among many variants to RNNs, **Long Short-Term Memory (LSTM)** is the most popular architecture, which is designed to better capture long term dependencies. LSTM addresses the gradient vanishing or exploding problems suffered by the

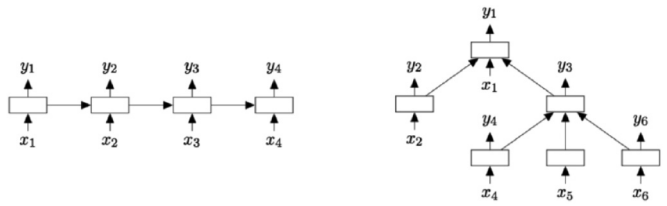

Fig. 3. (Left) A chain-structured LSTM network and (right) a tree-structured LSTM network with arbitrary branching factor [33]. Here x_i and y_i denote the input and output of each cell.

vanilla RNNs by introducing a memory cell to remember values over arbitrary time intervals, and three gates (input gate, output gate, forget gate) to regulate the flow of information into and out of the cell. There have been works on improving RNNs and LSTM models for TC by capturing richer information, such as tree structures of natural language, long-span word relations in text, document topics, and so on.

Tai et al. [33] develop a Tree-LSTM model, a generalization of LSTM to tree-structured network typologies, to learn rich semantic representations. The authors argue that Tree-LSTM is a better model than the chain-structured LSTM for NLP tasks, because natural language exhibits syntactic properties that would naturally combine words to phrases. They validate the effectiveness of Tree-LSTM on two tasks: sentiment classification and predicting the semantic relatedness of two sentences. The architectures of these models are shown in Figure 3. Zhu et al. [34] also extend the chain-structured LSTM to tree structures, using a memory cell to store the history of multiple child cells or multiple descendant cells in a recursive process. They argue that the new model provides a principled way of considering long-distance interaction over hierarchies, e.g., language or image parse structures.

To model long-span word relations for machine reading, Cheng et al. [35] augment the LSTM architecture with a memory network in place of a single memory cell. This enables adaptive memory usage during recurrence with neural attention, offering a way to weakly induce relations among tokens. This model achieves promising results on language modeling, sentiment analysis, and NLI.

The **Multi-Timescale LSTM (MT-LSTM)** neural network [36] is also designed to model long texts, such as sentences and documents, by capturing valuable information with different timescales. MT-LSTM partitions the hidden states of a standard LSTM model into several groups. Each group is activated and updated at different time periods. Thus, MT-LSTM can model very long documents. MT-LSTM is reported to outperform a set of baselines, including the models based on LSTM and RNN, on TC.

RNNs are good at capturing the local structure of a word sequence, but face difficulties remembering long-range dependencies. In contrast, latent topic models are able to capture the global semantic structure of a document but do not account for word ordering. Dieng et al. [37] propose a TopicRNN model to integrate the merits of RNNs and latent topic models. It captures local (syntactic) dependencies using RNNs and global (semantic) dependencies using latent topics. TopicRNN is reported to outperform RNN baselines for sentiment analysis.

There are other interesting RNN-based models. Liu et al. [38] use multi-task learning to train RNNs to leverage labeled training data from multiple related tasks. Johnson and Rie [39] explore a text region embedding method using LSTM. Zhou et al. [40] integrate a Bidirectional-LSTM (Bi-LSTM) model with two-dimensional max-pooling to capture text features. Wang et al. [41] propose a bilateral multi-perspective matching model under the "matching-aggregation" framework. Wan et al. [42] explore semantic matching using multiple positional sentence representations generated by a bi-directional LSMT model.

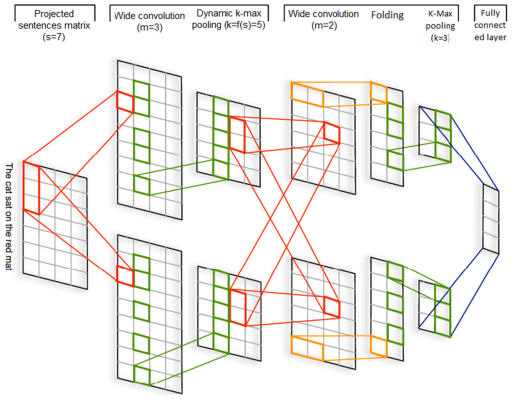

Fig. 4. The architecture of DCNN model [45].

It is worth noting that RNNs belong to a broad category of DNNs, known as *recursive neural networks*. A recursive neural network applies the same set of weights recursively over a structure input to produce a structured prediction or a vector representation over variable-size input. Whereas RNNs are recursive neural networks with a linear chain structure input, there are recursive neural networks that operate on hierarchical structures, such as parse trees of natural language sentences [43], combining child representations into parent representations. RNNs are the most popular recursive neural networks for TC, because they are effective and easy to use—they view text as a sequence of tokens without requiring additional structure labels such as parse trees.

2.3 CNN-based Models

RNNs are trained to recognize patterns across time, whereas CNNs learn to recognize patterns across space [44]. RNNs work well for the NLP tasks such as POS tagging or QA where the comprehension of long-range semantics is required, while CNNs work well where detecting local and position-invariant patterns is important. These patterns could be key phrases that express a particular sentiment like "I like" or a topic like "endangered species." Thus, CNNs have become one of the most popular model architectures for TC.

One of the first CNN-based models for TC is proposed by Kalchbrenner et al. [45]. The model uses dynamic k-max-pooling, and is called the **Dynamic CNN (DCNN)**. As illustrated in Figure 4, the first layer of DCNN constructs a sentence matrix using the embedding for each word in the sentence. Then a convolutional architecture that alternates wide convolutional layers with dynamic pooling layers given by dynamic k-max-pooling is used to generate a feature map over the sentence that is capable of explicitly capturing short and long-range relations of words and phrases. The pooling parameter k can be dynamically chosen depending on the sentence size and the level in the convolution hierarchy.

Later, Kim [46] proposes a much simpler CNN-based model than DCNN for TC. As shown in Figure 5, Kim's model uses only one layer of convolution on top of the word vectors obtained from an unsupervised neural language model i.e., word2vec. Kim also compares four different approaches to learning word embeddings: (1) CNN-rand, where all word embeddings are randomly initialized and then modified during training; (2) CNN-static, where the pre-trained word2vec embeddings are used and stay fixed during model training; (3) CNN-non-static, where the word2vec embeddings are fine-tuned during training for each task; and (4) CNN-multi-channel, where two sets of word embedding vectors are used, both are initialized using word2vec, with one updated

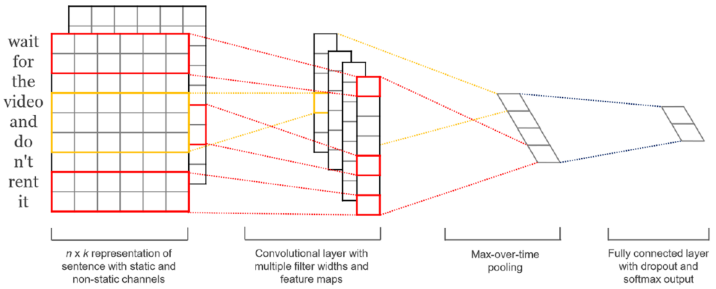

Fig. 5. The architecture of a sample CNN model for text classification. courtesy of Yoon Kim [46].

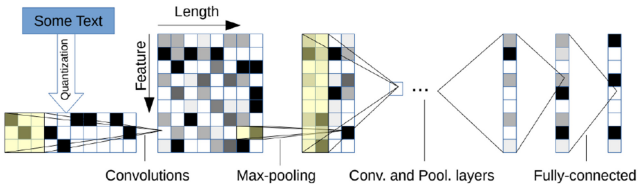

Fig. 6. The architecture of a character-level CNN model [50].

during model training while the other fixed. These CNN-based models are reported to improve upon the state of the art on sentiment analysis and question classification.

There have been efforts of improving the architectures of CNN-based models of References [45, 46]. Liu et al. [47] propose a new CNN-based model that makes two modifications to the architecture of Kim-CNN [46]. First, a dynamic max-pooling scheme is adopted to capture more fine-grained features from different regions of the document. Second, a hidden bottleneck layer is inserted between the pooling and output layers to learn compact document representations to reduce model size and boost model performance. In References [48, 49], instead of using pre-trained low-dimensional word vectors as input to CNNs, the authors directly apply CNNs to high-dimensional text data to learn the embeddings of small text regions for classification.

Character-level CNNs have also been explored for TC [50, 51]. One of the first such models is proposed by Zhang et al. [50]. As illustrated in Figure 6, the model takes as input the characters in a fixed-sized, encoded as one-hot vectors, passes them through a deep CNN model that consists of six convolutional layers with pooling operations and three fully connected layers. Prusa et al. [52] present an approach to encoding text using CNNs that greatly reduces memory consumption and training time required to learn character-level text representations. This approach scales well with alphabet size, allowing to preserve more information from the original text to enhance classification performance.

There are studies on investigating the impact of word embeddings and CNN architectures on model performance. Inspired by VGG [53] and ResNets [54], Conneau et al. [55] present a **Very Deep CNN (VDCNN)** model for text processing. It operates directly at the character level and uses only small convolutions and pooling operations. This study shows that the performance of VDCNN improves with the increase of the depth. Duque et al. [56] modify the structure of VDCNN to fit mobile platforms' constraints without much performance degradation. They are able to compress the model size by 10× to 20× with an accuracy loss between 0.4% to 1.3%. Le et al. [57] show that deep models indeed outperform shallow models when the text input is represented as a sequence of characters. However, a simple shallow-and-wide network outperforms deep models such as DenseNet [58] with word inputs. Guo et al. [59] study the impact of word embedding and propose

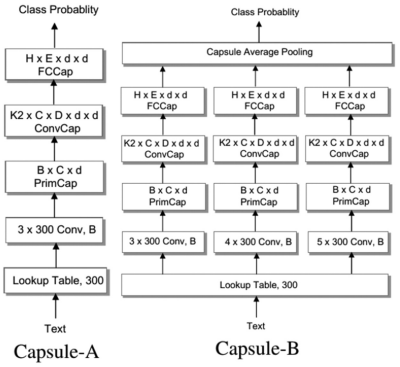

Fig. 7. CapsNet A and B for text classification [71].

to use weighted word embeddings via a multi-channel CNN model. Zhang et al. [60] examine the impact of different word embedding methods and pooling mechanisms, and find that using non-static word2vec and GloVe outperforms one-hot vectors, and that max-pooling consistently outperforms other pooling methods.

There are other interesting CNN-based models. Mou et al. [61] present a tree-based CNN to capture sentence-level semantics. Pang et al. [62] cast text matching as the image recognition task, and use multi-layer CNNs to identify salient n-gram patterns. Wang et al. [63] propose a CNN-based model that combines explicit and implicit representations of short text for TC. There is also a growing interest in applying CNNs to biomedical text classification [64–67].

2.4 Capsule Neural Networks

CNNs classify images or texts by using successive layers of convolutions and pooling. Although pooling operations identify salient features and reduce the computational complexity of convolution operations, they lose information regarding spatial relationships and are likely to mis-classify entities based on their orientation or proportion.

To address the problems of pooling, a new approach is proposed by Hinton et al., called **capsule networks (CapsNets)** [68, 69]. A capsule is a group of neurons whose activity vector represents different attributes of a specific type of entity such as an object or an object part. The vector's length represents the probability that the entity exists, and the orientation of the vector represents the attributes of the entity. Unlike max-pooling of CNNs, which selects some information and discards the rest, capsules "route" each capsule in the lower layer to its best parent capsule in the upper layer, using all the information available in the network up to the final layer for classification. Routing can be implemented using different algorithms, such as dynamic routing-by-agreement [69] or the EM algorithm [70].

Recently, capsule networks have been applied to TC, where capsules are adapted to represent a sentence or document as a vector. References [71–73] propose a TC model based on a variant of CapsNets. The model consists of four layers: (1) an n-gram convolutional layer, (2) a capsule layer, (3) a convolutional capsule layer, and (4) a fully connected capsule layer. The authors experiment three strategies to stabilize the dynamic routing process to alleviate the disturbance of the noise capsules that contain background information such as stopwords or the words that are unrelated to any document categories. They also explore two capsule architectures, Capsule-A and Capsule-B as in Figure 7. Capsule-A is similar to the CapsNet in Reference [69]. Capsule-B uses three parallel networks with filters with different window sizes in the n-gram convolutional layer to learn a more comprehensive text representation. CapsNet-B performs better in the experiments.

