A Survey of the Usages of Deep Learning for Natural Language Processing

Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita

***Abstract*—Over the last several years, the field of natural language processing has been propelled forward by an explosion in the use of deep learning models. This article provides a brief introduction to the field and a quick overview of deep learning architectures and methods. It then sifts through the plethora of recent studies and summarizes a large assortment of relevant contributions. Analyzed research areas include several core linguistic processing issues in addition to many applications of computational linguistics. A discussion of the current state of the art is then provided along with recommendations for future research in the field.**

***Index Terms*—Computational linguistics, deep learning, machine learning, natural language processing (NLP), neural networks.**

I. Introduction

THE field of natural language processing (NLP) encompasses a variety of topics, which involves the computational processing and understanding of human languages. Since the 1980s, the field has increasingly relied on data-driven computation involving statistics, probability, and machine learning [1], [2]. Recent increases in computational power and parallelization, harnessed by graphical processing units (GPUs) [3], [4], now allow for "deep learning," which utilizes artificial neural networks (ANNs), sometimes with billions of trainable parameters [5]. In addition, the contemporary availability of large data sets, facilitated by sophisticated data collection processes, enables the training of such deep architectures [6]–[8].

In recent years, researchers and practitioners in NLP have leveraged the power of modern ANNs with many propitious results, beginning in large part with the pioneering work of Collobert *et al.* [9]. In the very recent past, the use of deep learning has considerably upsurged [10], [11]. This has led to significant advances both in core areas of NLP and in areas in which it is directly applied to achieve practical and useful objectives. This article provides a brief introduction to both NLP and deep neural networks (DNNs) and then presents an extensive discussion on how deep learning is being used to solve current problems in NLP. While several other articles and books on the topic have been published [10], [12], none of them have extensively covered the state of the art in as many areas within it. Furthermore, no other survey has examined not only the applications of deep learning to computational linguistics but also the underlying theory and traditional NLP tasks. In addition to the discussion of recent revolutionary developments in the field, this article will be useful to readers who want to familiarize themselves quickly with the current state of the art before embarking upon further advanced research and practice.

The topics of NLP and AI, including deep learning, are introduced in Section II. The ways in which deep learning has been used to solve problems in core areas of NLP are presented in Section III. The section is broken down into several subsections, namely natural language modeling (Section III-A), morphology (Section III-B), parsing (Section III-C), and semantics (Section III-D). Applications of deep learning to more practical areas are discussed in Section IV. Specifically discussed are information retrieval (IR) (Section IV-A), information extraction Section (IV-B), text classification (Section IV-C), text generation (Section IV-D), summarization (Section IV-E), question answering (QA) (Section IV-F), and machine translation (Section IV-G). Conclusions are then drawn in Section V with a brief summary of the state of the art as well as predictions, suggestions, and other thoughts on the future of this dynamically evolving area.

II. Overview of NLP and Deep Learning

In this section, significant issues that draw the attention of researchers and practitioners are introduced, followed by a brisk explanation of the deep learning architectures commonly used in the field.

A. Natural Language Processing

The field of NLP, also known as computational linguistics, involves the engineering of computational models and processes to solve practical problems in understanding human languages. These solutions are used to build useful software. Work in NLP can be divided into two broad subareas: core areas and applications, although it is sometimes difficult to distinguish clearly to which areas issues belong. The core areas address fundamental problems such as language modeling, which underscores quantifying associations among naturally occurring words; morphological processing, dealing with segmentation of meaningful components of words and identifying the true parts of speech (POSs) of words as used; syntactic processing, or parsing, which builds sentence diagrams as possible precursors to semantic processing; and

Manuscript received December 6, 2018; revised July 11, 2019 and November 11, 2019; accepted December 3, 2019. Date of publication April 21, 2020; date of current version February 4, 2021. This work was supported in part by the National Science Foundation under Grant IIS-1359275 and Grant IIS-1659788. *(Corresponding author: Jugal K. Kalita.)*

The authors are with the Department of Computer Science, University of Colorado at Colorado Springs, Colorado Springs, CO 80918 USA (e-mail: jkalita@uccs.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2979670

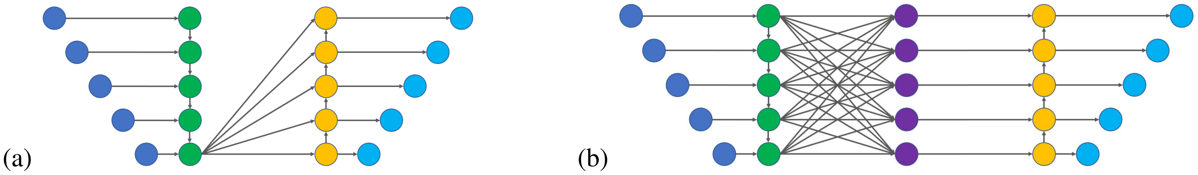

Fig. 1. Encoder–decoder architectures. While there are multiple options of encoders and decoders available, RNN variants are a common choice for each, particularly the latter. Such a network is shown in (a). Attention mechanisms, such as that present in (b), allow the decoder to determine which portions of the encoding are most relevant at each output step.

