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, *University of Tabriz*
Meysam Chenaghlu, *University of Tabriz*
Jianfeng Gao, *Microsoft Research, Redmond*

Abstract. 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 work, we provide a detailed review of more than 150 deep learning based models for text classification developed in recent years, and 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 discuss future research directions.

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. 2020. Deep Learning Based Text Classification: A Comprehensive Review. 1, 1 (April 2020), 42 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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, for example web data, emails, chats, social media, tickets, insurance claims, user reviews, questions and answers from customer services, and many more. Text is an extremely rich source of information, but extracting insights from it 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 three categories:

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

Authors' addresses: Shervin Minaee *Snapchat Inc*; Nal Kalchbrenner *Google Brain, Amsterdam*; Erik Cambria *Nanyang Technological University, Singapore*; Narjes Nikzad *University of Tabriz*; Meysam Chenaghlu *University of Tabriz*; Jianfeng Gao *Microsoft Research, Redmond*.

XXXX-XXXX/2020/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Rule-based methods classify text into different categories using a set of pre-defined rules. For example, any document with the words "football," "basketball," or "baseball" is assigned the "sport" label. These methods require a deep knowledge of the domain, and the systems are difficult to maintain. On the other hand, machine learning based approaches learn to make classifications based on past observations of the data. Using pre-labeled examples as training data, a machine learning algorithm can learn the inherent associations between pieces of text and their labels. Thus, machine learning based methods can detect hidden patterns in the data, are more scalable, and can be applied to various tasks. This is in contrast to rule-based methods, which need different sets of rules for different tasks. Hybrid methods, as the name suggests, use a combination of rule-based and machine learning methods to make predictions.

Machine learning models have drawn a lot of attention in recent years. Most classical machine learning based models follow the popular two-step procedure, where in the first step some hand-crafted features are extracted from the documents (or any other textual unit), and in the second step those features are fed to a classifier to make a prediction. Some of the 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 approaches have several limitations. For example, reliance on the hand-crafted features requires tedious feature engineering and analysis to obtain a good performance. In addition, the strong dependence on domain knowledge for designing features makes the method difficult to easily 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.

A paradigm shift started occurring in 2012, when a deep learning based model, AlexNet [1], won the ImageNet competition by a large margin. Since then, deep learning models have been applied to a wide range of tasks in computer vision and NLP, improving the state-of-the-art [2–5]. These models try to learn feature representations and perform classification (or regression), in an end-to-end fashion. They not only have the ability to uncover hidden patterns in data, but also are much more transferable from one application to another. Not surprisingly, these models are becoming the mainstream framework for various text classification tasks in recent years.

In this survey, we review more than 150 deep learning 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. We group these works into several categories based on their neural network architectures, such as models based on recurrent neural networks (RNNs), convolutional neural networks (CNNs), attention, Transformers, Capsule Nets, and more. The contributions of this paper can be summarized as follows:

- We present a detailed overview of more than 150 deep learning 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 deep learning models on 16 popular benchmarks.
- We discuss remaining challenges and future directions.

1.1 Text Classification Tasks

This section briefly introduces different text classification tasks discussed in this paper: sentiment analysis, news categorization, topic classification, question answering (QA), and natural language inference (NLI).

Sentiment Analysis. Sentiment analysis is a popular branch of text classification, which aims to analyze people's opinions in textual data (such as product reviews, movie reviews, and tweets), and extract their polarity and viewpoint. Sentiment classification can be either a binary or a multi-class problem. Binary sentiment analysis is the classification of texts into positive and negative classes, while multi-class sentiment analysis focuses on classifying data into fine-grained labels or multi-level intensities.

News Categorization. News contents are one of the most important sources of information that have a strong influence on people. A news classification system can help users obtain information of interest in real-time. Identifying emerging news topics and recommending relevant news based on user interests are two main applications of news classification.

Topic Analysis. Topic analysis tries to automatically obtain meaning from texts by identifying their topics. Topic classification is one of the most important component technologies for topic analysis. The goal of topic classification is to assign one or more topics to each of the documents to make it easier to analyze.

Question Answering (QA). There are two types of QA systems: extractive and generative. Extractive QA can be viewed as a special case of text classification. Given a question and a set of candidate answers (e.g., text spans in a given document in SQuAD [6]), we need to classify each candidate answer as correct or not. Generative QA learns to generate the answers from scratch (for example using a sequence-to-sequence model). The QA tasks discussed in this paper are extractive QA, unless otherwise stated.

Natural language inference (NLI). NLI, also known as *recognizing textual entailment* (RTE), predicts whether the meaning of one text can be inferred from another. In particular, a system needs to assign to each pair of text units a label such as entailment, contradiction, and neutral [7]. Paraphrasing is a generalized form of NLI, also known as *text pair comparison*. The task is to measure the semantic similarity of a sentence pair in order to determine whether one sentence is a paraphrase of the other.

1.2 Paper Structure

The rest of the paper is structured as follows: Section 2 presents a comprehensive overview of more than 150 deep learning based text classification models. Section 3 reviews some of the most popular text classification datasets. Section 4 presents a quantitative performance analysis of a selected set of deep learning models on 16 benchmarks. Section 5 discusses the main challenges and future directions for deep learning based text classification methods. Section 6 concludes the paper. Appendix A, provides an overview of some popular neural network model architectures that are commonly used for text classification.

2 DEEP LEARNING MODELS FOR TEXT CLASSIFICATION

In this section, we review more than 150 deep learning frameworks proposed for various text classification problems. To make it easier to follow, we group these models into the following categories, based on their main architectural contributions:

- Models based on feed-forward networks, which view text as a bag of words (Section 2.1).
- Models based on RNNs, which view text as a sequence of words, and are intended to capture word dependencies and text structures (Section 2.2).
- CNN-based models, which are trained to recognize patterns in text, such as key phrases, for classification (Section 2.3).
- Capsule networks, which address the information loss problem suffered by the pooling operations of CNNs, and recently have been applied to text classification (Section 2.4).
- The attention mechanism, which is effective to identify correlated words in text, and has become a useful tool in developing deep learning models (Section 2.5).
- Memory-augmented networks, which combine neural networks with a form of external memory, which the models can read from and write to (Section 2.6).
- Transformers, which allow for much more parallelization than RNNs, making it possible to efficiently (pre-)train very large language models using GPU clusters (Section 2.7).

- Graph neural networks, which are designed to capture internal graph structures of natural language, such as syntactic and semantic parse trees (Section 2.8).
- Siamese Neural Networks, designed for text matching, a special case of text classification (Section 2.9) .
- Hybrid models, which combine attention, RNNs, CNNs, etc. to capture local and global features of sentences and documents (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 deep learning models to comprehend the content of this section. For more details on the basic deep learning architectures and models, we refer the readers to the deep learning textbook by Goodfellow et al. [140], or the appendix of this paper.

2.1 Feed-Forward Neural Networks

Feed-forward networks are among the simplest deep learning models for text representation. Yet, they have achieved high accuracy on many text classification benchmarks. These models view a text as a bag of words. For each word, they learn a vector representation using an embedding model such as word2vec [8] or Glove [9], 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 [10]. An example of these models is the Deep Average Network (DAN) [10], whose architecture is shown in Fig. 1. Despite its simplicity, DAN outperforms other more sophisticated models which are designed to explicitly learn the compositionality of texts. For example, DAN outperforms syntactic models on datasets with high syntactic variance. Joulin et al. [11] propose a simple and efficient text classifier called fastText. Like DAN, fastText views a 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 while achieving comparable results to the methods that explicitly use the word order [12].

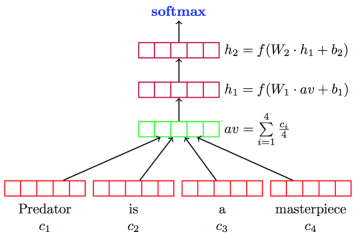

Fig. 1. The architecture of the Deep Average Network (DAN) [10].

Le and Mikolov [13] 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 Fig. 2, the architecture of doc2vec is similar to that of the Continuous Bag of Words (CBOW) model [8, 14]. 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 achieved new state-of-the-art results on several text classification and sentiment analysis tasks when it was published.

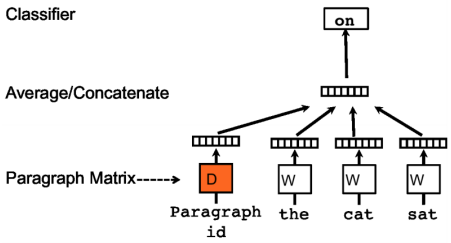

Fig. 2. The doc2vec model [13].

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 text classification. However, vanilla RNN models do not work well, and often underperform feed-forward neural networks. Among many variants of 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 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 text classification 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. [15] have developed 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 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 Fig. 3. Zhu et al. [16] 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.

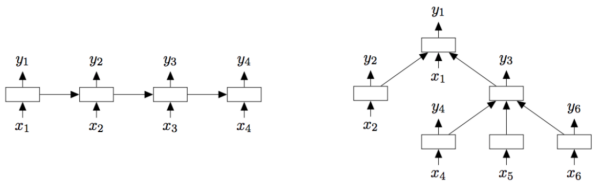

Fig. 3. (Left) A chain-structured LSTM network and (right) a tree-structured LSTM network with arbitrary branching factor [15].

To model long-span word relations for machine reading, Cheng et al. [17] 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 [18] 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 has been reported to outperform a set of baselines, including the models based on LSTM and RNN, on text classification.

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. Bieng et al. [19] 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 has been reported to outperform RNN baselines for sentiment analysis.

There are other interesting RNN-based models. Liu et al. [20] use multi-task learning to train RNNs to leverage labeled training data from multiple related tasks. Johnson and Rie [21] explore a text region embedding method using LSTM. Zhou et al. [22] integrate a Bidirectional-LSTM (Bi-LSTM) model with two-dimensional max pooling to capture text features. Wang et al. [23] propose a bilateral multi-perspective matching model under the "matching-aggregation" framework. Wan et al. [24] explore semantic matching using multiple positional sentence representations generated by a bi-directional LSMT model.

2.3 CNN-Based Models

RNNs are trained to recognize patterns across time, whereas CNNs learn to recognize patterns across space [25]. RNNs work well for 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 text classification.