The CapsNet-based model proposed by Kim et al. [74] uses a similar architecture. The model consists of (1) an input layer that takes a document as a sequence of word embeddings; (2) a convolutional layer that generates feature maps and uses a gated-linear unit to retain spatial information; (3) a convolutional capsule layer to form global features by aggregating local features detected by the convolutional layer; and (4) a text capsule layer to predict class labels. The authors observe that objects can be more freely assembled in texts than in images. For example, a document's semantics can remain the same even if the order of some sentences is changed, unlike the the positions of the eyes and nose on a human face. Thus, they use a static routing schema, which consistently outperforms dynamic routing [69] for TC. Aly et al. [75] propose to use CapsNets for **Hierarchical Multilabel Classification (HMC)**, arguing that the CapsNet's capability of encoding child-parent relations makes it a better solution than traditional methods to the HMC task where documents are assigned one or multiple class labels organized in a hierarchical structure. Their model's architecture is similar to the ones in References [71, 72, 74].

Ren et al. [76] propose another variant of CapsNets using a compositional coding mechanism between capsules and a new routing algorithm based on k-means clustering. First, the word embeddings are formed using all codeword vectors in codebooks. Then features captured by the lower-level capsules are aggregated in high-level capsules via k-means routing.

2.5 Models with Attention Mechanism

Attention is motivated by how we pay visual attention to different regions of an image or correlate words in one sentence. Attention becomes an increasingly popular concept and useful tool in developing DL models for NLP [77, 78]. In a nutshell, attention in language models can be interpreted as a vector of importance weights. To predict a word in a sentence, we estimate using the attention vector how strongly it is correlated with, or "attends to," other words and take the sum of their values weighted by the attention vector as the approximation of the target.

This section reviews some of the most prominent attention models that create new state of the arts on TC tasks, when they are published.

Yang et al. [79] propose a hierarchical attention network for text classification. This model has two distinctive characteristics: (1) a hierarchical structure that mirrors the hierarchical structure of documents, and (2) two levels of attention mechanisms applied at the word and sentence-level, enabling it to attend differentially to more and less important content when constructing the document representation. This model outperforms previous methods by a substantial margin on six TC tasks. Zhou et al. [80] extend the hierarchical attention model to cross-lingual sentiment classification. In each language, a LSTM network is used to model the documents. Then, classification is achieved by using a hierarchical attention mechanism, where the sentence-level attention model learns which sentences of a document are more important for determining the overall sentiment. while the word-level attention model learns which words in each sentence are decisive.

Shen et al. [81] present a directional self-attention network for RNN/CNN-free language understanding, where the attention between elements from input sequence(s) is directional and multi-dimensional. A light-weight neural net is used to learn sentence embedding, solely based on the proposed attention without any RNN/CNN structure. Liu et al. [82] present a LSTM model with inner-attention for NLI. This model uses a two-stage process to encode a sentence. First, average pooling is used over word-level Bi-LSTMs to generate a first stage sentence representation. Second, attention mechanism is employed to replace average pooling on the same sentence for better representations. The sentence's first-stage representation is used to attend words appeared in itself.

Attention models are widely applied to pair-wise ranking or text matching tasks too. Santos et al. [83] propose a two-way attention mechanism, known as **Attentive Pooling (AP)**, for

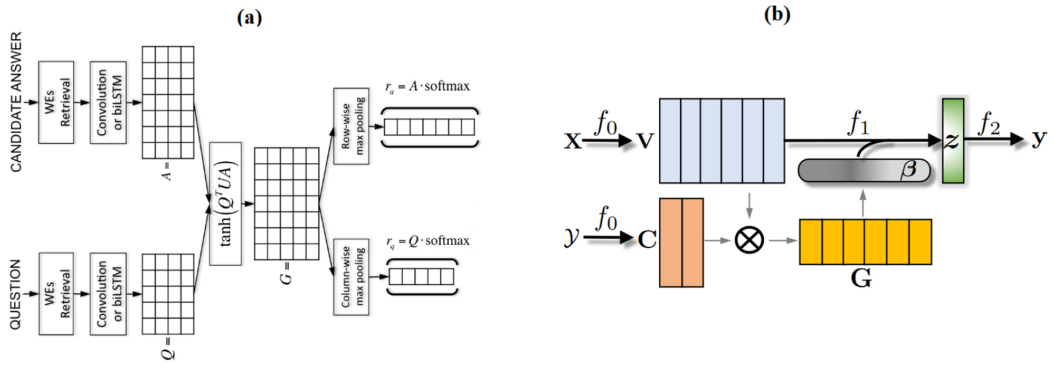

Fig. 8. (a) The architecture of attentive pooling networks [83]. (b) The architecture of label-text matching model [84].

pair-wise ranking. AP enables the pooling layer to be aware of the current input pair (e.g., a question-answer pair), in a way that information from the two input items can directly influence the computation of each other's representations. In addition to learning the representations of the input pair, AP jointly learns a similarity measure over projected segments of the pair, and subsequently derives the corresponding attention vector for each input to guide the pooling. AP is a general framework independent of the underlying representation learning, and can be applied to both CNNs and RNNs, as illustrated in Figure 8(a). Wang et al. [84] view TC as a label-word matching problem: each label is embedded in the same space with the word vector. The authors introduce an attention framework that measures the compatibility of embeddings between text sequences and labels via cosine similarity, as shown in Figure 8(b).

Kim et al. [85] propose a semantic sentence matching approach using a densely-connected recurrent and co-attentive network. Similar to DenseNet [58], each layer of this model uses concatenated information of attentive features as well as hidden features of all the preceding recurrent layers. It enables preserving the original and the co-attentive feature information from the bottom-most word embedding layer to the uppermost recurrent layer. Yin et al. [86] present another attention-based CNN model for sentence pair matching. They examine three attention schemes for integrating mutual influence between sentences into CNNs, so that the representation of each sentence takes into consideration its paired sentence. These interdependent sentence pair representations are shown to be more powerful than isolated sentence representations, as validated on multiple classification tasks including answer selection, paraphrase identification, and textual entailment. Tan et al. [87] employ multiple attention functions to match sentence pairs under the matching-aggregation framework. Yang et al. [88] introduce an attention-based neural matching model for ranking short answer texts. They adopt value-shared weighting scheme instead of position-shared weighting scheme for combining different matching signals and incorporated question term importance learning using question attention network. This model achieves promising results on the TREC QA dataset.

There are other interesting attention models. Lin et al. [89] used self-attention to extract interpretable sentence embeddings. Wang et al. [90] proposed a densely connected CNN with multi-scale feature attention to produce variable n-gram features. Yamada and Shindo [91] used neural attentive bag-of-entities models to perform TC using entities in a knowledge base. Parikh et al. [92] used attention to decompose a problem into sub-problems that can be solved separately. Chen et al. [93] explored generalized pooling methods to enhance sentence embedding, and proposed

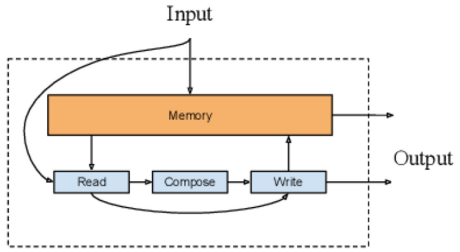

Fig. 9. The architecture of NSE [95].

a vector-based multi-head attention model. Basiri et al. [94] proposed an attention-based bidirectional CNN-RNN deep model for sentiment analysis.

2.6 Memory-augmented Networks

While the hidden vectors stored by an attention model during encoding can be viewed as entries of the model's *internal memory*, memory-augmented networks combine neural networks with a form of *external memory*, which the model can read from and write to.

Munkhdalai and Yu [95] present a memory-augmented neural network, called **Neural Semantic Encoder (NSE)**, for TC and QA. NSE is equipped with a variable sized encoding memory that evolves over time and maintains the understanding of input sequences through read, compose and write operations, as shown in Figure 9.

Weston et al. [96] design a memory network for a synthetic QA task, where a series of statements (memory entries) are provided to the model as supporting facts to the question. The model learns to retrieve one entry at a time from memory based on the question and previously retrieved memory. Sukhbaatar et al. [97] extend this work and propose end-to-end memory networks, where memory entries are retrieved in a soft manner with attention mechanism, thus enabling end-to-end training. They show that with multiple rounds (hops), the model is able to retrieve and reason about several supporting facts to answer a specific question.

Kumar et al. [98] propose a **Dynamic Memory Metwork (DMN)**, which processes input sequences and questions, forms episodic memories, and generates relevant answers. Questions trigger an iterative attention process, which allows the model to condition its attention on the inputs and the result of previous iterations. These results are then reasoned over in a hierarchical recurrent sequence model to generate answers. The DMN is trained end-to-end, and obtains state of the art results on QA and POS tagging. Xiong et al. [99] present a detailed analysis of the DMN, and improve its memory and input modules.

2.7 Graph Neural Networks

Although natural language texts exhibit a sequential order, they also contain internal graph structures, such as syntactic and semantic parse trees, which define the syntactic and semantic relations among words in sentences.

One of the earliest graph-based models developed for NLP is TextRank [100]. The authors propose to represent a natural language text as a graph $G(V, E)$, where V denotes a set of nodes and E a set of edges among the nodes. Depending on the applications at hand, nodes can represent text units of various types, e.g., words, collocations, entire sentences, and so on. Similarly, edges can be used to represent different types of relations between any nodes, e.g., lexical or semantic relations, contextual overlap, and so on.

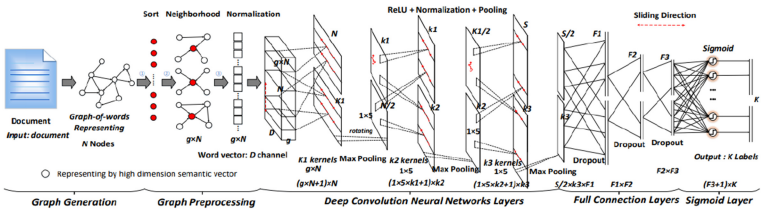

Fig. 10. The architecture of GNN used by Peng et al. [105].

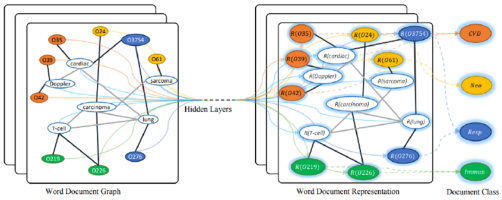

Fig. 11. The architecture of GCNN [107].

Modern **Graph Neural Networks (GNNs)** are developed by extending DL approaches for graph data, such as the text graphs used by TextRank. Deep neural networks, such as CNNs, RNNs, and autoencoders, have been generalized over the past few years to handle the complexity of graph data [101]. For example, a 2D convolution of CNNs for image processing is generalized to perform graph convolutions by taking the weighted average of a node's neighborhood information. Among various types of GNNs, convolutional GNNs, such as **Graph Convolutional Networks (GCNs)** [102] and their variants, are the most popular ones, because they are effective and convenient to compose with other neural networks, and have achieved state of the art results in many applications. GCNs are an efficient variant of CNNs on graphs. GCNs stack layers of learned first-order spectral filters followed by a nonlinear activation function to learn graph representations.

A typical application of GNNs in NLP is TC. GNNs utilize the inter-relations of documents or words to infer document labels [102–104]. In what follows, we review some variants of GCNs that are developed for TC.

Peng et al. [105] propose a graph-CNN-based DL model to first convert text to graph-of-words, and then use graph convolution operations to convolve the word graph, as shown in Figure 10. They show through experiments that the graph-of-words representation of texts has the advantage of capturing non-consecutive and long-distance semantics, and CNN models have the advantage of learning different level of semantics.

In Reference [106], Peng et al. propose a TC model based on hierarchical taxonomy-aware and attentional graph capsule CNNs. One unique feature of the model is the use of the hierarchical relations among the class labels, which in previous methods are considered independent. Specifically, to leverage such relations, the authors develop a hierarchical taxonomy embedding method to learn their representations, and define a novel weighted margin loss by incorporating the label representation similarity.