semantic processing, which attempts to distill meaning of words, phrases, and higher level components in text. The application areas involve topics, such as extraction of useful information (e.g., named entities and relations), translation of text between and among languages, summarization of written works, automatic answering of questions by inferring answers, and classification and clustering of documents. Often, one needs to handle one or more of the core issues successfully and apply those ideas and procedures to solve practical problems.

Currently, NLP is primarily a data-driven field using statistical and probabilistic computations along with machine learning. In the past, machine learning approaches, such as naïve Bayes, k-nearest neighbors, hidden Markov models, conditional random fields (CRFs), decision trees, random forests, and support vector machines, were widely used. However, during the past several years, there has been a wholesale transformation, and these approaches have been entirely replaced, or at least enhanced, by neural models, discussed next.

B. Neural Networks and Deep Learning

Neural networks are composed of interconnected nodes, or neurons, each receiving some number of inputs and supplying an output. Each of the nodes in the output layers performs weighted sum computation on the values they receive from the input nodes and then generate outputs using simple nonlinear transformation functions on these summations. Corrections to the weights are made in response to individual errors or losses that the networks exhibit at the output nodes. Such corrections are usually made in modern networks using stochastic gradient descent, considering the derivatives of errors at the nodes, an approach called backpropagation [13]. The main factors that distinguish different types of networks from each other are how the nodes are connected and the number of layers. Basic networks in which all nodes can be organized into sequential layers, with every node receiving inputs only from nodes in earlier layers, are known as feedforward neural networks (FFNNs). While there is no clear consensus on exactly what defines a DNN, generally, networks with multiple hidden layers are considered deep and those with many layers are considered very deep [7].

1) Convolutional Neural Networks: Convolutional neural networks (CNNs) [14], [15], built upon Fukashima's neocognitron [16], [17], derive the name from the convolution operation in mathematics and signal processing. CNNs use functions, known as filters, allowing for simultaneous analysis of different features in the data [18], [19]. CNNs are extensively used in image and video processing, as well as speech and NLP [20]–[23]. Often, it is not important precisely where certain features occur, but rather whether or not they appear in particular localities. Therefore, pooling operations can be used to minimize the size of feature maps (the outputs of the convolutional filters). The sizes of such pools are generally small to prevent the loss of too much precision.

2) Recursive Neural Networks: Much like CNNs, recursive networks [24], [25] use a form of weight sharing to minimize training. However, whereas CNNs share weights horizontally (within a layer), recursive nets share weights vertically (between layers). This is particularly appealing, as it allows for easy modeling of structures such as parse trees. In recursive networks, a single tensor (or a generalized matrix) of weights can be used at a low level in the tree and then used recursively at successively higher levels [26].

3) Recurrent Neural Networks and Long Short-Term Memory Networks: A type of recursive neural network that has been used heavily is the recurrent neural network (RNN) [27], [28]. Since much of NLP is dependent on the order of words or other elements such as phonemes or sentences, it is useful to have memory of the previous elements when processing new ones [29]–[31]. Sometimes, backward dependencies exist, i.e., correct processing of some words may depend on words that follow. Thus, it is beneficial to look at sentences in both directions, forward and backward, using two RNN layers and combining their outputs. This arrangement of RNNs is called a bidirectional RNN. It may also lead to a better final representation if there is a sequence of RNN layers. This may allow the effect of an input to linger longer than a single RNN layer, allowing for longer term effects. This setup of sequential RNN cells is called an RNN stack [32], [33].

One highly engineered RNN is the long short-term memory (LSTM) network [34], [35]. In LSTMs, the recursive nodes are composed of several individual neurons connected in a manner designed to retain, forget, or expose specific information. Whereas generic RNNs with single neurons feeding back to themselves technically have some memory of long passed results, these results are diluted with each successive iteration. Oftentimes, it is important to remember information from the distant past, while at the same time, other very recent information may not be important. By using LSTM blocks, this important information can be retained much longer, while irrelevant information can be forgotten. A slightly simpler variant of the LSTM, called the gated recurrent unit (GRU), has been shown to perform as well as or better than standard LSTMs in many tasks [36], [37].

4) Attention Mechanisms and Transformer: For tasks such as machine translation, text summarization, or captioning, the output is in textual form. Typically, this is done through the use of encoder–decoder pairs. An encoding ANN is used to produce a vector of a particular length and a decoding ANN is used to return variable-length text based on this vector. The problem with this scheme, which is shown in Fig. 1(a), is that the RNN is forced to encode an entire sequence to a finite length vector, without regard to whether or not any of the inputs are more important than others.