One of the first CNN-based models for text classification is proposed by Kalchbrenner et al. [26]. This model uses dynamic k-max pooling, and is called the Dynamic CNN (DCNN). As illustrated in Fig. 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 [27] proposed a much simpler CNN-based model than DCNN for text classification. As shown in Fig. 5, Kim's model uses only one layer of convolution on top of word vectors obtained from an unsupervised neural language model i.e., word2vec. Kim also compared 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 during model training while the other fixed. These CNN-based models were 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 [26, 27]. Liu et al. [28] propose a new CNN-based model that makes two modifications to the architecture of Kim-CNN [27]. First, a dynamic max-pooling scheme is adopted to captures more fine-grained features from different regions of the document. Second, a hidden bottleneck layer is inserted between pooling and output layer to learn compact document

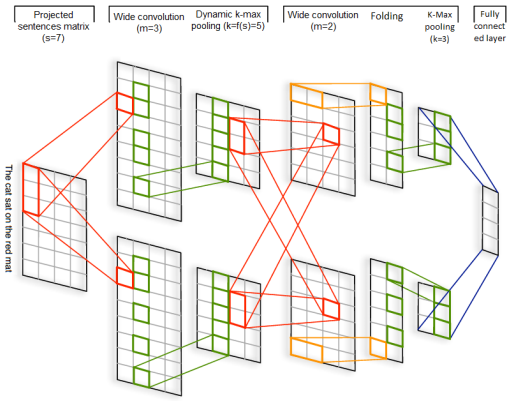

Fig. 4. The architecture of DCNN model [26].

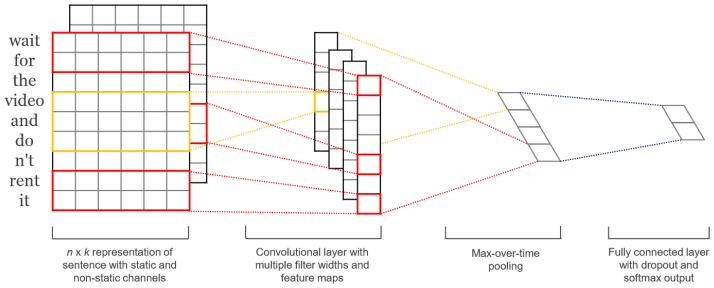

Fig. 5. The architecture of a sample CNN model for text classification. courtesy of Yoon Kim [27].

representations to reduce model size and boosts model performance. In [29, 30], 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 text classification [31, 32]. One of the first such models is proposed by Zhang et al. [31]. As illustrated in Fig. 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. [33] presented a 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 [34] and ResNets [35], Conneau et al. [36] presented 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 increases with the depth. Duque et al. [37] modify the structure of VDCNN to fit mobile platforms' constraints and keep performance. They were able to compress the model size by 10x to 20x with an accuracy loss between 0.4% to 1.3%. Le et al. [38] showed that deep models indeed outperform shallow models when the text input is represented as a sequence of characters. However, a simple

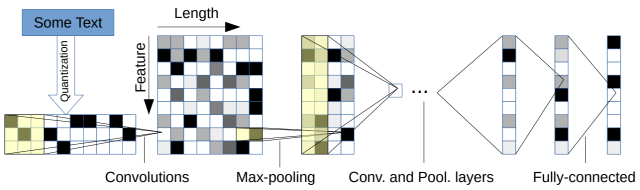

Fig. 6. The architecture of a character-level CNN model [31].

shallow-and-wide network outperforms deep models such as DenseNet[39] with word inputs. Guo et al. [40] studied the impact of word embedding and proposed to use weighted word embeddings via a multi-channel CNN model. Zhang et al. [41] examined the impact of different word embedding methods and pooling mechanisms, and found 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. [42] present a tree-based CNN to capture sentence-level semantics. Pang et al. [43] cast text matching as the image recognition task, and use multi-layer CNNs to identify salient n-gram patterns. Wang et al. [44] propose a CNN-based model that combines explicit and implicit representations of short text for classification. There is also a growing interest in applying CNNs to biomedical text classification [45–48].

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 Geoffrey Hinton, called capsule networks (CapsNets) [49, 50]. 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 "rout" 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 [50] or the EM algorithm [51].

Recently, capsule networks have been applied to text classification, where capsules are adapted to represent a sentence or document as a vector. [52–54] proposed a text classification 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 experimented 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 explored two capsule architectures, denoted as Capsule-A and Capsule-B as in Fig. 7. Capsule-A is similar to the CapsNet in [50]. 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. [55] 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

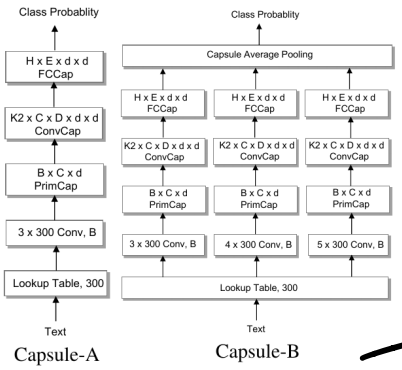

Fig. 7. CapsNet A and B for text classification [52].

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 [50] for text classification. Aly et al. [56] 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 [52, 53, 55].

Ren et al. [57] proposed 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 deep learning models for NLP [58, 59]. In a nutshell, attention in language models can be interpreted as a vector of importance weights. In order 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 which created new state-of-the-arts on text classification tasks, when they were published.

Yang et al. [60] proposed 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 outperformed previous methods by a substantial margin on six text classification tasks. Zhou et al. [61] extended 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. [62] presented 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. [63] presented a LSTM model with inner-attention for NLI. This model used a two-stage process to encode a sentence. Firstly, average pooling is used over word-level Bi-LSTM to generate a first stage sentence representation. Secondly, 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 matching tasks too. Santos et al. [64] proposed a two-way attention mechanism, known as Attentive Pooling (AP), for 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 Fig. 8 (a). Wang et al. [65] viewed text classification as a label-word matching problem: each label is embedded in the same space with the word vector. The authors introduced an attention framework that measures the compatibility of embeddings between text sequences and labels via cosine similarity, as shown in Fig. 8 (b).

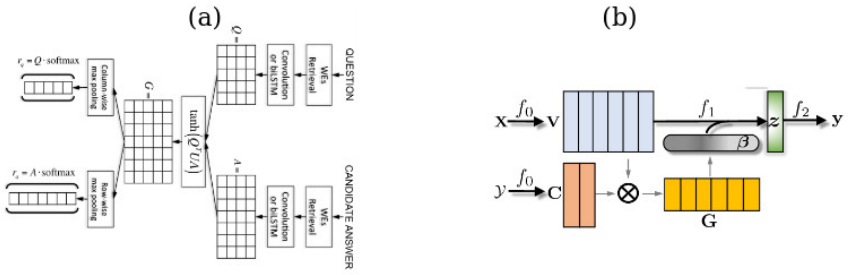

Fig. 8. (a) The architecture of attentive pooling networks [64]. (b) The architecture of label-text matching model [65].

Kim et al. [66] proposed a semantic sentence matching approach using a densely-connected recurrent and co-attentive network. Similar to DenseNet [39], 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 bottommost word embedding layer to the uppermost recurrent layer. Yin et al. [67] presented another attention-based CNN model for sentence pair matching. They examined 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. [68] employed multiple attention functions to match sentence pairs under the matching-aggregation framework. Yang et al. [69] introduced an attention-based neural matching model for ranking short answer texts. They adopted 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 achieved promising results on the TREC QA dataset.

There are other interesting attention models. Lin et al. [70] used self-attention to extract interpretable sentence embeddings. Wang et al. [71] proposed a densely connected CNN with multi-scale feature attention to produce variable n-gram features. Yamada and Shindo [72] used neural attentive bag-of-entities models to perform text classification using entities in a knowledge base. Parikh et al. [73] used attention to decompose a problem into subproblems that can be solved separately. Chen et al. [74] explored generalized pooling methods to enhance sentence embedding, and proposed a vector-based multi-head attention model. Liu and Lane [75] proposed an attention-based RNN model for joint intent detection and slot filling.

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 [76] presented a memory-augmented neural network, called Neural Semantic Encoder (NSE), for text classification 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 Fig. 9.

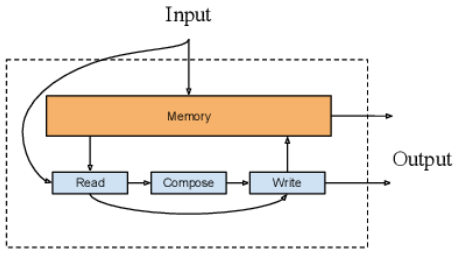

Fig. 9. The architecture of NSE [76].

Weston et al. [77] designed 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. [78] extended this work and proposed end-to-end memory networks, where memory entries are retrieved in a soft manner with attention mechanism, thus enabling end-to-end training. They showed 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. [79] proposed a Dynamic Memory Network (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 obtained state-of-the-art results on QA and POS tagging. Xiong et al. [80] presented a detailed analysis of the DMN, and improved its memory and input modules.

2.7 Transformers

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 increasing length of the sentence, similar to RNNs. Transformers [2] 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. 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 GPU clusters.

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 [81] or LSTMs [82], Transformer-based PLMs use much deeper network architectures (e.g., 48-layer Transformers [83]), 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 have been *fine-tuned* using task-specific labels, and created new state-of-the-art in many downstream NLP tasks, including text classification. Although pre-training is unsupervised, fine-tuning is supervised learning.

PLMs can be grouped into two categories, autoregressive and autoencoding PLMs. One of the earliest autoregressive PLMs is OpenGPT [83, 84], a unidirectional model which predicts a text sequence word by word from left to right (or right to left), with each word prediction depending on previous predictions. Fig. 10 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 text classification by adding task-specific linear classifiers and fine-tuning using task-specific labels.

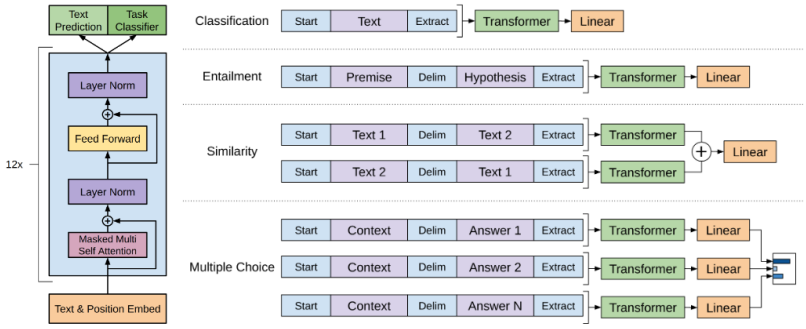

Fig. 10. The architecture of OpenGPT-1 [83]

One of the most widely used autoencoding PLMs is BERT [4]. Unlike OpenGPT which predicts words based on previous predictions, BERT is trained using the masked language modeling 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 [85] is more robust than BERT, and is trained using much more training data. ALBERT [86] lowers the memory consumption and increases the training speed of BERT. DistillBERT [87] 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 [88] extends BERT to better represent and predict text spans. BERT and its variants have been fine-tuned for various NLP tasks, including QA [89], text classification [90], and NLI [91, 92].

There have been attempts to combine the strengths of autoregressive and autoencoding PLMs. XLNet [5] 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 Fig. 11, 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.