Yao et al. [107] use a similar **Graph CNN (GCNN)** model for TC. They build a single text graph for a corpus based on word co-occurrence and document word relations, then learn a **Text Graph Convolutional Network (Text GCN)** for the corpus, as shown in Figure 11. The Text GCN is initialized with one-hot representation for word and document, and then jointly learns the embeddings for both words and documents, as supervised by the known class labels for documents.

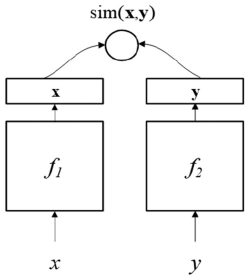

Fig. 12. The architecture of a DSSM, illustrated in Reference [115].

Building GNNs for a large-scale text corpus is costly. There have been works on reducing the modeling cost by either reducing the model complexity or changing the model training strategy. An example of the former is the **Simple Graph Convolution (SGC)** model proposed in Reference [108], where a deep convolutional GNN is simplified by repeatedly removing the nonlinearities between consecutive layers and collapsing the resulting functions (weight matrices) into a single linear transformation. An example of the latter is the text-level GNN [109]. Instead of building a graph for an entire text corpus, a text-level GNN produces one graph for each text chunk defined by a sliding window on the text corpus to reduce the memory consumption during training. Some of the other promising GNN-based works include, GraphSage [103], and contextualized non-local neural networks [110].

2.8 Siamese Neural Networks

Siamese neural networks (S2Nets) [111, 112] and their DNN variants, known as **Deep Structured Semantic Models (DSSMs)** [113, 114], are designed for text matching. The task is fundamental to many NLP applications, such as query-document ranking and answer selection in extractive QA. These tasks can be viewed as special cases of TC. For example, in question-document ranking, we want to classify a document as relevant or irrelevant to a given query.

As illustrated in Figure 12, a DSSM (or a S2Net) consists of a pair of DNNs, f_1 and f_2, which map inputs x and y into corresponding vectors in a common low-dimensional semantic space [115]. Then the similarity of x and y is measured by the cosine distance of the two vectors. While S2Nets assume that f_1 and f_2 share the same architecture and even the same parameters, in DSSMs, f_1 and f_2 can be of different architectures depending on x and y. For example, to compute the similarity of an image-text pair, f_1 can be a deep CNN and f_2 an RNN or MLP. These models can be applied to a wide range of NLP tasks depending on the definition of (x, y). For example, (x, y) could be a query-document pair for query-document ranking [114, 116], or a question-answer pair in QA [117, 118].

The model parameters θ are often optimized using a pair-wise rank loss. Take document ranking as an example. Consider a query x and two candidate documents y^+ and y^-, where y^+ is relevant to x and y^- is not. Let $\text{sim}_\theta(x, y)$ be the cosine similarity of x and y in the semantic space parameterized by θ. The training objective is to minimize the margin-based loss as

$$L(\theta) = \left[\gamma + \text{sim}_\theta(x, y^-) - \text{sim}_\theta(x, y^+)\right]_+, \tag{1}$$

where $[x]_+ := \max(0, x)$ and γ is the margin hyperparameter.

Since texts exhibit a sequential order, it is natural to implement f_1 and f_2 using RNNs or LSTMs to measure the semantic similarity between texts. Figure 13 shows the architecture of the siamese model proposed in Reference [119], where the two networks use the same LSTM model. Neculoiu et al. [120] present a similar model that uses character-level Bi-LSTMs for f_1 and f_2, and the

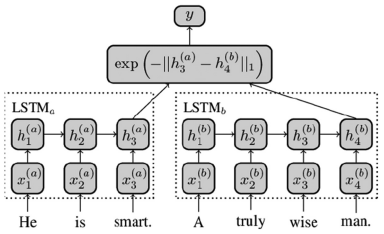

Fig. 13. The architecture of the Siamese model proposed by Mueller et al. [119].

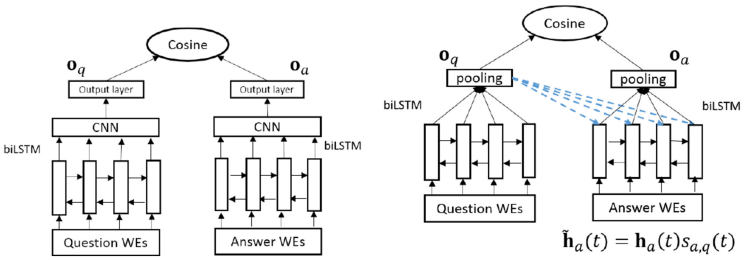

Fig. 14. The architectures of the Siamese models studied in Reference [118].

cosine function to calculate the similarity. Liu et al. [121] model the interaction of a sentence pair with two coupled-LSTMs. In addition to RNNs, BOW models and CNNs are also used in S2Nets to represent sentences. For example, He et al. [122] propose a S2Net that uses CNNs to model multi-perspective sentence similarity. Renter et al. [123] propose a Siamese CBOW model, which forms a sentence vector representation by averaging the word embeddings of the sentence, and calculates the sentence similarity as cosine similarity between sentence vectors. As BERT becomes the new state of the art sentence embedding model, there have been attempts to building BERT-based S2Nets, such as SBERT [124] and TwinBERT [125].

S2Nets and DSSMs have been widely used for QA. Das et al. [117] propose a Siamese CNN for QA (SCQA) to measure the semantic similarity between a question and its (candidate) answers. To reduce the computational complexity, SCQA uses character-level representations of question-answer pairs. The parameters of SCQA is trained to maximize the semantic similarities between a question and its relevant answers, as Equation (1), where x is a question and y its candidate answer. Tan et al. [118] present a series of siamese neural networks for answer selection. As shown in Figure 14, these are hybrid models that process text using convolutional, recurrent, and attention neural networks. Other siamese neural networks developed for QA include LSTM-based models for non-factoid answer selection [126], Hyperbolic representation learning [127], and QA using a deep similarity neural network [128].

2.9 Hybrid Models

Many Hybrid models have been developed to combine LSTM and CNN architectures to capture local and global features of sentences and documents. Zhu et al. [129] propose a Convolutional LSTM (C-LSTM) network. As illustrated in Figure 15(a), C-LSTM utilizes a CNN to extract a sequence of higher-level phrase (n-gram) representations, which are fed to a LSTM network to obtain the sentence representation. Similarly, Zhang et al. [130] propose a Dependency Sensitive CNN (DSCNN) for document modeling. As illustrated in Figure 15(b), the DSCNN is a hierarchical model, where

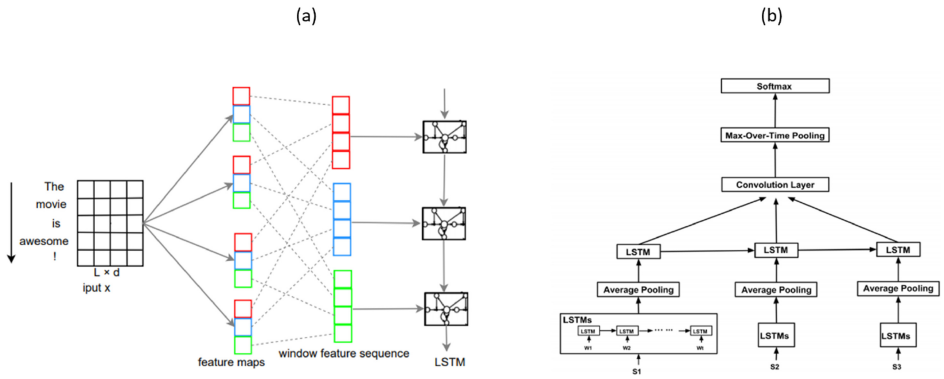

Fig. 15. (a) The architecture of C-LSTM [129]. (b) The architecture of DSCNN for document modeling [130].

LSTM learns the sentence vectors that are fed to the convolution and max-pooling layers to generate the document representation.

Chen et al. [131] perform multi-label TC through a CNN-RNN model that is able to capture both global and local textual semantics and, hence, to model high-order label correlations while having a tractable computational complexity. Tang et al. [132] use a CNN to learn sentence representations, and a gated RNN to learn a document representation that encodes the intrinsic relations between sentences. Xiao et al. [133] view a document as a sequence of characters, instead of words, and propose to use both character-based convolution and recurrent layers for document encoding. This model achieved comparable performances with much less parameters, compared with word-level models. The Recurrent CNN [134] applies a recurrent structure to capture long-range contextual dependence for learning word representations. To reduce the noise, max-pooling is employed to automatically select only the salient words that are crucial to the text classification task.

Chen et al. [135] propose a divide-and-conquer approach to sentiment analysis via sentence type classification, motivated by the observation that different types of sentences express sentiment in very different ways. The authors first apply a Bi-LSTM model to classify opinionated sentences into three types. Each group of sentences is then fed to a one-dimensional CNN separately for sentiment classification.

In Reference [136], Kowsari et al. propose a Hierarchical Deep Learning approach for Text classification (HDLTex). HDLTex employs stacks of hybrid DL model architectures, including MLP, RNN, and CNN, to provide specialized understanding at each level of the document hierarchy.

Liu [137] propose a robust **Stochastic Answer Network (SAN)** for multi-step reasoning in machine reading comprehension. SAN combines neural networks of different types, including memory networks, Transforms, Bi-LSTM, attention and CNN. The Bi-LSTM component obtains the context representations for questions and passages. Its attention mechanism derives a question-aware passage representation. Then, another LSTM is used to generate a working memory for the passage. Finally, a Gated Recurrent Unit–based answer module outputs predictions.

Several studies have been focused on combining highway networks with RNNs and CNNs. In typical multi-layer neural networks, information flows layer by layer. Gradient-based training of a DNN becomes more difficult with increasing depth. Highway networks [138] are designed to ease training of very deep neural networks. They allow unimpeded information flow across several layers on *information highways*, similar to the shortcut connections in ResNet [139]. Kim et al. [140] employ a highway network with CNN and LSTM over characters for language modeling. As illustrated in Figure 16, the first layer performs a lookup of character embeddings, then convolution

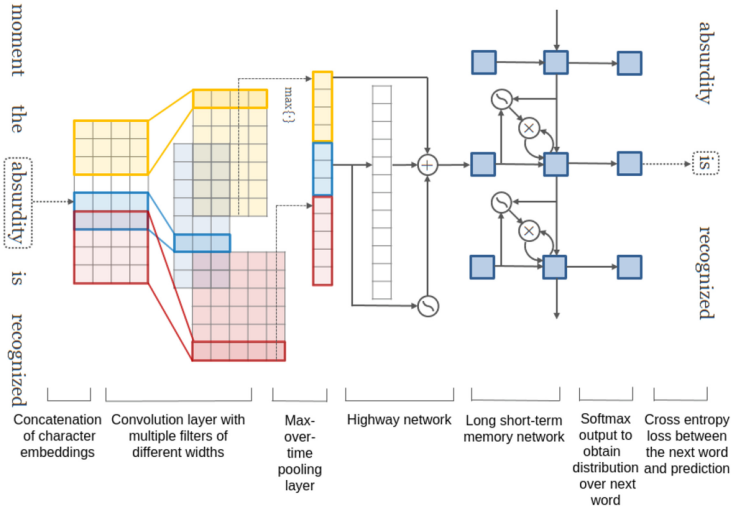

Fig. 16. The architecture of the highway network with CNN and LSTM [140].

and max-pooling operations are applied to obtain a fixed-dimensional representation of the word, which is given to the highway network. The highway network's output is used as the input to a multi-layer LSTM. Finally, an affine transformation followed by a softmax is applied to the hidden representation of the LSTM to obtain the distribution over the next word. Other highway-based hybrid models include recurrent highway networks [141], and RNN with highway [142].

2.10 Transformers and Pre-trained Language Models

One of the computational bottlenecks suffered by RNNs is the sequential processing of text. Although CNNs are less sequential than RNNs, the computational cost to capture relationships between words in a sentence also grows with the increasing length of the sentence, similar to RNNs. Transformers [5] overcome this limitation by applying self-attention to compute in parallel for every word in a sentence or document an "attention score" to model the influence each word has on another.[2] Due to this feature, Transformers allow for much more parallelization than CNNs and RNNs, which makes it possible to efficiently train very big models on large amounts of data on GPUs.