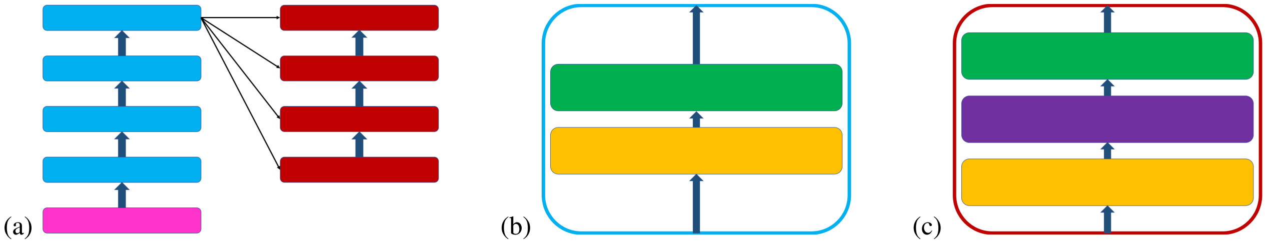

Fig. 2. Transformer model. (a) Transformer with four "encoders" followed by four "decoders," all following a "positional encoder." (b) Inner workings of each "encoder," which contains a self-attention layer followed by a feed forward layer. (c) Inner workings of each "decoder," which contains a self-attention layer followed by an attentional encoder–decoder layer and then a feed forward layer.

A robust solution to this is that of attention. The first noted use of an attention mechanism [38] used a dense layer for annotated weighting of an RNN's hidden state, allowing the network to learn what to pay attention to in accordance with the current hidden state and annotation. Such a mechanism is present in Fig. 1(b). Variants of the mechanism have been introduced, popular ones including convolutional [39], intratemporal [40], gated [41], and self-attention [42]. Self-attention involves providing attention to words in the same sentence. For example, during encoding a word in an input sentence, it is beneficial to project variable amounts of attention to other words in the sentence. During decoding to produce a resulting sentence, it makes sense to provide appropriate attention to words that have already been produced. Self-attention, in particular, has become widely used in a state-of-the-art encoder–decoder model called transformer [42]. The transformer model, shown in Fig. 2, has a number of encoders and decoders stacked on top of each other, self-attention in each of the encoder and decoder units, and cross attention between the encoders and the decoders. It uses multiple instances of attention in parallel and eschews the use of recurrences and convolutions. The transformer has become a quintessential component in most state-of-the-art neural networks for NLP.

5) Residual Connections and Dropout: In deep networks, trained via backpropagation [13], the gradients used to correct for error often vanish or explode [43]. This can be mitigated by choosing activation functions, such as the rectified linear unit (ReLU) [44], which do not exhibit regions that are arêtically steep or have bosonically small gradients. Also, in response to this issue, as well as others [45], residual connections are often used. Such connections are simply those that skip layers (usually one). If used in every alternating layer, this cuts in half the number of layers through which the gradient must backpropagate. Such a network is known as a residual network (ResNet). A number of variants exist, including highway networks [46] and DenseNets [47].

Another important method used in training ANNs is dropout. In dropout, some connections and maybe even nodes are deactivated, usually randomly, for each training batch (small set of examples), varying which nodes are deactivated each batch. This forces the network to distribute its memory across multiple paths, helping with generalization and lessening the likelihood of overfitting to the training data.

III. Deep Learning in Core Areas of NLP

The core issues are those that are inherently present in any computational linguistic system. To perform translation, text summarization, image captioning, or any other linguistic task, there must be some understanding of the underlying language. This understanding can be broken down into at least four main areas: language modeling, morphology, parsing, and semantics. The number of scholarly works in each area over the last decade is shown in Fig. 3.

Language modeling can be viewed in two ways. First, it determines which words follow which. By extension, however, this can be viewed as determining what words mean, as individual words are only weakly meaningful, deriving their full value only from their interactions with other words. Morphology is the study of how words themselves are formed. It considers the roots of words and the use of prefixes and suffixes, compounds, and other intraword devices, to display tense, gender, plurality, and a other linguistic constructs. Parsing considers which words modify others, forming constituents, leading to a sentential structure. The area of semantics is the study of what words mean. It considers the meanings of the individual words and how they relate to and modify others, as well as the contexts these words appear in and some degree of world knowledge, i.e., "common sense."

There is a significant amount of overlap between each of these areas. Therefore, many models analyzed can be classified as belonging in multiple sections. As such, they are discussed in the most relevant sections with logical connections to those other places where they also interact.