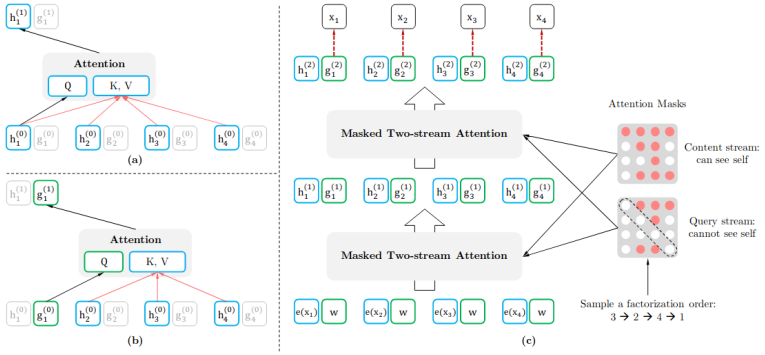

Fig. 11. The architecture of XLNet [5]: a) Content stream attention, b) Query stream attention, c) Overview of the permutation language modeling training with two- stream attention.

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 language understanding. The Unified language Model (UniLM) [93] 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 Fig. 12. The second version of UniLM [94] 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. [95] presented a unified Transformer-based framework that converts many NLP problems into a text-to-text format. They also conducted a systematic study to compare pre-training objectives, architectures, unlabeled datasets, fine-tuning approaches, and other factors on dozens of language understanding tasks.

2.8 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/semantic relations among words in sentences.

One of the earliest graph-based models developed for NLP is TextRank [96]. The authors proposed 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, etc. Similarly, edges can be used to represent different types of relations between any nodes, e.g., lexical or semantic relations, contextual overlap, etc.

Modern Graph Neural Networks (GNNs) are developed by extending deep learning approaches for graph data, such as the text graphs used by TextRank. Deep neural networks, such as CNNs, RNNs and autoencoders,

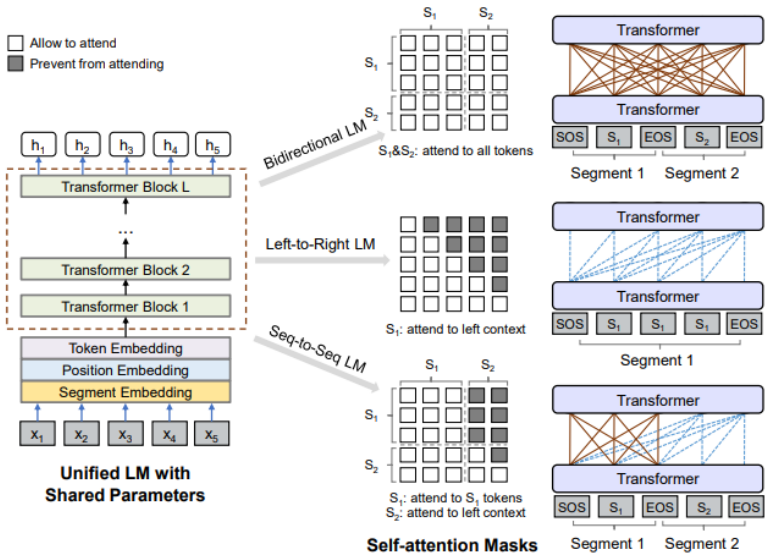

Fig. 12. Overview of UniLM pre-training [93]. The model parameters are shared across the language modeling objectives i.e., bidirectional, unidirectional, and sequence-to-sequence language modeling. Different self-attention masks are used to control the access to context for each word token.

have been generalized over the last few years to handle the complexity of graph data [97]. 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) [98] 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 text classification. GNNs utilize the inter-relations of documents or words to infer document labels [98–100]. In what follows, we review some variants of GCNs that are developed for text classification.

Peng et al. [101] proposed a graph-CNN based deep learning model to first convert text to graph-of-words, and then use graph convolution operations to convolve the word graph, as shown in Fig. 13. They showed 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 [102], Peng et al. proposed a text classification model based on hierarchical taxonomy-aware and attentional graph capsule CNNs. One unique feature of the model is its use of the hierarchical relations among the class labels, which in previous methods are considered independent. Specifically, to leverage such relations, the authors developed a hierarchical taxonomy embedding method to learn their representations, and defined a novel weighted margin loss by incorporating the label representation similarity.

Yao et al. [103] used a similar Graph CNN (GCNN) model for text classification. They built a single text graph for a corpus based on word co-occurrence and document word relations, then learned a Text Graph Convolutional Network (Text GCN) for the corpus, as shown in Fig. 14. The Text GCN is initialized with one-hot representation

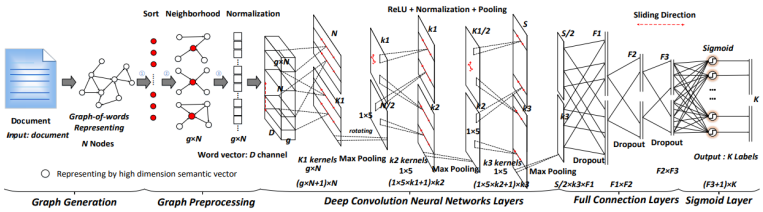

Fig. 13. The architecture of GNN used by Peng et al. [101].

for word and document, and then jointly learns the embeddings for both words and documents, as supervised by the known class labels for documents.

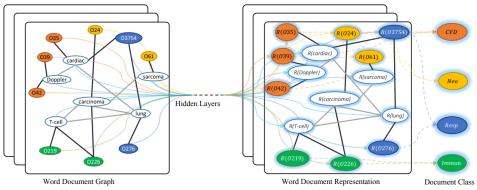

Fig. 14. The architecture of GCNN [103].

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 [104], where a deep convolutional GNN is simplified by repeatedly removing the non-linearities 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 [105]. 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 so as to reduce the memory consumption during training. The same idea motivates the development of GraphSage [99] — a batch-training algorithm for convolutional GNNs.

2.9 Siamese Neural Networks

Siamese neural networks (S2Nets) [106, 107] and their DNN variants, known as Deep Structured Semantic Models (DSSMs) [108], are designed for text matching. The task is fundamental to many NLP applications, such as query-document ranking and answer selection in QA. These tasks can be viewed as special cases of text classification. For example, in question-document ranking, we want to classify a document as relevant or irrelevant to a given query.

As illustrated in Fig. 15, 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. 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,

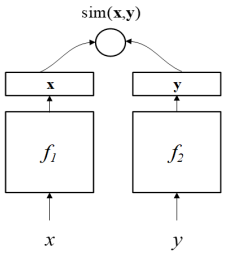

Fig. 15. The architecture of a DSSM

(x, y) could be a query-document pair for query-document ranking [108, 109], or a question-answer pair in QA [110, 111], and so on.

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. Fig. 16 shows the architecture of the siamese model proposed by Mueller et al. [112], where the two networks use the same LSTM model. Neculoiu et al. [113] presented a similar model that uses character-level Bi-LSTMs for f_1 and f_2, and the cosine function to calculate the similarity. In addition to RNNs, BOW models and CNNs are also used in S2Nets to represent sentences. For example, He et al. [114] proposed a S2Net that uses CNNs to model multi-perspective sentence similarity. Renter et al. [115] proposed 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 [116] and TwinBERT [117].

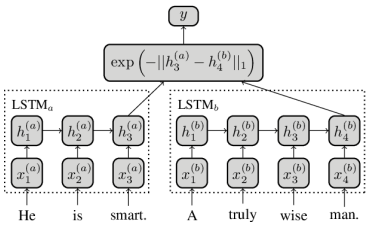

Fig. 16. The architecture of the Siamese model proposed by Mueller et al. [112].

S2Nets and DSSMs have been widely used for QA. Das et al. [110] proposed a Siamese CNN for Question Answer (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. [111] presented a series of Siamese neural networks for answer selection. As shown in Fig. 17, 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 [118], Hyperbolic representation learning [119], and question-answering using a deep similarity neural network [120].

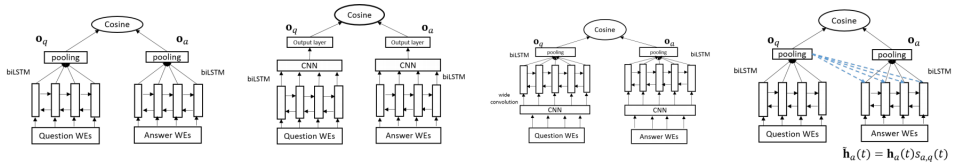

Fig. 17. The architectures of the Siamese models studied in [111].

2.10 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. [121] proposed a Convolutional LSTM (C-LSTM) network. As illustrated in Fig. 18 (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. [122] proposed a Dependency Sensitive CNN (DSCNN) for document modeling. As illustrated in Fig. 18 (b), the DSCNN is a hierarchical model, where LSTM learns the sentence vectors which are fed to the convolution and max-pooling layers to generate the document representation.

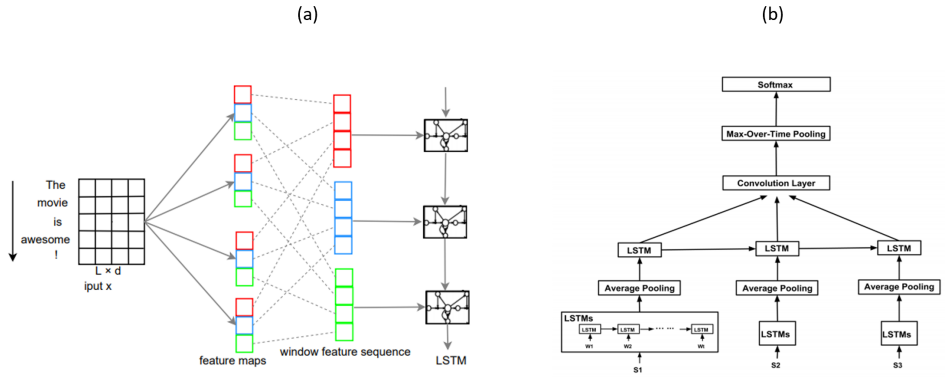

Fig. 18. (a) The architecture of C-LSTM [121]. (b) The architecture of DSCNN for document modeling [122].

Chen et al. [123] performed multi-label text categorization 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. [124] used 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. [125] viewed 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 [126] applied 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. [127] proposed 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 [128], Kowsari et al. proposed a Hierarchical Deep Learning approach for Text classification (HDLTex). HDLTex employs stacks of hybrid deep learning model architectures, including MLP, RNN and CNN, to provide specialized understanding at each level of the document hierarchy.

Liu [129] proposed a robust Stochastic Answer Network (SAN) for multi-step reasoning in machine reading comprehension. As illustrated in Fig. 19, 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 (GRU) based answer module outputs predictions.