Since 2018, we have seen the rise of a set of large-scale Transformer-based Pre-trained Language Models (PLMs). Compared to earlier contextualized embedding models based on CNNs [143] or LSTMs [4], Transformer-based PLMs use much deeper network architectures (e.g., 48-layer Transformers [144]), and are *pre-trained* on much larger amounts of text corpora to learn contextual text representations by predicting words conditioned on their context. These PLMs are *fine-tuned* using task-specific labels, and have created new state of the art in many downstream NLP tasks, including TC. Although pre-training is unsupervised (or self-supervised), fine-tuning is supervised learning. A recent survey of Qiu et al. [145] categories popular PLMs by their representation types, model architectures, pre-training tasks, and downstream tasks.

PLMs can be grouped into two categories, autoregressive and autoencoding PLMs. One of the earliest autoregressive PLMs is OpenGPT [6, 144], a unidirectional model that predicts a text sequence word by word from left to right (or right to left), with each word prediction depending

[2]Strictly speaking, Transformer is an instance of hybrid models (Section 2.9), since each Transformer layer is a composite structure consisting of a feed-forward layer and a multi-head attention layer.

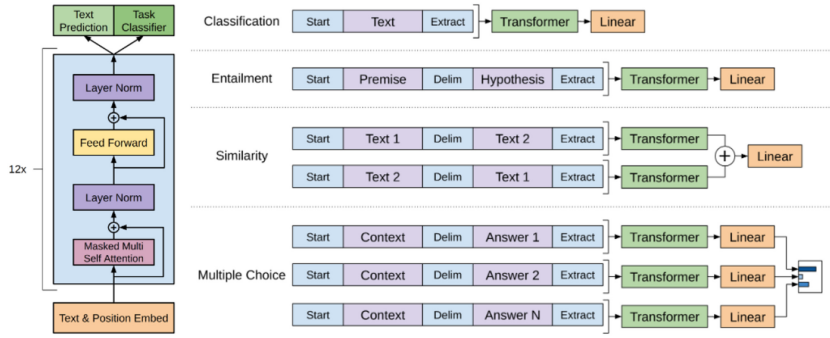

Fig. 17. The architecture of OpenGPT-1 [144].

on previous predictions. Figure 17 shows the architecture of OpenGPT. It consists of 12 layers of Transformer blocks, each consisting of a masked multi-head attention module, followed by a layer normalization and a position-wise feed forward layer. OpenGPT can be adapted to downstream tasks such as TC by adding task-specific linear classifiers and fine-tuning using task-specific labels.

One of the most widely used autoencoding PLMs is BERT [7]. Unlike OpenGPT, which predicts words based on previous predictions, BERT is trained using the **masked language modeling (MLM)** task that randomly masks some tokens in a text sequence, and then independently recovers the masked tokens by conditioning on the encoding vectors obtained by a bidirectional Transformer. There have been numerous works on improving BERT. RoBERTa [146] is more robust than BERT, and is trained using much more training data. ALBERT [147] lowers the memory consumption and increases the training speed of BERT. DistillBERT [148] utilizes knowledge distillation during pre-training to reduce the size of BERT by 40% while retaining 99% of its original capabilities and making the inference 60% faster. SpanBERT [149] extends BERT to better represent and predict text spans. Electra [150] uses a more sample-efficient pre-training task than MLM, called replaced token detection. Instead of masking the input, it corrupts it by replacing some tokens with plausible alternatives sampled from a small generator network. ERNIE [151, 152] incorporates domain knowledge from external knowledge bases, such as named entities, for model pre-training. ALUM [14] introduces an adversarial loss for model pretraining that improves the model's generalization to new tasks and robustness to adversarial attacks. BERT and its variants have been fine-tuned for various NLP tasks, including QA [153], TC [154], and NLI [23, 155].

There have been attempts to combine the strengths of autoregressive and autoencoding PLMs. XLNet [156] integrates the idea of autoregressive models like OpenGPT and bi-directional context modeling of BERT. XLNet makes use of a *permutation operation* during pre-training that allows context to include tokens from both left and right, making it a generalized order-aware autoregressive language model. The permutation is achieved by using a special attention mask in Transformers. XLNet also introduces a two-stream self-attention schema to allow position-aware word prediction. This is motivated by the observation that word distributions vary greatly depending on word positions. For example, the beginning of a sentence has a considerably different distribution from other positions in the sentence. As show in Figure 18, to predict the word token in position 1 in a permutation 3-2-4-1, a content stream is formed by including the positional embeddings and token embeddings of all previous words (3, 2, 4), then a query stream is formed by including the content stream and the positional embedding of the word to be predicted (word in position 1), and finally the model makes the prediction based on information from the query stream.

As mentioned earlier, OpenGPT uses a left-to-right Transformer to learn text representation for natural language generation, while BERT uses a bidirectional transformer for natural

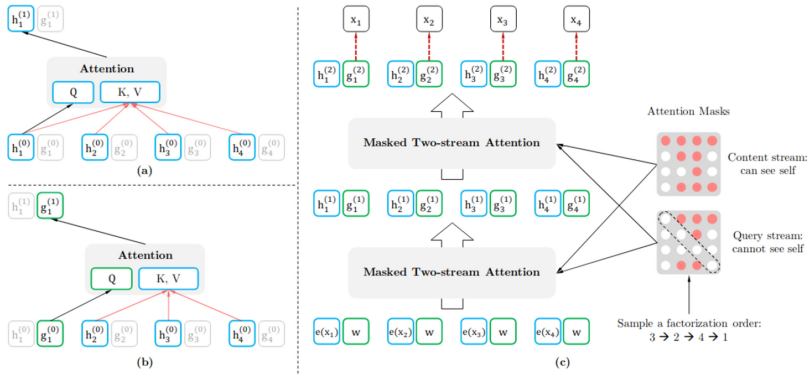

Fig. 18. The architecture of XLNet [156]: (a) Content stream attention, (b) Query stream attention, (c) Overview of the permutation language modeling training with two-stream attention.

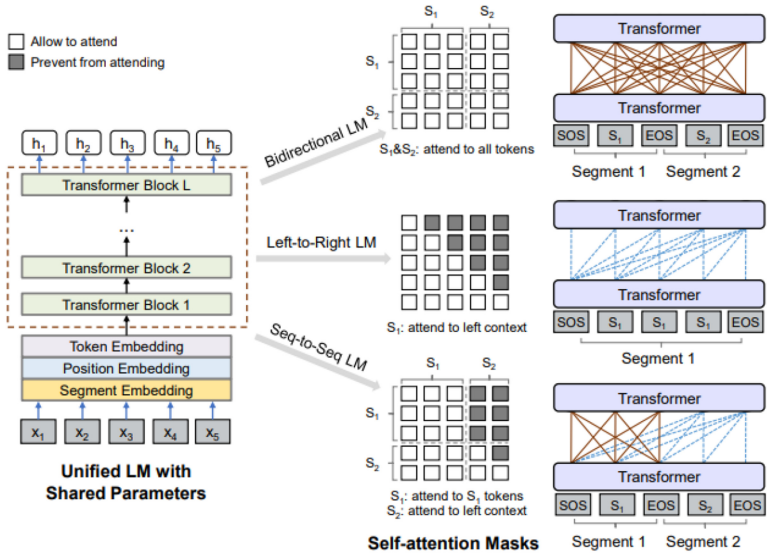

Fig. 19. Overview of UniLM pre-training [157]. The model parameters are shared across the language modeling objectives i.e., bidirectional, unidirectional, and sequence-to-sequence language modeling. Different self-attention masks are used to control the access to context for each word token.

language understanding. The **Unified language Model (UniLM)** [157] is designed to tackle both natural language understanding and generation tasks. UniLM is pre-trained using three types of language modeling tasks: unidirectional, bidirectional, and sequence-to-sequence prediction. The unified modeling is achieved by employing a shared Transformer network and utilizing specific self-attention masks to control what context the prediction conditions on, as shown in Figure 19. The second version of UniLM [158] is reported to achieve new state of the art on a wide range of natural language understanding and generation tasks, significantly outperforming previous PLMs, including OpenGPT-2, XLNet, BERT and its variants.

Raffel et al. [159] present a unified Transformer-based framework that converts many NLP problems into a text-to-text format. They also conduct a systematic study to compare pre-training

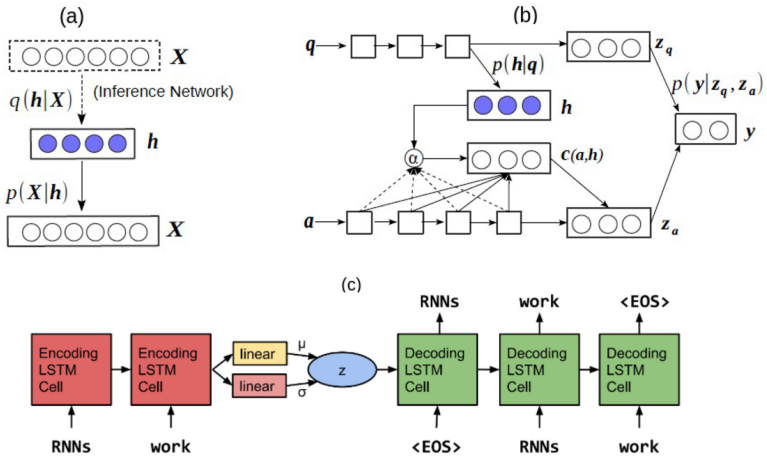

Fig. 20. (a) The neural variational document model for document modeling [166]. (b) The neural answer selection model for QA [166]. (c) The RNN-based variational autoencoder language model [167].

objectives, architectures, unlabeled datasets, fine-tuning approaches, and other factors on dozens of language understanding tasks.

2.11 Beyond Supervised Learning

Unsupervised Learning using Autoencoders. Similar to word embeddings, distributed representations for sentences can also be learned in an unsupervised fashion. by optimizing some auxiliary objectives, such as the reconstruction loss of an autoencoder [160]. The result of such unsupervised learning are sentence encoders, which can map sentences with similar semantic and syntactic properties to similar fixed-size vector representations. The Transformer-based PLMs described in Section 2.10 are also unsupervised models that can be used as sentence encoders. This section discusses unsupervised models based on auto-encoders and their variants.

Kiros et al. [161] propose the Skip-Thought model for unsupervised learning of a generic, sentence encoder. An encoder-decoder model is trained to reconstruct the surrounding sentences of an encoded sentence. Dai and Le [162] investigate the use of a sequence autoencoder, which reads the input sequence into a vector and predicts the input again, for sentence encoding. They show that pre-training sentence encoders on a large unsupervised corpus yields better accuracy than only pre-training word embeddings. Zhang et al. [163] propose a mean-max attention autoencoder, which uses the multi-head self-attention mechanism to reconstruct the input sequence. A mean-max strategy is used in encoding, where both mean and max pooling operations over the hidden vectors are applied to capture diverse information of the input.

While autoencoders learn a compressed representation of input, **Variational AutoEncoders (VAEs)** [164, 165] learn a distribution representing the data, and can be viewed as a regularized version of the autoencoder [26]. Since a VAE learns to model the data, we can easily sample from the distribution to generate new samples (e.g., new sentences). Miao et al. [166] extend the VAE framework to text, and propose a **Neural Variational Document Model (NVDM)** for document modeling and a **Neural Answer Selection Model (NASM)** for QA. As shown in Figure 20(a), the NVDM uses an MLP encoder to map a document to a continuous semantic representation. As shown in Figure 20(b), the NASM uses LSTM and a latent stochastic attention mechanism to model the semantics of question-answer pairs and predicts their relatedness. The attention model focuses on the phrases of an answer that are strongly connected to the question semantics and is

modeled by a latent distribution, allowing the model to deal with the ambiguity inherent in the task. Bowman et al. [167] propose an RNN-based VAE language model, as shown in Figure 20(c). This model incorporates distributed latent representations of entire sentences, allowing to explicitly model holistic properties of sentences such as style, topic, and high-level syntactic features. Gururangan et al. [168] pre-train a document model as a VAE on in-domain, unlabeled data and use its internal states as features for text classification. In general, data augmentation using VAE or other models [169, 170] is widely used for semi-supervised or weakly supervised TC.