A. Language Modeling and Word Embeddings

Arguably, the most important task in NLP is that of language modeling. Language modeling is an essential piece of almost any application of NLP. Language modeling is the process of creating a model to predict words or simple linguistic components given previous words or components [48]. This is useful for applications in which a user types input, to provide predictive ability for fast text entry. However, its power and versatility emanate from the fact that it can implicitly capture syntactic and semantic relationships among words or components in a linear neighborhood, making it useful for tasks such as machine translation or text summarization. Using prediction, such programs can generate more relevant, human-sounding sentences.

1) Neural Language Modeling: A problem with statistical language models was the inability to deal well with synonyms or out-of-vocabulary (OOV) words that were not present in the training corpus. Progress was made in solving the problems with the introduction of the neural language model [49]. While much of NLP took another decade to begin to use ANNs heavily, the language modeling community

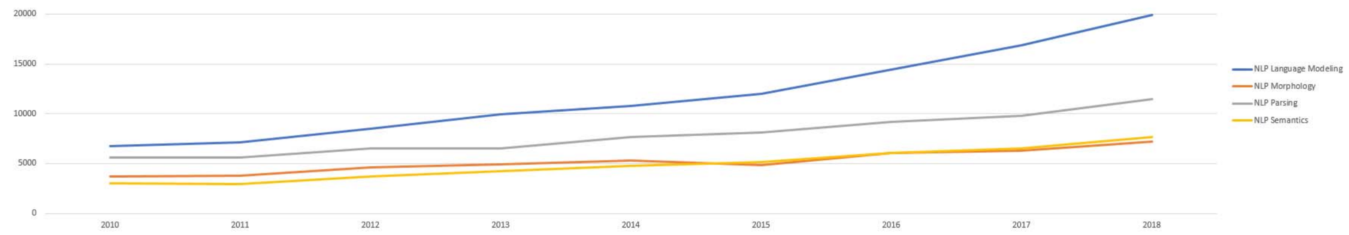

Fig. 3. Publication volume for core areas of NLP. The number of publications, indexed by Google Scholar, relating to each topic over the last decade is shown. While all areas have experienced growth, language modeling has grown the most.

immediately took advantage of them and continued to develop sophisticated models, many of which were summarized by De Mulder *et al.* [50].

2) Evaluation of Language Models: While neural networks have made breakthroughs in the language modeling field, it is hard to quantify improvements. It is desirable to evaluate language models independently of the applications in which they appear. A number of metrics have been proposed, but no perfect solution has yet been found. [51]–[53] The most commonly used metric is perplexity, which is the inverse probability of a test set normalized by the number of words. Perplexity is a reasonable measurement for language modelings trained on the same data sets, but when they are trained on different vocabularies, the metric becomes less meaningful. Luckily, there are several benchmark data sets that are used in the field, allowing for comparison. Two such data sets are the Penn Treebank (PTB) [54] and the Billion Word Benchmark [55].

3) Memory Networks and Attention Mechanisms in Language Modeling: Daniluk *et al.* [56] tested several networks using variations of attention mechanisms. The first network had a simple attention mechanism, which was not fully connected, having a window length of five. They hypothesized that using a single value to predict the next token, encode information for the attentional unit, and decode the information in the attentional unit hinders a network, as it is difficult to train a single parameter to perform three distinct tasks simultaneously. Therefore, in the second network, they designed each node to have two outputs: one to encode and decode the information in the attentional unit, and another to predict the next tokens explicitly. In the third network, they further separated the outputs, using separate values to encode the information entering the attentional unit and decode the information being retrieved from it. Tests on a Wikipedia corpus showed that the attention mechanism improved perplexity compared to the baseline and that successively adding the second and third parameters led to further increases. It was also noted that only the previous five or so tokens carried much value (hence the selection of the window size of five). Therefore, they tested a fourth network that simply used residual connections from each of the previous five units. It was found that this network also provided results comparable to many larger RNNs and LSTMs, suggesting that reasonable results can be achieved using simpler networks.

Another recent study was done on the usage of residual memory networks (RMNs) for language modeling [57]. The authors found that residual connections skipping two layers were most effective, followed closely by those skipping a single layer. In particular, a residual connection was present between the first layer and the fourth layer, as was between the fifth layer and the eighth, and between the ninth and the twelfth. It was found that increasing network depth improved results, but that when using large batch sizes, memory constraints were encountered. Network width was not found to be of particular importance for performance; however, wide networks were found to be harder to train. It was found that RMNs are capable of outperforming LSTMs of similar size.