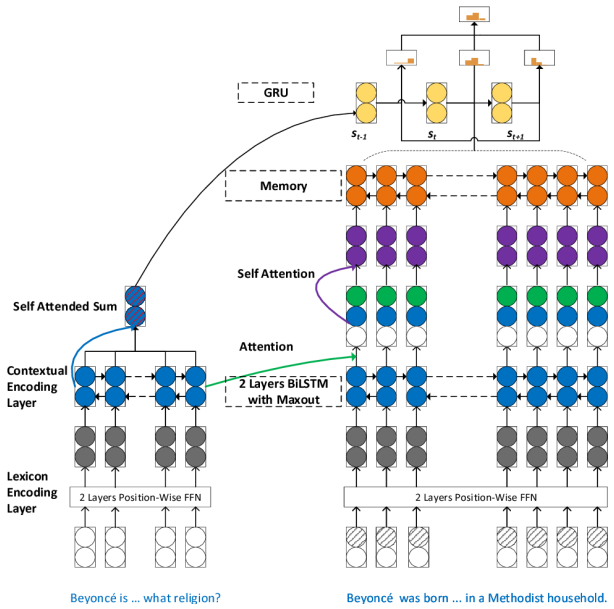

Fig. 19. The architecture of the stochastic answer network [129].

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 [130] 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 [3]. Kim et al. [131] employed a highway network with CNN and LSTM over characters for language modeling. As illustrated in Fig. 20, the first layer performs a lookup of character embeddings, then convolution 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 [132], and RNN with highway [133].

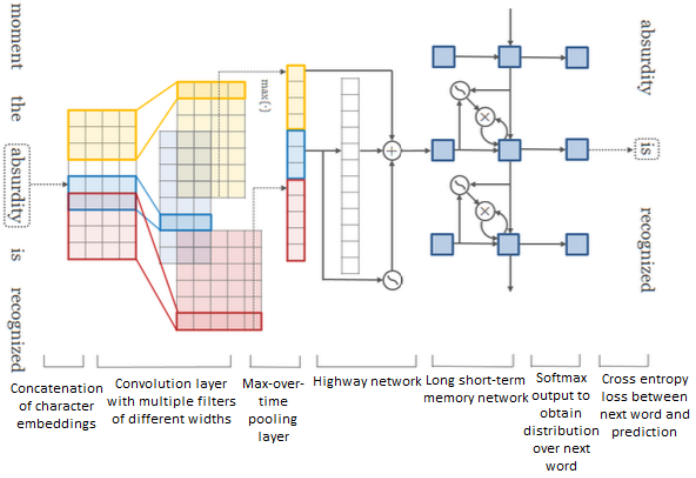

Fig. 20. The architecture of the highway network with CNN and LSTM [131].

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 [134]. 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 2.7 are also unsupervised models that can be used as sentence encoders. This section discusses unsupervised models based on auto-encoders and its variants.

Kiros et al. [135] proposed 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 [136] investigated the use of a sequence autoencoder, which reads the input sequence into a vector and predicts the input again, for sentence encoding. They showed that pre-training sentence encoders on a large unsupervised corpus yields better accuracy than only pre-training word embeddings. Zhang et al. [137] proposed 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) [138, 139] learn a distribution representing the data, and can be viewed as a regularized version of the autoencoder [140]. Since a VAE learns to model the data, we can easily sample from the distribution to generate new input data samples (e.g., new sentences). Miao et al. [141] extended the VAE framework to text, and proposed a Neural Variational Document Model (NVDM) for document modeling and a Neural Answer Selection Model (NASM) for QA. As shown in Fig. 21 (a), the NVDM uses an MLP encoder to map a document to a continuous semantic

representation. As shown in Fig. 21 (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. [142] proposed an RNN-based VAE language model, as shown in Fig. 21 (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.

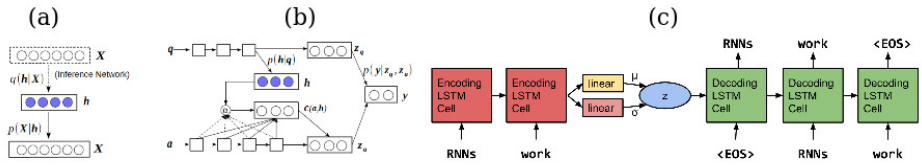

Fig. 21. (a) The neural variational document model for document modeling [141]. (b) The neural answer selection model for QA [141]. (c) The RNN-based variational autoencoder language model [142].

Adversarial Training. Adversarial training [143] 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 [144] extended 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. [145] extended adversarial and virtual adversarial training to supervised and semi-supervised text classification tasks by applying perturbations to the word embeddings in an RNN rather than the original input itself. Sachel et al. [146] studied LSTM models for semi-supervised text classification. They found 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. [147] extended adversarial training to the multi-task learning framework for text classification [18], aiming to alleviate the task-independent (shared) and task-dependent (private) latent feature spaces from interfering with each other.

Reinforcement Learning. Reinforcement learning (RL) [148] 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. [149] used a hard attention model to select a subset of critical word tokens of an input sequence for text classification. 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. [150] proposed a neural agent that models text classification 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. [151] presented 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. [152] combined 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. [153] proposed a RL-based method of learning structured representations for text classification. They proposed 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 Fig. 22. The policy network is optimized using policy gradient.

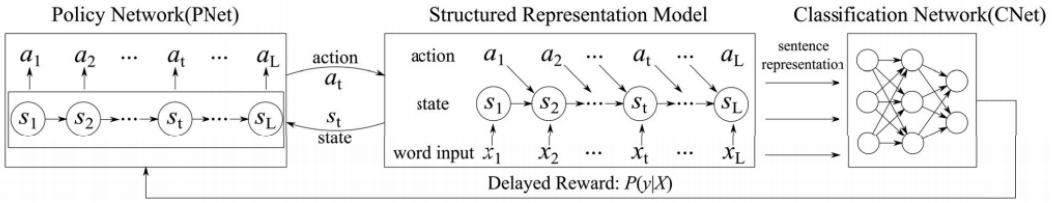

Fig. 22. The RL-based method of learning structured representations for text classification [153]. The policy network samples an action at each state. The structured representation model updates the state and outputs the final sentence representation to the classification network at the end of the episode. The text classification loss is used as a (negative) reward to train the policy.

3 TEXT CLASSIFICATION DATASETS

In this section, we describe some of the datasets that are widely used for text classification research. We group these datasets, based on their main target applications, into such categories as sentiment analysis, news categorization, topic classification, QA, and NLI.

3.1 Sentiment Analysis Datasets

Yelp. Yelp [154] is one of the most popular datasets for sentiment classification. Two classification tasks are defined on this dataset. 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. Fig. 23 shows the word cloud presentation of this dataset.

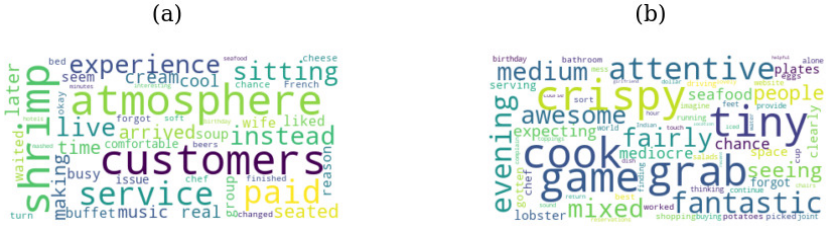

Fig. 23. Word cloud presentation of Yelp (a): Yelp Review Full or Yelp-5. (b): Yelp Review Polarity or Yelp-2.

IMDb. The IMDB dataset [155] 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 [156] 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. 10-fold cross validation with random split is usually used for testing on this dataset.

SST. The Stanford Sentiment Treebank (SST) dataset [157] 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 [158] 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 [159]. 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.

Aspect-Based Sentiment Analysis. Besides the above datasets, there are also several datasets proposed for the task of aspect-level sentiment analysis [160]. Some of the most popular datasets include SemEval-2014 Task 4 [161], Twitter [162], SentiHood [163], to name a few.

3.2 News Classification Datasets

AG News. The AG News dataset [31] 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 [164] 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.

Fig. 24. Word cloud presentation of two different news datasets. (a): AG News dataset. (b): Newsgroups dataset.

Sogou News. The Sogou News dataset [90] 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 [165] is one of the most widely used data collections for text categorization research. It 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. There are many other datasets derived from different subsets of the Reuters dataset, such as R8, R52, RCV1, and RCV1-v2.

Other datasets developed for news categorization includes: Bing news [166], NYTimes [167], BBC [168], Google news [169], to name a few.

3.3 Topic Classification Datasets

DBpedia. The DBpedia dataset [170] 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 [171] 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. Fig. 25 provides the word cloud representations of Ohsumed and DBpedia datasets.

Fig. 25. Word cloud representation of two different topic classification dataset. (a): Ohsumed dataset. (b): DBpedia dataset

EUR-Lex. The EUR-Lex dataset [172] 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 [128] 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 [173] 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 [174], Irony (which is composed of annotated comments from the social news website reddit, Twitter dataset for topic classification of tweets, arXiv collection) [175], to name a few.

3.4 QA Datasets

SQuAD. Stanford Question Answering Dataset (SQuAD) [6] 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 [176].

MS MARCO. This dataset is released by Microsoft [177]. 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. There have been various versions of MS MARCO used for extractive QA, passage ranking, etc.

TREC-QA. TREC-QA [178] 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 [179] 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 [180] 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) [181], WikiQA [179], SelQA [182], to name a few.

3.5 NLI Datasets

SNLI. The Stanford Natural Language Inference (SNLI) dataset [183] 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 [184] 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 [7] consists of about 10,000 English sentence pairs which are annotated with three labels: entailment, contradiction, and neutral.

MSRP. The Microsoft Research Paraphrase (MSRP) dataset [185] 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) [186], RTE [187], SciTail [188], to name a few.

Fig. 26 presents a set of text classification datasets, sorted based on their sizes.

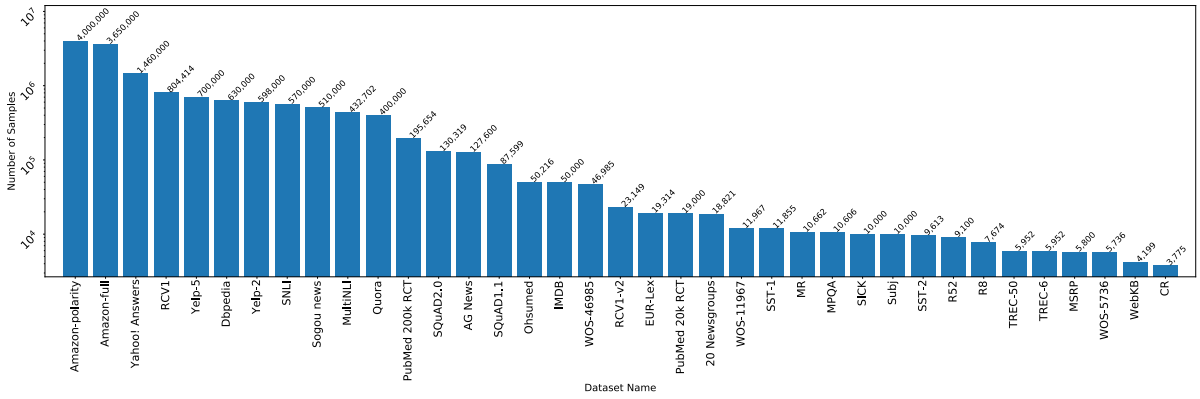

Fig. 26. An illustration of size of different text classification datasets. Dataset sizes are shown in log-scale for better visualization.