Adversarial Training. Adversarial training [171] is a regularization method for improving the generalization of a classifier. It does so by improving model's robustness to adversarial examples, which are created by making small perturbations to the input. Adversarial training requires the use of labels, and is applied to supervised learning. Virtual adversarial training [172] extend adversarial training to semi-supervised learning. This is done by regularizing a model so that given an example, the model produces the same output distribution as it produces on an adversarial perturbation of that example. Miyato et al. [173] extend adversarial and virtual adversarial training to supervised and semi-supervised TC tasks by applying perturbations to the word embeddings in an RNN rather than the original input itself. Sachel et al. [174] study LSTM models for semi-supervised TC. They find that using a mixed objective function that combines cross-entropy, adversarial, and virtual adversarial losses for both labeled and unlabeled data, leads to a significant improvement over supervised learning approaches. Liu et al. [175] extend adversarial training to the multi-task learning framework for TC [36], aiming to alleviate the task-independent (shared) and task-dependent (private) latent feature spaces from interfering with each other.

Reinforcement Learning. Reinforcement learning (RL) [176] is a method of training an agent to perform discrete actions according to a policy, which is trained to maximize a reward. Shen et al. [177] use a hard attention model to select a subset of critical word tokens of an input sequence for TC. The hard attention model can be viewed as an agent that takes actions of whether to select a token or not. After going through the entire text sequence, it receives a classification loss, which can be used as the reward to train the agent. Liu et al. [178] propose a neural agent that models TC as a sequential decision process. Inspired by the cognitive process of human text reading, the agent scans a piece of text sequentially and makes classification decisions at the time it wishes. Both the classification result and when to make the classification are part of the decision process, controlled by a policy trained with RL. Shen et al. [179] present a multi-step **Reasoning Network (ReasoNet)** for machine reading comprehension. ReasoNets tasks multiple steps to reason over the relation among queries, documents, and answers. Instead of using a fixed number of steps during inference, ReasoNets introduce a termination state to relax this constraint on the reasoning steps. With the use of RL, ReasoNets can dynamically determine whether to continue the comprehension process after digesting intermediate results, or to terminate reading when it concludes that existing information is adequate to produce an answer. Li et al. [180] combine RL, GANs, and RNNs to build a new model, termed **Category Sentence Generative Adversarial Network (CS-GAN)**, which is able to generate category sentences that enlarge the original dataset and to improve its generalization capability during supervised training. Zhang et al. [181] propose a RL-based method of learning structured representations for text classification. They propose two LSTM-based models. The first one selects only important, task-relevant words in the input text. The other one discovers phrase structures of sentences. Structure discovery using these two models is formulated as a sequential decision process guided by a policy network, which decides at each step which model to use, as illustrated in Figure 21. The policy network is optimized using policy gradient.

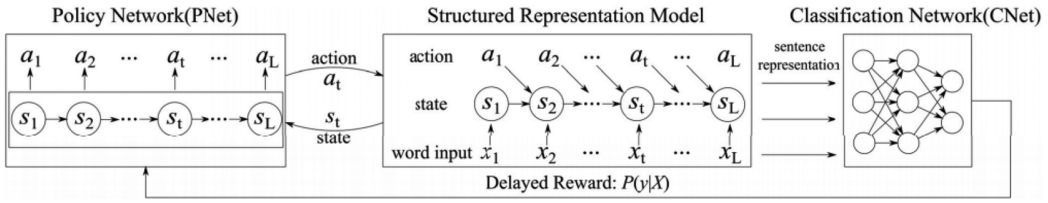

Fig. 21. The RL-based method of learning structured representations for text classification [181]. The policy network samples an action at each state. The structured representation model updates the state and outputs the final sentence representation to the classification network at the end of the episode. The text classification loss is used as a (negative) reward to train the policy.

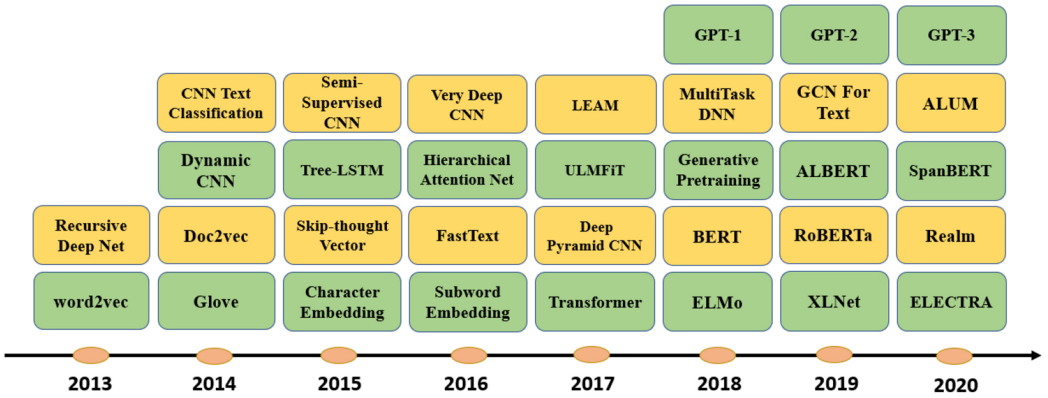

Fig. 22. Some of the most prominent deep learning models for text embedding and classification published from 2013 to 2020.

As a summary of this section, Figure 22 illustrates the timeline of some of the most popular DL-based models for TC since 2013.

3 HOW TO CHOOSE THE BEST NEURAL NETWORK MODEL FOR MY TASK

The answer to "what the best neural network architecture is for TC?" varies greatly depending on the nature of the target task and domain, the availability of in-domain labels, the latency and capacity constraints of the applications, and so on. Although it has no doubt that developing a text classifier is a trial-and-error process, by analyzing recent results on public benchmarks (e.g., GLUE [22]), we propose the following recipe to make the process easier. The recipe consists of five steps:

(1) **PLM Selection.** As will be shown in Section 5, using PLMs leads to significant improvements across all popular text classification tasks, and autoencoding PLMs (e.g., BERT or RoBERTa) often work better than autoregressive PLMs (e.g., OpenAI GPT). Hugging Face[3] maintains a rich repository of PLMs developed for various tasks and settings.
(2) **Domain adaptation.** Most PLMs are trained on general-domain text corpora (e.g., Web). If the target domain is dramatically different from general domain, then we might consider adapting the PLM using in-domain data by continual pre-training the selected

[3]https://huggingface.co/.

general-domain PLM. For domains with abundant unlabeled text, such as biomedicine, pretraining language models from scratch might also be a good choice [182].

(3) **Task-specific model design.** Given input text, the PLM produces a sequence of vectors in the contextual representation. Then, one or more task-specific layers are added on the top to generate the final output for the target task. The choice of the architecture of task-specific layers depends on the nature of the task, e.g., the linguistic structure of text needs to be captured. As described in Section 2, feed-forward neural networks view text as a bag of words, RNNs can capture word orders, CNNs are good at recognizing patterns such as key phrases, attention mechanisms are effective to identify correlated words in text, Siamese NNs are used for text matching tasks, and GNNs can be a good choice if graph structures of natural language (e.g., parse trees) are useful for the target task.

(4) **Task-specific fine-tuning.** Depending on the availability of in-domain labels, the task-specific layers can be either trained alone with the PLM fixed or trained together with the PLM. If multiple similar text classifiers need to be built (e.g., news classifiers for different domains), then multi-task fine-tuning [23] is a good choice to leverage labeled data of similar domains.

(5) **Model compression.** PLMs are expensive to serve. They often need to be compressed via e.g., knowledge distillation [183, 184] to meet the latency and capacity constraints in real-world applications.

4 TEXT CLASSIFICATION DATASETS

This section describes the datasets that are widely used for TC research. We group these datasets, based on their main target applications, into such categories as sentiment analysis, news categorization, topic classification, QA, and NLI.

4.1 Sentiment Analysis Datasets

Yelp. Yelp [185] dataset contains the data for two sentiment classification tasks. One is to detect fine-grained sentiment labels and is called Yelp-5. The other predicts the negative and positive sentiments, and is known as Yelp Review Polarity or Yelp-2. Yelp-5 has 650,000 training samples and 50,000 test samples for each class, and Yelp-2 includes 560,000 training samples and 38,000 test samples for negative and positive classes.

IMDb. The IMDB dataset [186] is developed for the task of binary sentiment classification of movie reviews. IMDB consists of equal number of positive and negative reviews. It is evenly divided between training and test sets with 25,000 reviews for each.

Movie Review. The **Movie Review (MR)** dataset [187] is a collection of movie reviews developed for the task of detecting the sentiment associated with a particular review and determining whether it is negative or positive. It includes 10,662 sentences with even numbers of negative and positive samples. Tenfold cross validation with random split is usually used for testing on this dataset.

SST. The **Stanford Sentiment Treebank (SST)** dataset [43] is an extended version of MR. Two versions are available, one with fine-grained labels (five-class) and the other binary labels, referred to as SST-1 and SST-2, respectively. SST-1 consists of 11,855 movie reviews, which are divided into 8,544 training samples, 1,101 development samples, and 2,210 test samples. SST-2 is partitioned into three sets with the sizes of 6,920, 872 and 1,821 as training, development and test sets, respectively.

MPQA. The **Multi-Perspective Question Answering (MPQA)** dataset [188] is an opinion corpus with two class labels. MPQA consists of 10,606 sentences extracted from news articles related to a wide variety of news sources. This is an imbalanced dataset with 3,311 positive documents and 7,293 negative documents.

Amazon. This is a popular corpus of product reviews collected from the Amazon website [189]. It contains labels for both binary classification and multi-class (5-class) classification. The Amazon binary classification dataset consists of 3,600,000 and 400,000 reviews for training and test, respectively. The Amazon 5-class classification dataset (Amazon-5) consists of 3,000,000 and 650,000 reviews for training and test, respectively.

4.2 News Classification Datasets

AG News. The AG News dataset [50] is a collection of news articles collected from more than 2,000 news sources by ComeToMyHead, an academic news search engine. This dataset includes 120,000 training samples and 7,600 test samples. Each sample is a short text with a four-class label.

20 Newsgroups. The 20 Newsgroups dataset [190] is a collection of newsgroup documents posted on 20 different topics. Various versions of this dataset are used for text classification, text clustering and so one. One of the most popular versions contains 18,821 documents that are evenly classified across all topics.

Sogou News. The Sogou News dataset [154] is a mixture of the SogouCA and SogouCS news corpora. The classification labels of the news are determined by their domain names in the URL. For example, the news with URL http://sports.sohu.com is categorized as a sport class.

Reuters news. The Reuters-21578 dataset [191] is one of the most widely used data collections for text categorization, and is collected from the Reuters financial newswire service in 1987. ApteMod is a multi-class version of Reuters-21578 with 10,788 documents. It has 90 classes, 7,769 training documents and 3,019 test documents. Other datasets derived from a subset of the Reuters dataset include R8, R52, RCV1, and RCV1-v2.

Other datasets developed for news categorization includes: Bing news [192], BBC [193], Google news [194].

4.3 Topic Classification Datasets

DBpedia. The DBpedia dataset [195] is a large-scale, multilingual knowledge base that has been created from the most commonly used infoboxes within Wikipedia. DBpedia is published every month and some classes and properties are added or removed in each release. The most popular version of DBpedia contains 560,000 training samples and 70,000 test samples, each with a 14-class label.

Ohsumed. The Ohsumed collection [196] is a subset of the MEDLINE database. Ohsumed contains 7,400 documents. Each document is a medical abstract that is labeled by one or more classes selected from 23 cardiovascular diseases categories.

EUR-Lex. The EUR-Lex dataset [197] includes different types of documents, which are indexed according to several orthogonal categorization schemes to allow for multiple search facilities. The most popular version of this dataset is based on different aspects of European Union law and has 19,314 documents and 3,956 categories.

WOS. The **Web of Science (WOS)** dataset [136] is a collection of data and meta-data of published papers available from the Web of Science, which is the world's most trusted publisher-independent global citation database. WOS has been released in three versions: WOS-46985,

WOS-11967, and WOS-5736. WOS-46985 is the full dataset. WOS-11967 and WOS-5736 are two subsets of WOS-46985.