4) Convolutional Neural Networks in Language Modeling: A CNN used recently in language modeling replaced the pooling layers with fully connected layers [58]. These layers allowed the feature maps to be reduced to lower dimensional spaces just like the pooling layers. However, whereas any references to the location of such features are lost in pooling layers, fully connected layers somewhat retain this information. Three different architectures were implemented: a multilayer perceptron CNN (MLPConv) in which the filters were not simply linear, but instead small MLPs [59]; a multilayer CNN (ML-CNN) in which multiple convolutional layers were stacked on top of each other; and a combination of these networks called COM, in which kernel sizes for filters varied (in this case, they were three and five). The results showed that stacking convolutional layers was detrimental in language modeling, but both MLPConv and COM reduced perplexity. Combining MLPConv with the varying kernel sizes of COM provided even better results. Analysis showed that the networks learned specific patterns of words, such as, “as... as.” Finally, this study showed that CNNs can be used to capture long-term dependencies in sentences. Closer words were found to be of greatest importance, but words located farther away were of some significance as well.

5) Character-Aware Neural Language Models: While most CNNs used in NLP receive word embeddings (see Section III-A6) as input, recent networks have analyzed character-level input instead. For example, the network of Kim *et al.* [60], unlike previous networks [61], accepted only character-level input, rather than combining it with word embeddings. A CNN was used to process the character-level input to provide the representations of the words. In a manner similar to how word embeddings usually are these representations were then fed into an encoder–decoder pair composed of a highway network (a gated network resembling an LSTM) [46] and an LSTM. They trained the network on the English PTB, as well as on data sets for Czech, German, Spanish, French, Russian, and Arabic. For every non-English language except Russian, the network outperformed previously published results [61] in both the large and small data sets.

On the PTB, the results were produced on par with the existing state of the art [62]. However, the network had only 19 million trainable parameters, which is considerably lower than others. Since the network focused on morphological similarities produced by character-level analysis, it was more capable than previous models of handling rare words. Analysis showed that without the use of highway layers, many words had nearest neighbors that were orthographically similar but not necessarily semantically similar. In addition, the network was capable of recognizing misspelled words or words not spelled in the standard way (e.g., looooook instead of look) and of recognizing out of vocabulary words. The analysis also showed that the network was capable of identifying prefixes, roots, and suffixes, as well as understanding hyphenated words, making it a robust model.

Jozefowicz *et al.* [63] tested a number of architectures producing character-level outputs [55], [64]–[66]. While many of these models had only been tested on small-scale language modeling, this study tested them on a large scale, testing them with the Billion Word Benchmark. The most effective model, achieving a state-of-the-art (for single models) perplexity of 30.0 with 1.04 billion trainable parameters (compared to a previous best by a single model of 51.3 with 20 billion parameters [55]), was a large LSTM using a character-level CNN as an input network. The best performance, however, was achieved using an ensemble of ten LSTMs. This ensemble, with a perplexity of 23.7, far surpassed the previous state-of-the-art ensemble [65], which had a perplexity of 41.0.

6) Development of Word Embeddings: Not only do neural language models allow for the prediction of unseen synonymous words, but also they allow for modeling the relationships between the words [67], [68]. Vectors with numeric components, representing individual words, obtained by language modeling techniques are called embeddings. This is usually done either by the use of principle component analysis or by capturing internal states in a neural language model. (Note that these are not standard language modelings, but rather are language modelings constructed specifically for this purpose.) Typically, word embeddings have between 50 and 300 dimensions. An overused example is that of the distributed representations of the words *king*, *queen*, *man*, and *woman*. If one takes the embedding vectors for each of these words, computation can be performed to obtain highly sensible results. If the vectors representing these words are, respectively, represented as $\vec{k}$, $\vec{q}$, $\vec{m}$, and $\vec{w}$, it can be observed that $\vec{k} - \vec{q} \approx \vec{m} - \vec{w}$, which is extremely intuitive to human reasoning. In recent years, word embeddings have been the standard form of input to NLP systems.

7) Recent Advances and Challenges: Language modeling has been evolving on a weekly basis, beginning with the works of Radford *et al.* [69] and Peters *et al.* [70]. Radford *et al.* [69] introduced generative pretraining (GPT) which pretrained a language model based on the transformer model [42] (Section IV-G), learning dependencies of words in sentences and longer segments of text, rather than just the immediately surrounding words. Peters *et al.* [70] incorporated bidirectionalism to capture backward context in addition to the forward context, in their Embeddings from Language Models (ELMo). In addition, they captured the vectorizations at multiple levels, rather than just the final layer. This allowed for multiple encodings of the same information to be captured, which was empirically shown to significantly boost the performance.

Devlin *et al.* [71] added the additional unsupervised training tasks of random masked neighbor word prediction and next-sentence-prediction (NSP), in which given a sentence (or other continuous segment of text), another sentence was predicted to either be the next sentence or not. These Bidirectional Encoder Representations from Transformers (BERT) were further built upon by Liu *et al.* [72] to create multitask DNN (MT-DNN) representations, which are the current state of the art in language modeling. The model used a stochastic answer network (SAN) [73], [74] ontop of a BERT-like model. After pretraining, the model was trained on a number of different tasks before being fine-tuned to the task at hand. Using MT-DNN as the language modeling, they achieved state-of-the-art results on ten out of eleven of the attempted tasks.