4 EXPERIMENTAL PERFORMANCE ANALYSIS

In this section, we first describe a set of metrics commonly used for evaluating text classification models' performance, and then present a quantitative analysis of the performance of a set of deep learning based text classification models on popular datasets.

4.1 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 Eq. 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 Eq. 3. The F1 score is the harmonic mean of the precision and recall, as in Eq. 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 Eq. 4, where Q is a set of all possible answers, and $rank_i$ is the ranking position of the ground-truth answer.

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

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

4.2 Quantitative Results

In this section, we tabulate the performance of several of the previously discussed algorithms on popular text classification benchmarks. In each table, in addition to the results of a set of representative deep learning models, we also present results using non-deep-learning models which are either previous state-of-the-art or widely used as baselines before the deep learning era. We can see that across all these tasks, the use of deep learning 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 some significant improvement in accuracy has been obtained since the introduction of the first deep learning 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.

Table 3 and Table 4 present the performance of some deep learning 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 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.

5 CHALLENGES AND OPPORTUNITIES

Text classification has seen a great progress over the last few years, with the help of deep learning based models. Several novel ideas have been proposed (such as neural embedding, attention mechanism, self attention, Transformer, BERT, and XLNet), which led to the fast progress over the past decade. Despite all the progress, there are still several challenges ahead of us that need to be solved. This section presents some of these challenges, and discusses research directions that we believe will help advance the field.

New Datasets for More Challenging Tasks. Although a number of large-scale datasets have been collected for common text classification tasks in recent years, there remains a need for new datasets for more challenging tasks such as QA with multi-step reasoning and text classification for multi-lingual documents. Having a large-scale labeled dataset for these tasks can help to accelerate the progress in these areas.

Modeling Commonsense Knowledge. Incorporating commonsense knowledge into deep learning 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 [202], much more research is required to explore how to effectively model and use commonsense knowledge in neural models.

Table 1. Performance 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. Italic indicates the non-deep-learning models.

Method	IMDB	SST-2	Amazon-2	Amazon-5	Yelp-2	Yelp-5
Naive Bayes [157]	-	81.8	-	-	-	-
LDA [189]	67.4	-	-	-	-	-
BoW+SVM [12]	87.8	-	-	-	-	-
tf.Δ idf [190]	88.1	-	-	-	-	-
Char-level CNN [31]	-	-	94.49	59.46	95.12	62.05
Deep Pyramid CNN [30]	-	84.46	96.68	65.82	97.36	69.40
ULMFiT [191]	95.4	-	-	-	97.84	70.02
BLSTM-2DCNN [22]	-	89.5	-	-	-	-
Neural Semantic Encoder [76]	-	89.7	-	-	-	-
BCN+Char+CoVe [192]	91.8	90.3	-	-	-	-
GLUE ELMo baseline [193]	-	90.4	-	-	-	-
BERT ELMo baseline [4]	-	90.4	-	-	-	-
CCCapsNet [57]	-	-	94.96	60.95	96.48	65.85
Virtual adversarial training [145]	94.1	-	-	-	-	-
Block-sparse LSTM [194]	94.99	93.2	-	-	96.73	
BERT-base [4, 90]	95.63	93.5	96.04	61.6	98.08	70.58
BERT-large [4, 90]	95.79	94.9	96.07	62.2	98.19	71.38
ALBERT [86]	-	95.2	-	-	-	-
Multi-Task DNN [92]	83.2	95.6	-	-	-	-
Snorkel MeTaL [195]	-	96.2	-	-	-	-
BERT Finetune + UDA [196]	95.8		96.5	62.88	97.95	62.92
RoBERTa (+additional data) [85]	-	96.4	-	-	-	-
XLNet-Large (ensemble) [5]	96.21	96.8	97.6	67.74	98.45	72.2

Interpretable Deep Learning Models. While deep learning models have achieved promising performance on challenging benchmarks, most of these models are not interpretable and there remain many open questions. For example, why does a model outperform another model on one dataset, but underperform on other datasets? What exactly have deep learning 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 to 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. But these models have to be simplified and compressed in order to meet the computation and storage constraints of mobile devices. This can be done either by building student models using knowledge distillation, or by using model compression techniques. Developing a task-agnostic model simplification method is an active research topic [203].

Few-Shot and Zero-Shot Learning. Most deep learning models are supervised models that require large amounts of domain labels. In practice, it is expensive to collect such labels for each new domain. Finetuning a

Table 2. Performance of classification models on news categorization, and topic classification tasks. Italic indicates the non-deep-learning models.

	News Categorization			**Topic Classification**	
Method	AG News	20NEWS	Sogou News	DBpedia	Ohsumed
Hierarchical Log-bilinear Model [197]	-	-	-	-	52
Text GCN [103]	67.61	86.34	-	-	68.36
Simplfied GCN [104]	-	88.5	-	-	68.50
Char-level CNN [31]	90.49	-	95.12	98.45	-
CCCapsNet [57]	92.39	-	97.25	98.72	-
LEAM [65]	92.45	81.91	-	99.02	58.58
fastText [11]	92.5	-	96.8	98.6	55.7
CapsuleNet B [52]	92.6	-	-	-	-
Deep Pyramid CNN [30]	93.13	-	98.16	99.12	-
ULMFiT [191]	94.99	-	-	99.2	-
L MIXED [146]	95.05	-	-	99.3	-
BERT-large [196]	-	-	-	99.32	-
XLNet [5]	95.51	-	-	99.38	-

Table 3. Performance of classification models on SQuAD question answering datasets. Here, the F1 score measures the average overlap between the prediction and ground truth answer. Italic denotes the non-deep-learning models.

	SQuAD1.1		SQuAD2.0	
Method	EM	F1-score	EM	F1-score
Sliding Window+Dist. [198]	13.00	20.00	-	-
Hand-crafted Features+Logistic Regression [6]	40.40	51.00	-	-
BiDAF + Self Attention + ELMo [82]	78.58	85.83	63.37	66.25
SAN (single model) [129]	76.82	84.39	68.65	71.43
FusionNet++ (ensemble) [199]	78.97	86.01	70.30	72.48
SAN (ensemble) [129]	79.60	86.49	71.31	73.70
BERT (single model) [4]	85.08	91.83	80.00	83.06
BERT-large (ensemble) [4]	87.43	93.16	80.45	83.51
BERT + Multiple-CNN [129]	-	-	84.20	86.76
XL-Net [5]	89.90	95.08	84.64	88.00
SpanBERT [88]	88.83	94.63	71.31	73.70
RoBERTa [85]	-	-	86.82	89.79
ALBERT (single model) [86]	-	-	88.10	90.90
ALBERT (ensemble) [86]	-	-	89.73	92.21
Retro-Reader on ALBERT	-	-	90.11	92.58

pre-trained language model (PLM), such as BERT and OpenGPT, to a specific task requires many 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.

Table 4. Performance of classification models on the WikiQA datasets.

Method	MAP	MRR
Paragraph vector [13]	0.511	0.516
Neural Variational Inference [141]	0.655	0.674
Attentive pooling networks [64]	0.688	0.695
HyperQA [119]	0.712	0.727
BERT (single model) [4]	0.813	0.828
TANDA-RoBERTa [89]	0.920	0.933

Table 5. Performance of classification models on natural language inference datasets. For Multi-NLI, Matched and Mismatched refer to the matched and mismatched test accuracies, respectively. Italic denotes the non-deep-learning models.

	SNLI	MultiNLI	
Method	Accuracy	Matched	Mismatched
Unigrams Features [183]	71.6	-	-
Lexicalized [183]	78.2	-	-
LSTM encoders (100D) [183]	77.6	-	-
Tree Based CNN [42]	82.1	-	-
biLSTM Encoder [184]	81.5	67.5	67.1
Neural Semantic Encoders (300D) [76]	84.6	-	-
RNN Based Sentence Encoder [200]	85.5	73.2	73.6
DiSAN (300D) [62]	85.6	-	-
Decomposable Attention Model [73]	86.3	-	-
Reinforced Self-Attention (300D) [149]	86.3	-	-
Generalized Pooling (600D) [74]	86.6	73.8	74.0
Bilateral multi-perspective matching [23]	87.5	-	-
Multiway Attention Network [68]	88.3	78.5	77.7
ESIM + ELMo [82]	88.7	72.9	73.4
DMAN with Reinforcement Learning [201]	88.8	88.8	78.9
BiLSTM + ELMo + Attn [193]	-	74.1	74.5
Fine-Tuned LM-Pretrained Transformer [84]	89.9	82.1	81.4
Multi-Task DNN [92]	91.6	86.7	86.0
SemBERT [91]	91.9	84.4	84.0
RoBERTa [85]	92.6	90.8	90.2
XLNet [5]	-	90.2	89.8

6 CONCLUSION

In this paper, we survey more than 150 deep learning models, which are developed in the past 6 to 7 years and have significantly improved state-of-the-art on various text classification tasks, including sentiment analysis, news categorization, topic classification, QA, and NLI. We also provide an overview of more than 40 popular text classification datasets, and 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 would like to thank Richard Socher, Kristina Toutanova, and Brooke Cowan for reviewing this work, and providing very insightful comments.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[5] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5754–5764.
[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.
[7] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment," in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, 2014, pp. 1–8.
[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[9] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
[10] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "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 (Volume 1: Long Papers)*, 2015, pp. 1681–1691.
[11] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext. zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.
[12] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*. Association for Computational Linguistics, 2012, pp. 90–94.
[13] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, 2014, pp. 1188–1196.
[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
[15] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.
[16] X. Zhu, P. Sobihani, and H. Guo, "Long short-term memory over recursive structures," in *International Conference on Machine Learning*, 2015, pp. 1604–1612.
[17] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.
[18] P. Liu, X. Qiu, X. Chen, S. Wu, and X.-J. Huang, "Multi-timescale long short-term memory neural network for modelling sentences and documents," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 2326–2335.
[19] A. B. Dieng, C. Wang, J. Gao, and J. Paisley, "Topicrnn: A recurrent neural network with long-range semantic dependency," *arXiv preprint arXiv:1611.01702*, 2016.
[20] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.
[21] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using lstm for region embeddings," *arXiv preprint arXiv:1602.02373*, 2016.
[22] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *arXiv preprint arXiv:1611.06639*, 2016.
[23] Z. Wang, W. Hamza, and R. Florian, "Bilateral multi-perspective matching for natural language sentences," *arXiv preprint arXiv:1702.03814*, 2017.