PubMed. PubMed [198] is a search engine developed by the National Library of Medicine for medical and biological scientific papers, which contains a document collection. Each document has been labeled with the classes of the MeSH set, which is a label set used in PubMed. Each sentence in an abstract is labeled with its role in the abstract using one of the following classes: background, objective, method, result, or conclusion.

Other datasets for topic classification includes PubMed 200k RCT [199], Irony (which is composed of annotated comments from the social news website reddit, Twitter dataset for topic classification of tweets, arXiv collection) [200], to name a few.

4.4 QA Datasets

SQuAD. **Stanford Question Answering Dataset (SQuAD)** [24] is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers of questions can be any sequence of tokens in the given text. Because the questions and answers are produced by humans through crowdsourcing, it is more diverse than some other question-answering datasets. SQuAD 1.1 contains 107,785 question-answer pairs on 536 articles. SQuAD2.0, the latest version, combines the 100,000 questions in SQuAD1.1 with over 50,000 un-answerable questions written adversarially by crowdworkers in forms that are similar to the answerable ones [201].

MS MARCO. This dataset is released by Microsoft [202]. Unlike SQuAD where all questions are produced by edits; In MS MARCO, all questions are sampled from user queries and passages from real web documents using the Bing search engine. Some of the answers in MS MARCO are generative. So, the dataset can be used to develop generative QA systems.

TREC-QA. TREC-QA [203] is one of the most popular and studied datasets for QA research. This dataset has two versions, known as TREC-6 and TREC-50. TREC-6 consists of questions in 6 categories while TREC-50 in fifty classes. For both versions, the training and test datasets contain 5,452 and 500 questions, respectively.

WikiQA. The WikiQA dataset [204] consists of a set of question-answer pairs, collected and annotated for open-domain QA research. The dataset also includes questions for which there is no correct answer, allowing researchers to evaluate answer triggering models.

Quora. The Quora dataset [205] is developed for paraphrase identification (to detect duplicate questions). For this purpose, the authors present a subset of Quora data that consists of over 400,000 question pairs. A binary value is assigned to each question pair indicating whether the two questions are the same or not.

Other datasets for QA includes Situations With Adversarial Generations (SWAG) [206], WikiQA [204], SelQA [207].

4.5 NLI Datasets

SNLI. The Stanford Natural Language Inference (SNLI) dataset [208] is widely used for NLI. This dataset consists of 550,152, 10,000, and 10,000 sentence pairs for training, development, and test, respectively. Each pair is annotated with one of the three labels: neutral, entailment, contradiction.

Multi-NLI. The Multi-Genre Natural Language Inference (MNLI) dataset [209] is a collection of 433k sentence pairs annotated with textual entailment labels. The corpus is an extension of SNLI,

covers a wider range of genres of spoken and written text, and supports a distinctive cross-genre generalization evaluation.

SICK. The Sentences Involving Compositional Knowledge (SICK) dataset [25] consists of about 10,000 English sentence pairs that are annotated with three labels: entailment, contradiction, and neutral.

MSRP. The Microsoft Research Paraphrase (MSRP) dataset [210] is commonly used for the text similarity task. MSRP consists of 4,076 samples for training and 1,725 samples for testing. Each sample is a sentence pair, annotated with a binary label indicating whether the two sentences are paraphrases or not.

Other NLI datasets includes **Semantic Textual Similarity (STS)** [211], RTE [212], SciTail [213], to name a few.

5 EXPERIMENTAL PERFORMANCE ANALYSIS

In this section, we first describe a set of metrics commonly used for evaluating TC models' performance, and then present a quantitative analysis of the performance of a set of DL-based TC models on popular benchmarks.

5.1 Popular Metrics for Text Classification

Accuracy and Error Rate. These are primary metrics to evaluate the quality of a classification model. Let TP, FP, TN, FN denote true positive, false positive, true negative, and false negative, respectively. The classification Accuracy and Error Rate are defined in Equation (2),

$$\textbf{Accuracy} = \frac{(\text{TP} + \text{TN})}{N}, \quad \textbf{Error rate} = \frac{(\text{FP} + \text{FN})}{N}, \tag{2}$$

where N is the total number of samples. Obviously, we have **Error Rate = 1 - Accuracy**.

Precision / Recall / F1 score. These are also primary metrics, and are more often used than accuracy or error rate for imbalanced test sets, e.g., the majority of the test samples have one class label. Precision and recall for binary classification are defined as Equation (3). The F1 score is the harmonic mean of the precision and recall, as in Equation (3). An F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0:

$$\textbf{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \textbf{Recall} \quad = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \textbf{F1-score} = \frac{2\ \text{Prec}\ \text{Rec}}{\text{Prec} + \text{Rec}}. \tag{3}$$

For multi-class classification problems, we can always compute precision and recall for each class label and analyze the individual performance on class labels or average the values to get the overall precision and recall.

Exact Match (EM). The exact match metric is a popular metric for question-answering systems, which measures the percentage of predictions that match any one of the ground truth answers exactly. EM is one of the main metrics used for SQuAD.

Mean Reciprocal Rank (MRR). MRR is often used to evaluate the performance of ranking algorithms in NLP tasks such as query-document ranking and QA. MRR is defined in Equation (4), where Q is a set of all possible answers, and $rank_i$ is the ranking position of the ground-truth answer:

$$\textbf{MRR} = \frac{1}{|Q|} \sum_{i=1}^{Q} \frac{1}{rank_i}. \tag{4}$$

Table 1. Accuracy of Deep Learning–based Text Classification Models on Sentiment Analysis Datasets (in Terms of Classification Accuracy), Evaluated on the IMDB, SST, Yelp, and Amazon Datasets

Method	IMDB	SST-2	Amazon-2	Amazon-5	Yelp-2	Yelp-5
Naive Bayes [43]	-	81.80	-	-	-	-
LDA [214]	67.40	-	-	-	-	-
BoW+SVM [31]	87.80	-	-	-	-	-
tf.Δ idf [215]	88.10	-	-	-	-	-
Char-level CNN [50]	-	-	94.49	59.46	95.12	62.05
Deep Pyramid CNN [49]	-	84.46	96.68	65.82	97.36	69.40
ULMFiT [216]	95.40	-	-	-	97.84	70.02
BLSTM-2DCNN [40]	-	89.50	-	-	-	-
Neural Semantic Encoder [95]	-	89.70	-	-	-	-
BCN+Char+CoVe [217]	91.80	90.30	-	-	-	-
GLUE ELMo baseline [22]	-	90.40	-	-	-	-
BERT ELMo baseline [7]	-	90.40	-	-	-	-
CCCapsNet [76]	-	-	94.96	60.95	96.48	65.85
Virtual adversarial training [173]	94.10	-	-	-	-	-
Block-sparse LSTM [218]	94.99	93.20	-	-	96.73	
BERT-base [7, 154]	95.63	93.50	96.04	61.60	98.08	70.58
BERT-large [7, 154]	95.79	94.9	96.07	62.20	98.19	71.38
ALBERT [147]	-	95.20	-	-	-	-
Multi-Task DNN [23]	83.20	95.60	-	-	-	-
Snorkel MeTaL [219]	-	96.20	-	-	-	-
BERT Finetune + UDA [220]	95.80		96.50	62.88	97.95	62.92
RoBERTa (+additional data) [146]	-	96.40	-	-	-	-
XLNet-Large (ensemble) [156]	96.21	96.80	97.60	67.74	98.45	72.20

Italic indicates the non-deep-learning models.

Other widely used metrics include **Mean Average Precision (MAP)**, **Area Under Curve (AUC)**, False Discovery Rate, False Omission Rate, to name a few.

5.2 Quantitative Results

We tabulate the performance of several of the previously discussed algorithms on popular TC benchmarks. In each table, in addition to the results of a set of representative DL models, we also present results using non-deep-learning models that are either previous state of the art or widely used as baselines before the DL era. We can see that across all these tasks, the use of DL models leads to significant improvements.

Table 1 summarizes the results of the models described in Section 2 on several sentiment analysis datasets, including Yelp, IMDB, SST, and Amazon. We can see that significant improvement in accuracy has been obtained since the introduction of the first DL-based sentiment analysis model, e.g., with around 78% relative reduction in classification error (on SST-2).

Table 2 reports the performance on three news categorization datasets (i.e., AG News, 20-NEWS, Sogou News) and two topic classification datasets (i.e., DBpedia and Ohsummed). A similar trend to that in sentiment analysis is observed.

Tables 3 and 4 present the performance of some DL models on SQuAD, and WikiQA, respectively. It is worth noting that on both datasets the significant performance lift is attributed to the use of BERT.

Table 2. Accuracy of Classification Models on News Categorization, and Topic Classification Tasks

	News Categorization			**Topic Classification**	
Method	AG News	20NEWS	Sogou News	DBpedia	Ohsumed
Hierarchical Log-bilinear Model [221]	-	-	-	-	52
Text GCN [107]	67.61	86.34	-	-	68.36
Simplfied GCN [108]	-	88.50	-	-	68.50
Char-level CNN [50]	90.49	-	95.12	98.45	-
CCCapsNet [76]	92.39	-	97.25	98.72	-
LEAM [84]	92.45	81.91	-	99.02	58.58
fastText [30]	92.50	-	96.80	98.60	55.70
CapsuleNet B [71]	92.60	-	-	-	-
Deep Pyramid CNN [49]	93.13	-	98.16	99.12	-
ULMFiT [216]	94.99	-	-	99.20	-
L MIXED [174]	95.05	-	-	99.30	-
BERT-large [220]	-	-	-	99.32	-
XLNet [156]	95.51	-	-	99.38	-

Italic indicates the non-deep-learning models.

Table 3. Performance of Classification Models on SQuAD Question Answering Datasets

	SQuAD1.1		SQuAD2.0	
Method	EM	F1-score	EM	F1-score
Sliding Window+Dist. [222]	13.00	20.00	—	—
Hand-crafted Features+Logistic Regression [24]	40.40	51.00	—	—
BiDAF + Self Attention + ELMo [4]	78.58	85.83	63.37	66.25
SAN (single model) [137]	76.82	84.39	68.65	71.43
FusionNet++ (ensemble) [223]	78.97	86.01	70.30	72.48
SAN (ensemble) [137]	79.60	86.49	71.31	73.70
BERT (single model) [7]	85.08	91.83	80.00	83.06
BERT-large (ensemble) [7]	87.43	93.16	80.45	83.51
BERT + Multiple-CNN [137]	—	—	84.20	86.76
XL-Net [156]	89.90	95.08	84.64	88.00
SpanBERT [149]	88.83	94.63	71.31	73.70
RoBERTa [146]	—	—	86.82	89.79
ALBERT (single model) [147]	—	—	88.10	90.90
ALBERT (ensemble) [147]	—	—	89.73	92.21
Retro-Reader on ALBERT	—	—	90.11	92.58
ELECTRA+ALBERT+EntitySpanFocus	—	—	90.42	92.79

Here, the F1 score measures the average overlap between the prediction and ground truth answer. Italic denotes the non-deep-learning models.

Table 4. Performance of Classification Models on the WikiQA Datasets

Method	MAP	MRR
Paragraph vector [32]	0.511	0.516
Neural Variational Inference [166]	0.655	0.674
Attentive pooling networks [83]	0.688	0.695
HyperQA [127]	0.712	0.727
BERT (single model) [7]	0.813	0.828
TANDA-RoBERTa [153]	0.920	0.933

Table 5. Performance of Classification Models on Natural Language Inference Datasets

	SNLI	MultiNLI	
Method	Accuracy	Matched	Mismatched
Unigrams Features [208]	71.6	—	—
Lexicalized [208]	78.2	—	—
LSTM encoders (100D) [208]	77.6	—	—
Tree-based CNN [61]	82.1	—	—
biLSTM Encoder [209]	81.5	67.5	67.1
Neural Semantic Encoders (300D) [95]	84.6	—	—
RNN-based Sentence Encoder [224]	85.5	73.2	73.6
DiSAN (300D) [81]	85.6	—	—
Decomposable Attention Model [92]	86.3	—	—
Reinforced Self-Attention (300D) [177]	86.3	—	—
Generalized Pooling (600D) [93]	86.6	73.8	74.0
Bilateral multi-perspective matching [41]	87.5	—	—
Multiway Attention Network [87]	88.3	78.5	77.7
ESIM + ELMo [4]	88.7	72.9	73.4
DMAN with Reinforcement Learning [225]	88.8	88.8	78.9
BiLSTM + ELMo + Attn [22]	—	74.1	74.5
Fine-Tuned LM-Pretrained Transformer [6]	89.9	82.1	81.4
Multi-Task DNN [23]	91.6	86.7	86.0
SemBERT [155]	91.9	84.4	84.0
RoBERTa [146]	92.6	90.8	90.2
XLNet [156]	—	90.2	89.8

For Multi-NLI, Matched and Mismatched refer to the matched and mismatched test accuracies, respectively. Italic denotes the non-deep-learning models.