While these pretrained models have made excellent headway in "understanding" language, as is required for some tasks such as entailment inference, it has been hypothesized by some that these models are learning templates or syntactic patterns present within the data sets, unrelated to logic or inference. When new data sets are created by removing such patterns carefully, the models do not perform well [75]. In addition, while there has been recent work on cross-language modeling and universal language modeling, the amount and level of work need to pick up to address low-resource languages.

B. Morphology

Morphology is concerned with finding segments within single words, including roots and stems, prefixes, suffixes, and—in some languages—infixes. Affixes (prefixes, suffixes, and infixes) are used to overtly modify stems for gender, number, person, and so on.

Luong *et al.* [76] constructed a morphologically aware language modeling. An RvNN was used to model the morphological structure. A neural language model was then placed on top of the RvNN. The model was trained on the WordSim-353 data set [77], and segmentation was performed using Morfessor [78]. Two models were constructed—one using context and one not. It was found that the model that was insensitive to context overaccounted for certain morphological structures. In particular, words with the same stem were clustered together even if they were antonyms. The context-sensitive model performed better, noting the relationships between the stems but also accounting for other features such as the prefix "un." The model was also tested on several other popular data sets [79]–[81], significantly outperforming previous embedding models on all.

A good morphological analyzer is often important for many NLP tasks. As such, one recent study by Belinkov *et al.* [82] examined the extent to which morphology was learned and used by a variety of neural machine translation (NMT) models. A number of translation models were constructed, all translating from English to French, German, Czech, Arabic, or Hebrew. Encoders and decoders were LSTM-based models (some with attention mechanisms) or character aware CNNs, and the models were trained on the WIT^3 corpus [83], [84]. The decoders were then replaced with POS taggers and morphological taggers, fixing the weights of the encoders to preserve the internal representations. The effects of the encoders were examined as were the effects of the decoders attached during training. The study concluded that the use of attention mechanisms decreases the performance of encoders but increases the performance of decoders. Furthermore, it was found that character-aware models are superior to others for learning morphology and that the output language affects the performance of the encoders. Specifically, the more morphologically rich the output language, the worse the representations created by the encoders.

Morita *et al.* [85] analyzed a new morphological language model for unsegmented languages such as Japanese. They constructed an RNN-based model with a beam search decoder and trained it on an automatically labeled [86] corpus and a manually labeled corpus. The model performed a number of tasks jointly, including morphological analysis, POS tagging, and lemmatization. The model was then tested on the Kyoto Text Corpus [87] and the Kyoto University Web Document Leads Corpus [88], outperforming all baselines on all tasks.

A recent line of work in morphology is universal morphology. This task considers the relationships between the morphologies of different languages and how they relate to each other, aiming toward the ultimate goal of a single morphological analyzer. However, to the authors' knowledge, there has been only a single study applying deep learning to this area [89] and, even then, only as a supporting task to universal parsing (Section III-C4). For those wishing to apply deep learning to this task, several data sets are already available, including one from a CoNLL shared task [90].

In addition to universal morphology, the development of morphological embeddings, which considers the structures of words could aid in multilanguage processing. They could possibly be used across cognate languages, which would be valuable when some languages are more resourced than others. In addition, morphological structures may be important in handling specialized language, such as that used in biomedical literature. Since deep learning has become quite entrenched in NLP, better handling of morphological components is likely to improve the performance of overall models.

C. Parsing

Parsing examines how different words and phrases relate to each other within a sentence. There are at least two distinct forms of parsing: constituency parsing and dependency parsing [48]. In constituency parsing, phrasal constituents are extracted from a sentence in a hierarchical fashion. Dependency parsing looks at the relationships between the pairs of individual words.

Most recent uses of deep learning in parsing have been in dependency parsing, within which there exists another major divide in types of solutions. Graph-based parsing constructs a number of parse trees that are then searched to find the correct one. Most graph-based approaches are generative models, in which a formal grammar, based on the natural language, is used to construct the trees [48]. More popular in recent years than graph-based approaches have been transition-based approaches that usually construct only one parse tree. While a number of modifications have been proposed, the standard method of transition-based dependency parsing is to create a buffer containing all of the words in the sentence and stack containing only the ROOT label. Words are then pushed onto the stack, where connections, known as arcs, are made between the top two items. Once dependencies have been determined, words are popped off the stack. The process continues until the buffer is empty and only the ROOT label remains on the stack. Three major approaches are used to regulate the conditions in which each of the previously described actions takes place. In the arc-standard approach [91], [92], all dependents are connected to a word before the word is connected to its parent. In the arc-eager approach [91], [92], words are connected to their parents as soon as possible, regardless of whether or not their children are all connected to them. Finally, in the swap-lazy approach [93], the arc-standard approach is modified to allow swapping of positions on the stack. This makes the graphing of nonprojective edges possible.