[24] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[26] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 2014.
[27] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.
[28] J. Liu, W. C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.
[29] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," in *NAACL HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, 2015.
[30] ——, "Deep pyramid convolutional neural networks for text categorization," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 562–570.
[31] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.
[32] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[33] J. D. Prusa and T. M. Khoshgoftaar, "Designing a better data representation for deep neural networks and text classification," in *Proceedings - 2016 IEEE 17th International Conference on Information Reuse and Integration, IRI 2016*, 2016.
[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.
[36] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*, 2016.
[37] A. B. Duque, L. L. J. Santos, D. Macêdo, and C. Zanchettin, "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)*, 2019.
[38] H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?" in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.
[40] B. Guo, C. Zhang, J. Liu, and X. Ma, "Improving text classification with weighted word embeddings via a multi-channel TextCNN model," *Neurocomputing*, 2019.
[41] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.
[42] L. Mou, R. Men, G. Li, Y. Xu, L. Zhang, R. Yan, and Z. Jin, "Natural language inference by tree-based convolution and heuristic matching," *arXiv preprint arXiv:1512.08422*, 2015.
[43] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng, "Text matching as image recognition," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016.
[44] J. Wang, Z. Wang, D. Zhang, and J. Yan, "Combining knowledge with deep convolutional neural networks for short text classification," in *IJCAI International Joint Conference on Artificial Intelligence*, 2017.
[45] S. Karimi, X. Dai, H. Hassanzadeh, and A. Nguyen, "Automatic Diagnosis Coding of Radiology Reports: A Comparison of Deep Learning and Conventional Classification Methods," 2017.
[46] S. Peng, R. You, H. Wang, C. Zhai, H. Mamitsuka, and S. Zhu, "DeepMeSH: Deep semantic representation for improving large-scale MeSH indexing," *Bioinformatics*, 2016.
[47] A. Rios and R. Kavuluru, "Convolutional neural networks for biomedical text classification: Application in indexing biomedical articles," in *BCB 2015 - 6th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2015.
[48] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, "Medical Text Classification Using Convolutional Neural Networks," *Studies in Health Technology and Informatics*, 2017.
[49] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International conference on artificial neural networks*. Springer, 2011, pp. 44–51.
[50] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in neural information processing systems*, 2017, pp. 3856–3866.

[51] S. Sabour, N. Frosst, and G. Hinton, "Matrix capsules with em routing," in *6th international conference on learning representations, ICLR*, 2018, pp. 1–15.
[52] W. Zhao, J. Ye, M. Yang, Z. Lei, S. Zhang, and Z. Zhao, "Investigating capsule networks with dynamic routing for text classification," *arXiv preprint arXiv:1804.00538*, 2018.
[53] M. Yang, W. Zhao, L. Chen, Q. Qu, Z. Zhao, and Y. Shen, "Investigating the transferring capability of capsule networks for text classification," *Neural Networks*, vol. 118, pp. 247–261, 2019.
[54] W. Zhao, H. Peng, S. Eger, E. Cambria, and M. Yang, "Towards scalable and reliable capsule networks for challenging NLP applications," in *ACL*, 2019, pp. 1549–1559.
[55] J. Kim, S. Jang, E. Park, and S. Choi, "Text classification using capsules," *Neurocomputing*, vol. 376, pp. 214–221, 2020.
[56] R. Aly, S. Remus, and C. Biemann, "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*, 2019, pp. 323–330.
[57] H. Ren and H. Lu, "Compositional coding capsule network with k-means routing for text classification," *arXiv preprint arXiv:1810.09177*, 2018.
[58] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[59] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.
[60] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.
[61] X. Zhou, X. Wan, and J. Xiao, "Attention-based lstm network for cross-lingual sentiment classification," in *Proceedings of the 2016 conference on empirical methods in natural language processing*, 2016, pp. 247–256.
[62] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, "Disan: Directional self-attention network for rnn/cnn-free language understanding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[63] Y. Liu, C. Sun, L. Lin, and X. Wang, "Learning natural language inference using bidirectional lstm model and inner-attention," *arXiv preprint arXiv:1605.09090*, 2016.
[64] C. d. Santos, M. Tan, B. Xiang, and B. Zhou, "Attentive pooling networks," *arXiv preprint arXiv:1602.03609*, 2016.
[65] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint embedding of words and labels for text classification," *arXiv preprint arXiv:1805.04174*, 2018.
[66] S. Kim, I. Kang, and N. Kwak, "Semantic sentence matching with densely-connected recurrent and co-attentive information," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6586–6593.
[67] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259–272, 2016.
[68] C. Tan, F. Wei, W. Wang, W. Lv, and M. Zhou, "Multiway attention networks for modeling sentence pairs," in *IJCAI*, 2018, pp. 4411–4417.
[69] L. Yang, Q. Ai, J. Guo, and W. B. Croft, "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*, 2016, pp. 287–296.
[70] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130*, 2017.
[71] S. Wang, M. Huang, and Z. Deng, "Densely connected cnn with multi-scale feature attention for text classification." in *IJCAI*, 2018, pp. 4468–4474.
[72] I. Yamada and H. Shindo, "Neural attentive bag-of-entities model for text classification," *arXiv preprint arXiv:1909.01259*, 2019.
[73] A. P. Parikh, O. Tackstrom, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," *arXiv preprint arXiv:1606.01933*, 2016.
[74] Q. Chen, Z.-H. Ling, and X. Zhu, "Enhancing sentence embedding with generalized pooling," *arXiv preprint arXiv:1806.09828*, 2018.
[75] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," *arXiv preprint arXiv:1609.01454*, 2016.
[76] T. Munkhdalai and H. Yu, "Neural semantic encoders," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 1. NIH Public Access, 2017, p. 397.
[77] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
[78] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.
[79] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *33rd International Conference on Machine Learning, ICML 2016*, 2016.

[80] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," in *33rd International Conference on Machine Learning, ICML 2016*, 2016.
[81] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.
[82] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.
[83] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[84] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.
[85] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[86] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.
[87] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
[88] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *arXiv preprint arXiv:1907.10529*, 2019.
[89] S. Garg, T. Vu, and A. Moschitti, "Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection," *arXiv preprint arXiv:1911.04118*, 2019.
[90] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" in *China National Conference on Chinese Computational Linguistics*. Springer, 2019, pp. 194–206.
[91] Z. Zhang, Y. Wu, H. Zhao, Z. Li, S. Zhang, X. Zhou, and X. Zhou, "Semantics-aware bert for language understanding," *arXiv preprint arXiv:1909.02209*, 2019.
[92] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," *arXiv preprint arXiv:1901.11504*, 2019.
[93] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 042–13 054.
[94] H. Bao, L. Dong, F. Wei, W. Wang, N. Yang, X. Liu, Y. Wang, S. Piao, J. Gao, M. Zhou *et al.*, "Unilmv2: Pseudo-masked language models for unified language model pre-training," *arXiv preprint arXiv:2002.12804*, 2020.
[95] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.
[96] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 404–411.
[97] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.
[98] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[99] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.
[100] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
[101] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang, "Large-scale hierarchical text classification with recursively regularized deep graph-cnn," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 1063–1072.
[102] H. Peng, J. Li, Q. Gong, S. Wang, L. He, B. Li, L. Wang, and P. S. Yu, "Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification," *arXiv preprint arXiv:1906.04898*, 2019.
[103] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7370–7377.
[104] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.
[105] L. Huang, D. Ma, S. Li, X. Zhang, and H. WANG, "Text level graph neural network for text classification," *arXiv preprint arXiv:1910.02356*, 2019.
[106] J. BROMLEY, J. W. BENTZ, L. BOTTOU, I. GUYON, Y. LECUN, C. MOORE, E. SÄCKINGER, and R. SHAH, "Signature verification using a Siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, 1993.

[107] W. tau Yih, K. Toutanova, J. C. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *CoNLL 2011 - Fifteenth Conference on Computational Natural Language Learning, Proceedings of the Conference*, 2011.
[108] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 101–110.
[109] A. Severyn and A. Moschittiy, "Learning to rank short text pairs with convolutional deep neural networks," in *SIGIR 2015 - Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.
[110] A. Das, H. Yenala, M. Chinnakotla, and M. Shrivastava, "Together we stand: Siamese networks for similar question retrieval," in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2016.
[111] M. Tan, C. D. Santos, B. Xiang, and B. Zhou, "Improved representation learning for question answer matching," in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2016.
[112] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016.
[113] P. Neculoiu, M. Versteegh, and M. Rotaru, "Learning Text Similarity with Siamese Recurrent Networks," 2016.
[114] H. He, K. Gimpel, and J. Lin, "Multi-perspective sentence similarity modeling with convolutional neural networks," in *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, 2015.
[115] T. Renter, A. Borisov, and M. De Rijke, "Siamese CBOW: Optimizing word embeddings for sentence representations," in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2016.
[116] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," 2019.
[117] W. Lu, J. Jiao, and R. Zhang, "Twinbert: Distilling knowledge to twin-structured bert models for efficient retrieval," *arXiv preprint arXiv:2002.06275*, 2020.
[118] M. Tan, C. d. Santos, B. Xiang, and B. Zhou, "Lstm-based deep learning models for non-factoid answer selection," *arXiv preprint arXiv:1511.04108*, 2015.
[119] Y. Tay, L. A. Tuan, and S. C. Hui, "Hyperbolic representation learning for fast and efficient neural question answering," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 583–591.
[120] S. Minaee and Z. Liu, "Automatic question-answering using a deep similarity neural network," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 923–927.
[121] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.
[122] R. Zhang, H. Lee, and D. Radev, "Dependency sensitive convolutional neural networks for modeling sentences and documents," in *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, 2016.
[123] G. Chen, D. Ye, E. Cambria, J. Chen, and Z. Xing, "Ensemble application of convolutional and recurrent neural networks for multi-label text categorization," in *IJCNN*, 2017, pp. 2377–2383.
[124] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422–1432.
[125] Y. Xiao and K. Cho, "Efficient character-level document classification by combining convolution and recurrent layers," *arXiv preprint arXiv:1602.00367*, 2016.
[126] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.
[127] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using bilstm-crf and cnn," *Expert Systems with Applications*, vol. 72, pp. 221 – 230, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416305929
[128] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 364–371.
[129] X. Liu, Y. Shen, K. Duh, and J. Gao, "Stochastic answer networks for machine reading comprehension," *arXiv preprint arXiv:1712.03556*, 2017.
[130] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems*, 2015.
[131] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-Aware neural language models," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016.
[132] J. G. Zilly, R. K. Srivastava, J. Koutnik, and J. Schmidhuber, "Recurrent highway networks," in *34th International Conference on Machine Learning, ICML 2017*, 2017.
[133] Y. Wen, W. Zhang, R. Luo, and J. Wang, "Learning text representation using recurrent convolutional neural network with highway layers," *arXiv preprint arXiv:1606.06905*, 2016.
[134] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[135] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.
[136] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Advances in Neural Information Processing Systems*, 2015.
[137] M. Zhang, Y. Wu, W. Li, and W. Li, "Learning Universal Sentence Representations with Mean-Max Attention Autoencoder," 2019.
[138] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.
[139] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *ICML*, 2014.
[140] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[141] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *International conference on machine learning*, 2016.
[142] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, 2016.
[143] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[144] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," in *ICLR*, 2016.
[145] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.
[146] D. S. Sachan, M. Zaheer, and R. Salakhutdinov, "Revisiting lstm networks for semi-supervised text classification via mixed objective function," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6940–6948.
[147] P. Liu, X. Qiu, and X. Huang, "Adversarial multi-task learning for text classification," *arXiv preprint arXiv:1704.05742*, 2017.
[148] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[149] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang, "Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling," *arXiv preprint arXiv:1801.10296*, 2018.
[150] X. Liu, L. Mou, H. Cui, Z. Lu, and S. Song, "Finding decision jumps in text classification," *Neurocomputing*, vol. 371, pp. 177–187, 2020.
[151] Y. Shen, P.-S. Huang, J. Gao, and W. Chen, "Reasonet: Learning to stop reading in machine comprehension," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1047–1055.
[152] Y. Li, Q. Pan, S. Wang, T. Yang, and E. Cambria, "A generative model for category text generation," *Information Sciences*, vol. 450, pp. 301–315, 2018.
[153] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[154] https://www.kaggle.com/yelp-dataset/yelp-dataset.
[155] https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.
[156] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL conference on Empirical methods in natural language processing*, 2002, pp. 79–86.
[157] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.
[158] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language resources and evaluation*, vol. 39, no. 2-3, pp. 165–210, 2005.
[159] https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products.
[160] Y. Ma, H. Peng, and E. Cambria, "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM," in *AAAI*, 2018, pp. 5876–5883.
[161] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. Al-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq *et al.*, "Semeval-2016 task 5: Aspect based sentiment analysis," in *International Workshop on Semantic Evaluation (SemEval)*, 2016.
[162] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu, "Adaptive recursive neural network for target-dependent twitter sentiment classification," in *Proceedings of the 52nd annual meeting of the association for computational linguistics (Short papers)*, 2014, pp. 49–54.
[163] M. Saeidi, G. Bouchard, M. Liakata, and S. Riedel, "Sentihood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods," *arXiv preprint arXiv:1610.03771*, 2016.
[164] http://qwone.com/~jason/20Newsgroups/.
[165] https://martin-thoma.com/nlp-reuters.
[166] F. Wang, Z. Wang, Z. Li, and J.-R. Wen, "Concept-based short text classification and ranking," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 1069–1078.
[167] T. P. Jurka, L. Collingwood, A. E. Boydstun, E. Grossman, and W. van Atteveldt, "Rtexttools: Automatic text classification via supervised learning," *R package version*, vol. 1, no. 9, 2012.
[168] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 2006, pp. 377–384.