Table 5 presents the results on two NLI datasets (i.e., SNLI and MNLI). We observe a steady performance improvement on both datasets over the last 5 years.

6 CHALLENGES AND OPPORTUNITIES

TC has seen a great progress over the last few years, with the help of DL models. Several novel ideas have been proposed (such as neural embedding, attention mechanism, self attention, Transformer, BERT, and XLNet), which lead to the fast progress over the past decade. Despite the progress, there

are still challenges to be addressed. This section presents some of these challenges, and discusses research directions that could help advance the field.

New Datasets for More Challenging Tasks. Although a number of large-scale datasets have been collected for common TC tasks in recent years, there remains a need for new datasets for more challenging TC tasks such as QA with multi-step reasoning, text classification for multilingual documents, and TC for extremely long documents.

Modeling Commonsense Knowledge. Incorporating commonsense knowledge into DL models has a potential to significantly improve model performance, pretty much in the same way that humans leverage commonsense knowledge to perform different tasks. For example, a QA system equipped with a commonsense knowledge base could answer questions about the real world. Commonsense knowledge also helps to solve problems in the case of incomplete information. Using widely held beliefs about everyday objects or concepts, AI systems can reason based on "default" assumptions about the unknowns in a similar way people do. Although this idea has been investigated for sentiment classification, much more research is required to explore to effectively model and use commonsense knowledge in DL models.

Interpretable DL Models. While DL models have achieved promising performance on challenging benchmarks, most of these models are not interpretable. For example, why does a model outperform another model on one dataset, but underperform on other datasets? What exactly have DL models learned? What is a minimal neural network architecture that can achieve a certain accuracy on a given dataset? Although the attention and self-attention mechanisms provide some insight toward answering these questions, a detailed study of the underlying behavior and dynamics of these models is still lacking. A better understanding of the theoretical aspects of these models can help develop better models curated toward various text analysis scenarios.

Memory Efficient Models. Most modern neural language models require a significant amount of memory for training and inference. These models have to be compressed to meet the computation and storage constraints of edge applications. This can be done either by building student models using knowledge distillation, or by using model compression techniques. Developing a task-agnostic model compression method is an active research topic [226].

Few-Shot and Zero-Shot Learning. Most DL models are supervised models that require large amounts of domain labels. In practice, it is expensive to collect such labels for each new domain. Fine-tuning a PLM (e.g., BERT and OpenGPT) to a specific task requires much fewer domain labels than training a model from scratch, thus opening opportunities of developing new zero-shot or few-shot learning methods based on PLMs.

7 CONCLUSION

In this article, we survey more than 150 DL models, which have been developed over the past six years and have significantly improved state of the art on various TC tasks. We also provide an overview of more than 40 popular TC datasets, and we present a quantitative analysis of the performance of these models on several public benchmarks. Finally, we discuss some of the open challenges and future research directions.

ACKNOWLEDGMENTS

The authors thank Richard Socher, Kristina Toutanova, Brooke Cowan, and all the anonymous reviewers for reviewing this work and providing very insightful comments.

REFERENCES

[1] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *J. Amer. Soc. Info. Sci.* 41, 6 (1990), 391–407.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3 (Feb. 2003), 1137–1155.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. MIT Press, 3111–3119.

[4] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. Retrieved from https://arXiv:1802.05365 (2018).

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. MIT Press, 5998–6008.

[6] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Retrieved from https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper.pdf.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.Retrieved from https://arXiv:1810.04805.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. 2020. Language models are few-shot learners. Retrieved from https://arXiv:2005.14165.

[9] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. Retrieved from https://arXiv:2006.16668.

[10] Gary Marcus and Ernest Davis. 2019. *Rebooting AI: Building Artificial Intelligence We Can Trust*. Pantheon.

[11] Gary Marcus. 2020. The next decade in ai: Four steps towards robust artificial intelligence. Retrieved from https://arXiv:2002.06177.

[12] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. Retrieved from https://arXiv:1910.14599.

[13] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? Natural language attack on text classification and entailment. Retrieved from https://arXiv:1907.11932 2.

[14] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. Adversarial training for large neural language models. Retrieved from https://arXiv:2004.08994.

[15] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. Retrieved from https://arXiv:1601.01705.

[16] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1821–1831.

[17] Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. 2019. Enhancing the transformer with explicit relational encoding for math problem solving. Retrieved from https://arXiv:1910.06611.

[18] Jianfeng Gao, Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Heung-Yeung Shum. 2020. Robust conversational AI with grounded text generation. Retrieved from https://arXiv:2009.03457.

[19] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information* 10, 4 (2019), 150.

[20] Christopher D. Manning, Hinrich Schütze, and Prabhakar Raghavan. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[21] Daniel Jurasky and James H. Martin. 2008. Speech and language processing: An introduction to natural language Processing. *Computational Linguistics and Speech Recognition*. Prentice Hall, NJ.

[22] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. Retrieved from https://arXiv:1804.07461.

[23] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. Retrieved from https://arXiv:1901.11504.

[24] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. Retrieved from https://arXiv:1606.05250.

[25] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through

semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval'14)*. 1–8.
[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. Retrieved from https://arXiv:1301.3781.
[28] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1532–1543.
[29] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 1681–1691.
[30] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. Retrieved from https://arXiv:1612.03651.
[31] Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 90–94.
[32] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*. 1188–1196.
[33] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. Retrieved from https://arXiv:1503.00075.
[34] Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the International Conference on Machine Learning*. 1604–1612.
[35] Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. Retrieved from https://arXiv:1601.06733.
[36] Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuan-Jing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2326–2335.
[37] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. Retrieved from https://arXiv:1611.01702.
[38] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. Retrieved from https://arXiv:1605.05101.
[39] Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using LSTM for region embeddings. Retrieved from https://arXiv:1602.02373.
[40] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. Retrieved from https://arXiv:1611.06639.
[41] Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. Retrieved from https://arXiv:1702.03814.
[42] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.
[43] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1631–1642.
[44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[45] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*. DOI: http://dx.doi.org/10.3115/v1/p14-1062 arxiv:1404.2188
[46] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. DOI: http://dx.doi.org/10.3115/v1/d14-1181 arxiv:1408.5882
[47] Jingzhou Liu, Wei Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM Conference on Research and Development in Information Retrieval (SIGIR'17)*. DOI: http://dx.doi.org/10.1145/3077136.3080834
[48] Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference (HLT'15)*. DOI: http://dx.doi.org/10.3115/v1/n15-1011 arxiv:1412.1058

[49] Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 562–570.
[50] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*. MIT Press, 649–657.
[51] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.
[52] Joseph D. Prusa and Taghi M. Khoshgoftaar. 2016. Designing a better data representation for deep neural networks and text classification. In *Proceedings of the IEEE 17th International Conference on Information Reuse and Integration (IRI'16)*. DOI: http://dx.doi.org/10.1109/IRI.2016.61
[53] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*. Retrieved from https://arxiv:1409.1556.
[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. DOI: http://dx.doi.org/10.1109/CVPR.2016.90 arxiv:1512.03385
[55] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. Retrieved from https://arXiv:1606.01781.
[56] Andréa B. Duque, Luã Lázaro J. Santos, David Macêdo, and Cleber Zanchettin. 2019. Squeezed very deep convolutional neural networks for text classification. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. DOI: http://dx.doi.org/10.1007/978-3-030-30487-4_16 Retrieved from https://arxiv:1901.09821.
[57] Hoa T. Le, Christophe Cerisara, and Alexandre Denis. 2018. Do convolutional networks need to be deep for text classification? In *Proceedings of the Workshops at the 32nd AAAI Conference on Artificial Intelligence*.
[58] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. DOI: http://dx.doi.org/10.1109/CVPR.2017.243 arxiv:1608.06993
[59] Bao Guo, Chunxia Zhang, Junmin Liu, and Xiaoyi Ma. 2019. Improving text classification with weighted word embeddings via a multi-channel TextCNN model. *Neurocomputing* (2019). DOI: http://dx.doi.org/10.1016/j.neucom.2019.07.052
[60] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. Retrieved from https://arXiv:1510.03820.
[61] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2015. Natural language inference by tree-based convolution and heuristic matching. Retrieved from https://arXiv:1512.08422.
[62] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. Retrieved from https://arxiv:1602.06359.
[63] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'17)*. DOI: http://dx.doi.org/10.24963/ijcai.2017/406
[64] Sarvnaz Karimi, Xiang Dai, Hamedh Hassanzadeh, and Anthony Nguyen. 2017. Automatic diagnosis coding of radiology reports: A comparison of deep learning and conventional classification methods. BioNLP. DOI: http://dx.doi.org/10.18653/v1/w17-2342
[65] Shengwen Peng, Ronghui You, Hongning Wang, Chengxiang Zhai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2016. DeepMeSH: Deep semantic representation for improving large-scale MeSH indexing. *Bioinformatics* (2016). Retrieved from DOI: http://dx.doi.org/10.1093/bioinformatics/btw294
[66] Anthony Rios and Ramakanth Kavuluru. 2015. Convolutional neural networks for biomedical text classification: Application in indexing biomedical articles. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB'15)*. DOI: http://dx.doi.org/10.1145/2808719.2808746
[67] Mark Hughes, Irene Li, Spyros Kotoulas, and Toyotaro Suzumura. 2017. Medical text classification using convolutional neural networks. *Studies Health Technol. Info.* (2017). DOI: http://dx.doi.org/10.3233/978-1-61499-753-5-246 Retrieved from https://arxiv:1704.06841.
[68] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming auto-encoders. In *Proceedings of the International Conference on Artificial Neural Networks*. Springer, 44–51.
[69] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*. MIT Press, 3856–3866.
[70] Sara Sabour, Nicholas Frosst, and Geoffrey Hinton. 2018. Matrix capsules with EM routing. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*. 1–15.

[71] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. 2018. Investigating capsule networks with dynamic routing for text classification. Retrieved from https://arXiv:1804.00538.
[72] Min Yang, Wei Zhao, Lei Chen, Qiang Qu, Zhou Zhao, and Ying Shen. 2019. Investigating the transferring capability of capsule networks for text classification. *Neural Netw.* 118 (2019), 247–261.
[73] Wei Zhao, Haiyun Peng, Steffen Eger, Erik Cambria, and Min Yang. 2019. Towards scalable and reliable capsule networks for challenging NLP applications. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 1549–1559.
[74] Jaeyoung Kim, Sion Jang, Eunjeong Park, and Sungchul Choi. 2020. Text classification using capsules. *Neurocomputing* 376 (2020), 214–221.
[75] Rami Aly, Steffen Remus, and Chris Biemann. 2019. Hierarchical multi-label classification of text with capsule networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. 323–330.
[76] Hao Ren and Hong Lu. 2018. Compositional coding capsule network with k-means routing for text classification. Retrieved from https://arXiv:1810.09177.
[77] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. Retrieved from https://arXiv:1409.0473.
[78] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. Retrieved from https://arXiv:1508.04025.
[79] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.
[80] Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based LSTM network for cross-lingual sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 247–256.
[81] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[82] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. Retrieved from https://arXiv:1605.09090.
[83] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. Retrieved from https://arXiv:1602.03609.
[84] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. Retrieved from https://arXiv:1805.04174.
[85] Seonhoon Kim, Inho Kang, and Nojun Kwak. 2019. Semantic sentence matching with densely-connected recurrent and co-attentive information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6586–6593.
[86] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Trans. Assoc. Comput. Ling.* 4 (2016), 259–272.
[87] Chuanqi Tan, Furu Wei, Wenhui Wang, Weifeng Lv, and Ming Zhou. 2018. Multiway attention networks for modeling sentence pairs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'18)*. 4411–4417.
[88] Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 287–296.
[89] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. Retrieved from https://arXiv:1703.03130.
[90] Shiyao Wang, Minlie Huang, and Zhidong Deng. 2018. Densely connected CNN with multi-scale feature attention for text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'18)*. 4468–4474.
[91] Ikuya Yamada and Hiroyuki Shindo. 2019. Neural attentive bag-of-entities model for text classification. Retrieved from https://arXiv:1909.01259.
[92] Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. Retrieved from https://arXiv:1606.01933.
[93] Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. 2018. Enhancing sentence embedding with generalized pooling. Retrieved from https://arXiv:1806.09828.
[94] Mohammad Ehsan Basiri, Shahla Nemati, Moloud Abdar, Erik Cambria, and U. Rajendra Acharya. 2020. ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis. *Future Gen. Comput. Syst.* 115 (2020), 279–294.