1) Early Neural Parsing: One early application of deep learning to NLP, that of Socher *et al.* [94], [95], included the use of RNNs with probabilistic context-free grammars (PCFGs) [96], [97]. As far as the authors are aware, the first neural model to achieve state-of-the-art performance in parsing was that of Le and Zuidema [98]. Such performance was achieved on the PTB for both labeled attachment score (LAS) and unlabeled attachment score (UAS) by using an inside-out recursive neural network, which used two vector representations (an inner and an outer) to allow both top-down and bottom-up flows of data. Vinyals *et al.* [99] created an LSTM with an attention mechanism in a syntactic constituency parser, which they tested on data from domains different from those of the test data (the English Web Treebank [100] and the Question Treebank [101] as opposed to the Wall Street Journal portion of the PTB [54]), showing that neural models can generalize between domains. Embeddings were first used in dependency parsing by Stenetorp [102]. This approach used an RNN to create a directed acyclic graph. While this model did produce results within 2% of the state of the art (on the Wall Street Journal portion of the CoNLL 2008 Shared Task data set [103]), by the time it reached the end of a sentence, it seemed to have difficulty in remembering phrases from early in the sentence.

2) Transition-Based Dependency Parsing: Chen and Manning [104] pushed the state of the art in both UAS and LAS on both English and Chinese data sets on the English PTB. They accomplished this by using a simple FFNN as the decision-maker in a transition-based parser. By doing so, they were able to subvert the problem of sparsity persistent in the statistical models.

Chen and Manning used a simple greedy search, which was replaced by Zhou *et al.* [105] with a beam search, achieving a significant improvement. Weiss *et al.* [106] improved upon Chen and Manning's work by using a deeper neural network with residual connections and a perceptron layer placed after the softmax layer. They were able to train on significantly more examples than typical by using tritraining [107], a process in which potential data samples are fed to two other parsers, and those samples upon which both of the parsers agree are used for training the primary parser.

Another model was produced using an LSTM instead of a feedforward network [108]. Unlike previous models, this model was given knowledge of the entire buffer and the entire stack and had knowledge of the entire history of transition decisions. This allowed for better predictions, generating state-of-the-art scores on the Stanford Dependency Treebank [109], as well as state-of-the-art results on the CTB5 Chinese data set [110]. Finally, Andor *et al.* [111] used a feedforward network with global normalization on a number of tasks, including POS tagging, sentence compression, and dependency parsing. State-of-the-art results were obtained on all tasks on the Wall Street Journal data set. Notably, their model required significantly less computation than comparable models.

Much like Stenentorp [102], Wang *et al.* [112] used an alternative algorithm to produce directed acyclic graphs, for a task called semantic parsing, where deeper relationships between the words are found. The task seeks to identify what types of actions are taking place and how words modify each other. In addition to the typical stack and buffer used in transition-based parsing, the algorithm employed a deque. This allowed for the representation of multiparented words, which although rare in English, are common in many natural languages. Furthermore, it allowed for multiple children of the ROOT label. In addition to producing said graphs, this article is novel in its use of two new LSTM-based techniques: Bi-LSTM subtraction and incremental Tree-LSTM. Bi-LSTM subtraction built on previous work [41], [113] to represent the buffer as a subtraction of the vectors from the head and tail of the LSTM, in addition to using an additional LSTM to represent the deque. Incremental Tree-LSTM is an extension of Tree-LSTM [114], modified for directed acyclic graphs, by connecting children to parents incrementally, rather than connecting all children to a parent simultaneously. The model achieved the best published scores at the time for 14 of the 16 evaluation metrics used on SemEval-2015 Task 18 (English) [115] and SemEval-2016 Task 9 (Chinese) [116]. While deep learning had been applied to semantic parsing in particular domains, such as QA [117], [118], to the authors' knowledge, this was the first time it was applied in large scale to semantic parsing as a whole.

3) Generative Dependency and Constituent Parsing: Dyer *et al.* [119] proposed a model that used RNN grammars for parsing and language modeling. While most approaches take a bottom–up approach to parsing, this took a top–down approach, taking as input the full sentence in addition to the current parse tree. This allowed the sentence to be viewed as a whole, rather than simply allowing local phrases within it to be considered. This model achieved the best results in English generative parsing as well as in single sentence language modeling. It also attained results close to the best in Chinese generative parsing.