[169] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 271–280.
[170] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
[171] http://davis.wpi.edu/xmdv/datasets/ohsumed.html.
[172] E. L. Mencia and J. Fürnkranz, "Efficient pairwise multilabel classification for large-scale problems in the legal domain," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 50–65.
[173] Z. Lu, "Pubmed and beyond: a survey of web tools for searching biomedical literature," *Database*, vol. 2011, 2011.
[174] F. Dernoncourt and J. Y. Lee, "Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts," *arXiv preprint arXiv:1710.06071*, 2017.
[175] B. C. Wallace, L. Kertz, E. Charniak *et al.*, "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 (Volume 2: Short Papers)*, 2014, pp. 512–516.
[176] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," *arXiv preprint:1806.03822*, 2018.
[177] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "Ms marco: a human-generated machine reading comprehension dataset," 2016.
[178] https://cogcomp.seas.upenn.edu/Data/QA/QC/.
[179] Y. Yang, W.-t. Yih, and C. Meek, "Wikiqa: A challenge dataset for open-domain question answering," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 2013–2018.
[180] https://data.quora.com/First-Quora-Dataset-Release-QuestionPairs.
[181] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, "Swag: A large-scale adversarial dataset for grounded commonsense inference," *arXiv preprint arXiv:1808.05326*, 2018.
[182] T. Jurczyk, M. Zhai, and J. D. Choi, "Selqa: A new benchmark for selection-based question answering," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016, pp. 820–827.
[183] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.
[184] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.
[185] B. Dolan, C. Quirk, and C. Brockett, "Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources," in *Proceedings of the 20th international conference on Computational Linguistics*. ACL, 2004, p. 350.
[186] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation," *arXiv preprint arXiv:1708.00055*, 2017.
[187] I. Dagan, O. Glickman, and B. Magnini, "The PASCAL Recognising Textual Entailment Challenge," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.
[188] T. Khot, A. Sabharwal, and P. Clark, "Scitail: A textual entailment dataset from science question answering," in *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018.
[189] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, 2011, pp. 142–150.
[190] J. C. Martineau and T. Finin, "Delta tfidf: An improved feature space for sentiment analysis," in *Third international AAAI conference on weblogs and social media*, 2009.
[191] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.
[192] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems*, 2017, pp. 6294–6305.
[193] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.
[194] S. Gray, A. Radford, and D. P. Kingma, "Gpu kernels for block-sparse weights," *arXiv preprint arXiv:1711.09224*, vol. 3, 2017.
[195] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, "Training complex models with multi-task weak supervision," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4763–4771.
[196] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le, "Unsupervised data augmentation," *arXiv preprint arXiv:1904.12848*, 2019.
[197] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*, 2015, pp. 957–966.
[198] M. Richardson, C. J. Burges, and E. Renshaw, "Mctest: A challenge dataset for the open-domain machine comprehension of text," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 193–203.
[199] H.-Y. Huang, C. Zhu, Y. Shen, and W. Chen, "Fusionnet: Fusing via fully-aware attention with application to machine comprehension," *arXiv preprint arXiv:1711.07341*, 2017.

[200] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Recurrent neural network-based sentence encoder with gated attention for natural language inference," *arXiv preprint arXiv:1708.01353*, 2017.
[201] B. Pan, Y. Yang, Z. Zhao, Y. Zhuang, D. Cai, and X. He, "Discourse marker augmented network with reinforcement learning for natural language inference," *arXiv preprint arXiv:1907.09692*, 2019.
[202] E. Cambria, S. Poria, R. Bajpai, and B. Schuller, "Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives," in *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, 2016, pp. 2666–2677.
[203] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *arXiv preprint arXiv:2002.10957*, 2020.
[204] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st ed. USA: Prentice Hall PTR, 2000.
[205] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
[206] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint:1508.07909*, 2015.
[207] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.
[208] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2018.
[209] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.
[210] http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[211] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
[212] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *arXiv preprint arXiv:2001.05566*, 2020.
[213] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.
[214] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun, and D. Zhang, "Biometric recognition using deep learning: A survey," *arXiv preprint arXiv:1912.00271*, 2019.
[215] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

A DEEP NEURAL NETWORK OVERVIEW

This appendix introduces some of the commonly used deep learning models for NLP, including MLPs, CNNs, RNNs, LSTMs, encoder-decoders, and Transformers. Interested readers are referred to [140] for a comprehensive discussion.

A.1 Neural Language Models and Word Embedding

Language modeling adopts data-driven approaches to capture salient statistical properties of text sequences in natural language, which can later be used to predict future words in a sequence, or to perform slot-filling in related tasks. N-gram models are the simplest statistical language models, which capture the relation between successive tokens. However, these models cannot capture long-distance dependence of tokens which often encodes semantic relations [204]. Therefore, there have been a lot of efforts of developing richer language models, among which one of the most successful is the neural language model [205].

Neural language models learn to represent textual-tokens (such as words) as dense vectors, referred as to word embeddings, in a self-supervised fashion. These learned representations can then be used for various NLP applications. One popular neural language model is word2vec [8], which learns to map the words that come in similar contexts to similar vector representations. The learned word2vec representations also allow for some simple algebraic operations on word embeddings in vector space, as shown in Eq. 5.

$$\text{“}king\text{”} - \text{“}man\text{”} + \text{“}woman\text{”} = \text{“}queen\text{”} \tag{5}$$

Despite its popularity and semantic richness, word2vec suffers from some problems such as out of vocabulary (OOV) extension, inability to capture word morphology and word context. There have been many works trying to improve word2vec model, and depending on the textual units they deal with and whether being context dependent or not, they can be grouped into the following categories:

- Word-Level Embedding
- Subword Embedding
- Contextual Embedding

Word-Level Embedding. Two main categories of word-level embedding models are prediction-based and count-based models. The models in the former category are trained to recover the missing tokens in a token sequence. Word2vec is an early example of this category, which proposed two architectures for word embedding, Continuous Bag of Words (CBOW) and Skip-Gram [8, 14], as shown in Fig. 27. A Skip-Gram model predicts

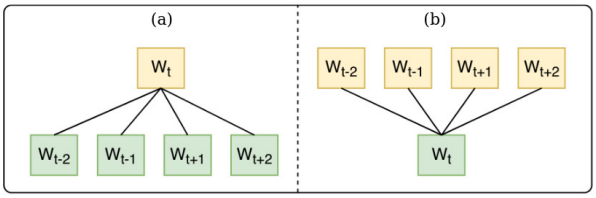

Fig. 27. Two word2vec models [8] (a) CBOW (b) Skip-Gram

each context word from the central word, while a CBOW model predicts the central word based on its context words. The training objectives of these model are to maximize the prediction probability of the correct words.

For example, the training objectives of CBOW and Skip-Gram are shown in Eq. 6 and Eq. 7, respectively.

$$\mathcal{L}_{CBOW} = -\frac{1}{|C| - C} \sum_{k=C+1}^{|C|-C} \log P(w_k | w_{k-C}, \ldots, w_{k-1}, w_{k+1}, \ldots, w_{k+c}) \tag{6}$$

$$\mathcal{L}_{Skip-Gram} = -[\log \sigma({v'_w}^\top v_{w_I}) + \sum_{\substack{i=1 \\ \tilde{w}_i \sim Q}}^{N} \log \sigma(-{v'_{\tilde{w}_i}}^\top v_{w_I})] \tag{7}$$

GloVe [9] is one of the most widely used count-based embedding models. It performs matrix factorization on the co-occurrence matrix of words to learn the embeddings.

Subword and Character Embedding. Word-level embedding models suffer from problems such as OOV. One remedy is to segment words into subwords or characters for embeddings. Character-based embedding models not only can handle the OOV words [31, 32], but also can reduce the embedding model size. Subword methods find the most frequent character segments (subwords), and then learn the embeddings of these segments. FastText [11] is a popular subword embedding model, which represents each word as a bag of character n-grams. This is similar to the letter tri-grams used in DSSMs. Other popular subword tokenizers include byte pair encoding [206], WordPiece [207], SentencePiece [208], and so on.