[95] Tsendsuren Munkhdalai and Hong Yu. 2017. Neural semantic encoders. In *Proceedings of the Conference of the Association for Computational Linguistics*, Vol. 1. NIH Public Access, 397.

[96] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*. Retrieved from https://arxiv:1410.3916.

[97] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. MIT Press, 2440–2448.

[98] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*. Retrieved from https://arXiv:1506.07285.

[99] Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*. Retrieved from https://arxiv:1603.01417.

[100] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 404–411.

[101] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. Retrieved from https://arXiv:1901.00596.

[102] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. Retrieved from https://arXiv:1609.02907.

[103] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. MIT Press, 1024–1034.

[104] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. Retrieved from https://arXiv:1710.10903.

[105] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1063–1072.

[106] Hao Peng, Jianxin Li, Qiran Gong, Senzhang Wang, Lifang He, Bo Li, Lihong Wang, and Philip S. Yu. 2019. Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification. Retrieved from https://arXiv:1906.04898.

[107] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7370–7377.

[108] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying graph convolutional networks. Retrieved from https://arXiv:1902.07153.

[109] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. Retrieved from https://arXiv:1910.02356.

[110] Pengfei Liu, Shuaichen Chang, Xuanjing Huang, Jian Tang, and Jackie Chi Kit Cheung. 2019. Contextualized non-local neural networks for sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6762–6769.

[111] Jane Bromley, James W. Bentz, Leon Bottou, Isabelle Guyon, Yann Lecun, Cliff Moore, Eduard Sackinger, and Roopak Shah. 1993. Signature verification using a Siamese time delay neural network. *Int. J. Pattern Recogn. Artific. Intell.* 7, 4 (1993), 669–688. DOI: http://dx.doi.org/10.1142/s0218001493000339

[112] Wen tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the 15th Conference on Computational Natural Language Learning (CoNLL'11)*.

[113] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. 2333–2338.

[114] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the ACM International Conference on Conference on Information and Knowledge Management*. ACM, 101–110.

[115] Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational ai. *Found. Trends Info. Retriev.* 13, 2-3 (2019), 127–298.

[116] Aliaksei Severyn and Alessandro Moschittiy. 2015. 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 (SIGIR'15)*. DOI: http://dx.doi.org/10.1145/2766462.2767738

[117] Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. DOI: http://dx.doi.org/10.18653/v1/p16-1036

[118] Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. DOI: http://dx.doi.org/10.18653/v1/p16-1044

[119] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*.

[120] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. Retrieved from DOI: http://dx.doi.org/10.18653/v1/w16-1617.

[121] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Modelling interaction of sentence pair with coupled-lstms. Retrieved from https://arXiv:1605.05573.

[122] Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*. DOI: http://dx.doi.org/10.18653/v1/d15-1181

[123] Tom Renter, Alexey Borisov, and Maarten De Rijke. 2016. Siamese CBOW: Optimizing word embeddings for sentence representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. DOI: http://dx.doi.org/10.18653/v1/p16-1089 arxiv:1606.04640

[124] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-Networks. DOI: http://dx.doi.org/10.18653/v1/d19-1410 Retrieved from https://arxiv:1908.10084.

[125] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. TwinBERT: Distilling knowledge to twin-structured BERT models for efficient retrieval. Retrieved from https://arXiv:2002.06275.

[126] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. Retrieved from https://arXiv:1511.04108.

[127] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 583–591.

[128] Shervin Minaee and Zhu Liu. 2017. Automatic question-answering using a deep similarity neural network. In *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP'17)*. IEEE, 923–927.

[129] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A C-LSTM neural network for text classification. Retrieved from https://arXiv:1511.08630.

[130] Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT'16)*. DOI: http://dx.doi.org/10.18653/v1/n16-1177 arxiv:1611.02361

[131] Guibin Chen, Deheng Ye, Erik Cambria, Jieshan Chen, and Zhenchang Xing. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'17)*. 2377–2383.

[132] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1422–1432.

[133] Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. Retrieved from https://arXiv:1602.00367.

[134] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*.

[135] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. *Expert Syst. Appl.* 72 (2017), 221–230. DOI: http://dx.doi.org/10.1016/j.eswa.2016.10.065

[136] Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. Hdltex: Hierarchical deep learning for text classification. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA'17)*. IEEE, 364–371.

[137] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. Retrieved from https://arXiv:1712.03556.

[138] Rupesh Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*. Retrieved from https://arxiv:1507.06228.

[139] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[140] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware neural language models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*. Retrieved from https://arxiv:1508.06615.

[141] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutnik, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*. Retrieved from https://arxiv:1607.03474.

[142] Ying Wen, Weinan Zhang, Rui Luo, and Jun Wang. 2016. Learning text representation using recurrent convolutional neural network with highway layers. Retrieved from https://arXiv:1606.06905.
[143] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12 (Aug. 2011), 2493–2537.
[144] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.
[145] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. Retrieved from https://arXiv:2003.08271.
[146] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. Retrieved from https://arXiv:1907.11692.
[147] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. Retrieved from https://arXiv:1909.11942.
[148] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. Retrieved from https://arXiv:1910.01108.
[149] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. Retrieved from https://arXiv:1907.10529.
[150] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. Retrieved from https://arXiv:2003.10555.
[151] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. Retrieved from https://arXiv:1904.09223.
[152] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'20)*. 8968–8975.
[153] Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2019. TANDA: Transfer and adapt pre-trained transformer models for answer sentence selection. Retrieved from https://arXiv:1911.04118.
[154] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification?. In *China National Conference on Chinese Computational Linguistics*. Springer, 194–206.
[155] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2019. Semantics-aware BERT for language understanding. Retrieved from https://arXiv:1909.02209.
[156] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*. MIT Press, 5754–5764.
[157] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*. MIT Press, 13042–13054.
[158] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou et al. 2020. UniLMv2: Pseudo-masked language models for unified language model pre-training. Retrieved from https://arXiv:2002.12804.
[159] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. Retrieved from https://arXiv:1910.10683.
[160] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning Internal Representations by Error Propagation*. Technical Report. University of California San Diego, La Jolla Institute for Cognitive Science.
[161] Ryan Kiros, Yukun Zhu, Russ R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*. MIT Press, 3294–3302.
[162] Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. Retrieved from https://arxiv:1511.01432.
[163] Minghua Zhang, Yunfang Wu, Weikang Li, and Wei Li. 2019. Learning universal sentence representations with mean-max attention autoencoder. DOI: http://dx.doi.org/10.18653/v1/d18-1481 Retrieved from https://arxiv:1809.06590.
[164] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*. arxiv:1312.6114
[165] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *Proceedings of the International Conference on Machine Learning (ICML'14)*.

[166] Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of the International Conference on Machine Learning*.
[167] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. DOI: http://dx.doi.org/10.18653/v1/k16-1002 Retrieved from https://arxiv:1511.06349.
[168] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A Smith. 2019. Variational pretraining for semi-supervised text classification. Retrieved from https://arXiv:1906.02242.
[169] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the Conference on Information and Knowledge Management (CIKM'18)*.
[170] Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the Meeting of the Association for Computational Linguistics (ACL'20)*.
[171] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. Retrieved from https://arXiv:1412.6572.
[172] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.
[173] Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. Retrieved from https://arXiv:1605.07725.
[174] Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. 2019. Revisiting LSTM networks for semi-supervised text classification via mixed objective function. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6940–6948.
[175] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. Retrieved from https://arXiv:1704.05742.
[176] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press.
[177] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018. Reinforced self-attention network: A hybrid of hard and soft attention for sequence modeling. Retrieved from https://arXiv:1801.10296.
[178] Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. 2020. Finding decision jumps in text classification. *Neurocomputing* 371 (2020), 177–187.
[179] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1047–1055.
[180] Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. 2018. A generative model for category text generation. *Info. Sci.* 450 (2018), 301–315.
[181] Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[182] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. Retrieved from https://arXiv:2007.15779.
[183] Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. XtremeDistil: Multi-stage distillation for massive multilingual models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2221–2234.
[184] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from BERT into simple neural networks. Retrieved from https://arXiv:1903.12136.
[185] kaggle.[n. d.]. Retrieved from https://www.kaggle.com/yelp-dataset/yelp-dataset.
[186] kaggle. [n. d.]. Retrieved from https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.
[187] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing*. 79–86.
[188] Lingjia Deng and Janyce Wiebe. 2015. MPQA 3.0: An entity/event-level sentiment corpus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1323–1328.
[189] kaggle. [n.d.]. Retrieved from https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products.
[190] 20 Newsgroups. [n.d.]. Retrieved from http://qwone.com/jason/20Newsgroups/.
[191] Reuters. [n.d.]. Retrieved from https://martin-thoma.com/nlp-reuters.
[192] Fang Wang, Zhongyuan Wang, Zhoujun Li, and Ji-Rong Wen. 2014. Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 1069–1078.

[193] Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 377–384.
[194] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, 271–280.
[195] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer et al. 2015. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.
[196] Ohsumed. [n.d.]. Retrieved from http://davis.wpi.edu/xmdv/datasets/ohsumed.html.
[197] Eneldo Loza Mencia and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 50–65.
[198] Zhiyong Lu. 2011. PubMed and beyond: A survey of web tools for searching biomedical literature. Retrieved from https://pubmed.ncbi.nlm.nih.gov/21245076/.
[199] Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: A dataset for sequential sentence classification in medical abstracts. Retrieved from https://arXiv:1710.06071.
[200] Byron C. Wallace, Laura Kertz, Eugene Charniak et al. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 512–516.
[201] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. Retrieved from https://arXiv preprint:1806.03822.
[202] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human-generated machine reading comprehension dataset. CoCo@ NIPS.
[203] University of Pennsylvania [n.d.]. Retrieved from https://cogcomp.seas.upenn.edu/Data/QA/QC/.
[204] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2013–2018.
[205] Quora. [n.d.]. Retrieved from https://data.quora.com/First-Quora-Dataset-Release-QuestionPairs.
[206] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. Retrieved from https://arXiv:1808.05326.
[207] Tomasz Jurczyk, Michael Zhai, and Jinho D. Choi. 2016. Selqa: A new benchmark for selection-based question answering. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 820–827.
[208] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. Retrieved from https://arXiv:1508.05326.
[209] Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. Retrieved from https://arXiv:1704.05426.
[210] Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*. ACL, 350.
[211] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. Retrieved from https://arXiv:1708.00055.
[212] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. DOI: http://dx.doi.org/10.1007/11736790_9
[213] Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*.
[214] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 142–150.
[215] Justin Christopher Martineau and Tim Finin. 2009. Delta tfidf: An improved feature space for sentiment analysis. In *Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media*.
[216] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. Retrieved from https://arXiv:1801.06146.
[217] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. MIT Press, 6294–6305.
[218] Scott Gray, Alec Radford, and Diederik P. Kingma. 2017. Gpu kernels for block-sparse weights. Retrieved from https://arXiv:1711.09224.

[219] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4763–4771.

[220] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation. Retrieved from https://arXiv:1904.12848.

[221] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the International Conference on Machine Learning*. 957–966.

[222] Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 193–203.

[223] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. 2017. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. Retrieved from https://arXiv:1711.07341.

[224] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. Retrieved from https://arXiv:1708.01353

[225] Boyuan Pan, Yazheng Yang, Zhou Zhao, Yueting Zhuang, Deng Cai, and Xiaofei He. 2019. Discourse marker augmented network with reinforcement learning for natural language inference. Retrieved from https://arXiv:1907.09692.

[226] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Retrieved from https://arXiv:2002.10957.

Received April 2020; revised October 2020; accepted November 2020