Contextual Embedding. The meaning of a word depends on its context. For example, the word "play" in the sentence "kid is playing" has a different meaning from when it is in "this play was written by Mozart". Therefore, word embedding is desirable to be context sensitive. Neither Word2vec nor Glove is context sensitive. They simply map a word into the same vector regardless of its context. Contextualized word embedding models, on the other hand, can map a word to different embedding vectors depending on its context. ELMo [82] is the first large-scale context-sensitive embedding model which uses two LSTMs in forward and backward directions to encode word context.

A.2 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

RNNs [209] are widely used for processing sequential data, such as text, speech, video. The architecture of a vanilla RNN model is shown in Fig. 28 (left). The model gets the input from the current time X_i and the hidden state from the previous step h_{i-1} and generates a hidden state and optionally an output. The hidden state from the last time-stamp (or a weighted average of all hidden states) can be used as the representation of the input sequence for downstream tasks.

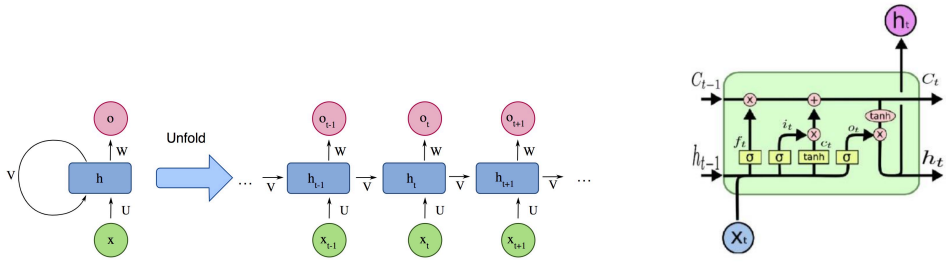

Fig. 28. (Left) The architecture of a RNN. (Right) The architecture of a standard LSTM module [210].

RNNs cannot capture long-term dependencies of very long sequences, which appear in many real applications, due to the gradient vanishing and explosion issue. LSTM is a variation of RNNs designed to better capture

long-term dependencies. As shown in Fig. 28 (right) and Eq. 8, the LSTM layer consists of a memory cell, which remembers values over arbitrary time intervals, and three gates (input gate, output gate, forget gate) that regulate the flow of information in and out the cell. The relationship between input, hidden states, and different gates of LSTM is shown in Equation 8:

$$
\begin{aligned}
f_t &= \sigma(\mathbf{W}^{(f)}x_t + \mathbf{U}^{(f)}h_{t-1} + b^{(f)}),\\
i_t &= \sigma(\mathbf{W}^{(i)}x_t + \mathbf{U}^{(i)}h_{t-1} + b^{(i)}),\\
o_t &= \sigma(\mathbf{W}^{(o)}x_t + \mathbf{U}^{(o)}h_{t-1} + b^{(o)}),\\
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(\mathbf{W}^{(c)}x_t + \mathbf{U}^{(c)}h_{t-1} + b^{(c)}),\\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{8}
$$

where $x_t \in R^k$ is a k-D word embedding input at time-step t, σ is the element-wise sigmoid function, $\odot$ is the element-wise product, $\mathbf{W}$, $\mathbf{U}$ and b are model parameters, c_t is the memory cell, the forget gate f_t determines whether to reset the memory cell, and the input gate i_t and output gate o_t control the input and output of the memory cell, respectively.

A.3 Convolutional Neural Networks (CNNs)

CNNs are originally developed for computer vision tasks, but later on made their way in various NLP applications. CNNs were initially proposed by Fukushima in his seminal paper "Neocognitron" [211], based on the model of the human visual system proposed by Hubel and Wiesel. Yann LeCun and his colleagues popularized CNNs by developing an efficient method of training CNNs based on back-propagation [25]. The architecture of the CNN model developed by LeCun et al. is shown in Fig. 29.

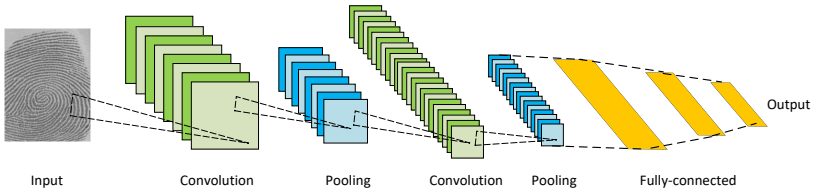

Fig. 29. Architecture of a CNN model, courtesy of Yann LeCun [25].

CNNs consist of three types of layers: (1) the convolutional layers, where a sliding kernel is applied to a region of an image (or a text segment) to extract local features; (2) the nonlinear layers, where a non-linear activation function is applied to (local) feature values; and (3) the pooling layers, where local features are aggregated (via the max-pooling or mean-pooling operation) to form global features. One advantage of CNNs is the weight sharing mechanism due to the use of the kernels, which results in a significantly smaller number of parameters than a similar fully-connected neural network, making CNNs much easier to train. CNNs have been widely used in computer vision, NLP, and speech recognition problems [1, 3, 26, 212–215].

A.4 Encoder-Decoder Models

Encoder-Decoder models learn to map input to output via a two-stage process: (1) the encoding stage, where an encoder $f(.)$ compresses input x into a latent-space vector representation z as $z = f(x)$; and (2) the decoding

stage, where a decoder $g(.)$ reconstructs or predicts output y from z as $y = g(z)$. The latent representation z is expected to capture the underlying semantics of the input. These models are widely used in sequence-to-sequence tasks such as machine translation, as illustrated in Fig. 30.

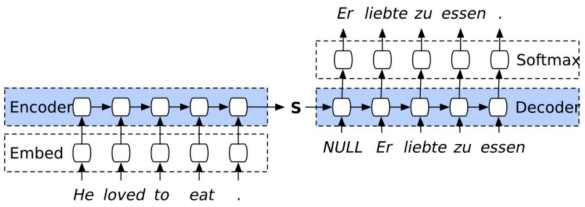

Fig. 30. A simple encoder-decoder model for machine translation. The input is a sequence of words in English, and the output is its translated version in German.

Autoencoders are special cases of the encoder-decoder models in which the input and output are the same. Autoencoders can be trained in an unsupervised fashion by minimizing the reconstruction loss.

A.5 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 deep learning models for NLP [58, 59]. In a nutshell, attention in language models can be interpreted as a vector of importance weights. In order to predict a word in a sentence, using the attention vector, we estimate 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.

Bahdanau et al. [58] conjectured that the use of a fixed-length state vector in CNNs is the bottleneck in improving the performance of the encoder-decoder model, and proposed to allow the decoder to search for parts in a source sentence that are relevant to predicting the target word, without having to compress the source sentence into the state vector. As shown in Fig. 31 (left), a linear combination of hidden vectors of input words h, weighted by attention scores α, is used to generate the output y. As we can see from Fig. 31 (right), different words in the source sentence are attended with different weights when generating a word in the target sentence.

Self-attention is a special attention mechanism, which allows to learn the correlation among the words in the same sentence [17]. This is very useful in NLP tasks such as machine reading, abstractive summarization, and image captioning. Transformers, which will be described later, also use self-attention.

A.6 Transformer

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 meaningful relationships between words in a sentence also grows with increasing length of the sentence, similar to RNNs. Transformers [2] overcome this limitation by computing in parallel for every word in a sentence or document an "attention score" to model the influence each word has on another. Due to this feature, Transformers allow for much more parallelization than CNNs and RNNs, and make it possible to efficiently train very big models on large amounts of data on GPU clusters.

As shown in Fig. 32 (a), the Transformer model consists of stacked layers in both encoder and decoder components. Each layer has two sub-layers comprising a multi-head attention layer (Fig. 32 (c)) followed by a position-wise feed forward network. For each set of queries Q, keys K and values V, the multi-head attention

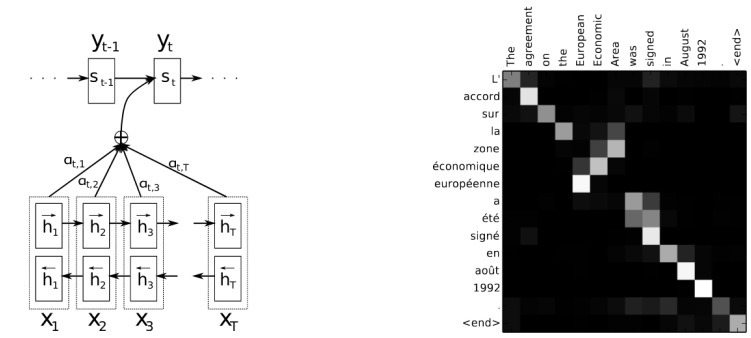

Fig. 31. (Left) The proposed attention mechanism in [58]. (Right) An example of attention mechanism in French to English machine translation, which shows the impact of each word in French in translating to English, Brighter cells have more impact.

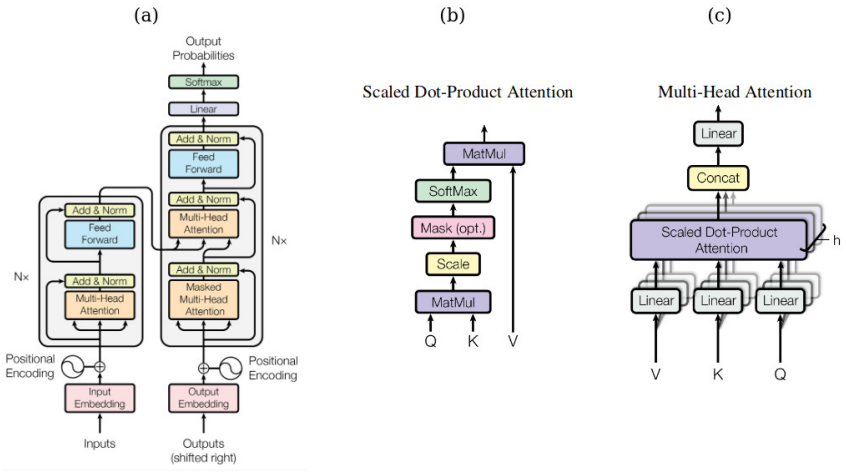

Fig. 32. (a) The Transformer model architecture. (b) Scaled Dot-Product Attention. (3) Multi-Head Attention consists of several attention layers running in parallel. [2]

.

module performs attention h times using the scaled dot-product attention as in Fig. 32 (b), where Mask (option) is the attention mask that is applied to prevent the target word information to be predicted from leaking to the decoder (during training) before prediction. Experiments show that multi-head attention is more effective than single-head attention. The attention of multiple heads can be interpreted as each head processing a different subspace at a different position. Visualization of the self-attention of multiple heads reveal that each head processes syntax and semantic structures [2].