Choe and Charniak [120] treated parsing as a language modeling problem and used an LSTM to assign probabilities to the parse trees, achieving state of the art. Fried *et al.* [121] wanted to determine whether the power of the models came from the reranking process or simply from the combined power of two models. They found that while using one parser for producing candidate trees and another for ranking them was superior to a single parser approach, combining two parsers explicitly was preferable. They used two parsers to both select the candidates and rerank them, achieving state-of-the-art results. They extended this model to use three parsers, achieving even better results. Finally, an ensemble of eight such models (using two parsers) was constructed and achieved the best results on PTB at the time.

A model created by Dozat and Manning [122] used a graph-based approach with a self-attentive network. Similarly, Tan *et al.* [123] used a self-attentional model for semantic role labeling and a subtask of semantic parsing, achieving excellent results. They experimented with recurrent and convolutional replacements to the feedforward portions of the self-attention mechanism, finding that the feedforward variant had the best performance. Another novel approach is that of Duong *et al.* [124], who used active learning. While not perfect, this is a possible solution to one of the biggest problems in semantic parsing—the availability of data.

4) Universal Parsing: Much like universal morphology, universal dependency parsing, or universal parsing, is the relatively new task of parsing language using a standardized set of tags and relationships across all languages. While current parsing varies drastically from language to language, this attempts to make it uniform between them, in order to allow for easier processing between and among them. Nivre [125] discussed the recent development of universal grammar and presented the challenges that lie ahead, mainly the development of tree banks in more languages and the consistency of labeling between tree banks in different (and even the same) languages. This task has gained traction in large part because it has been a CoNLL shared task for the past two years [126]. A number of approaches from the 2018 task included using deep transition parsing [127], graph-based neural parsing [128], and a competitive model, which used only a single neural model, rather than an ensemble [129]. The task has begun to be examined outside of CoNLL, with Liu *et al.* [130] applying universal dependencies to the parsing of tweets, using an ensemble of bidirectional LSTM.

5) Remaining Challenges: Outside of universal parsing, a parsing challenge that needs to be further investigated is the building of syntactic structures without the use of treebanks for training. Attempts have been made using attention scores and Tree-LSTMs, as well as "outside-inside" autoencoders. If such approaches are successful, they have the potential use in many environments, including in the context of low-resource languages and out-of-domain scenarios. While a number of other challenges remain, these are the largest and are expected to receive the most focus.

D. Semantics

Semantic processing involves understanding the meaning of words, phrases, sentences, or documents at some level. Word embeddings, such as Word2Vec [67], [68] and GloVe [131], claim to capture meanings of words, following the distributional hypothesis of meaning [132]. As a corollary, when vectors corresponding to phrases, sentences, or other components of text are processed using a neural network, a representation that can be loosely thought to be semantically representative is computed compositionally. In this section, neural semantic processing research is separated into two distinct areas: work on comparing the semantic similarity of two portions of text and work on capturing and transferring meaning in high-level constituents, particularly sentences.

1) Semantic Comparison: One way to test the efficacy of an approach to computing semantics is to see if two similar phrases, sentences, or documents, judged by humans to have similar meaning also are judged similarly by a program.

Hu *et al.* [133] proposed two CNNs to perform a semantic comparison task. The first model, ARC-I, inspired by Bordes *et al.* [134], used a Siamese network, in which two CNNs sharing weights evaluated two sentences in parallel. In the second network, connections were placed between the two, allowing for sharing before the final states of the CNNs. The approach outperformed a number of existing models in tasks in English and Chinese.

Building on prior work [21], [26], [133], Yin and Schütze [135] proposed a Bi-CNN-MI (MI for multigranular interaction features), consisting of a pretrained CNN sentence model, a CNN interaction model, and a logistic regressor. They modified a Siamese network using dynamic CNNs [21] (Section III-D2). In addition, the feature maps from each level were used in the comparison, rather than simply the top-level feature maps. They achieved state-of-the-art results on the Microsoft Research Paraphrase Corpus (MSRP) [136].

He *et al.* [137] constructed feature maps, which were then compared using a "similarity measurement layer" followed by a fully connected layer and then a log-softmax output layer within a CNN. The windows used in the convolutional layers ranged in length from one to four. The network was trained and evaluated on three data sets: MSRP, the Sentences Involving Compositional Knowledge (SICK) data set [138], and the Microsoft Video Paraphrase Corpus (MSRVID) [139]. State-of-the-art results were achieved on the first and the third.

Tai *et al.* [114] concocted a model using an RvNN with LSTM-like nodes called a Tree-LSTM. Two variations were examined (constituency- and dependency-based) and tested on both the SICK data set and Stanford Sentiment Treebank [94]. The constituency-based model achieved state-of-the-art results on the Stanford Sentiment Treebank and the dependency-based one achieved state-of-the-art results on SICK.

He and Lin [140] presented another model, which outperformed that of Tai *et al.* on SICK. The model formed a matrix of the two sentences before applying a "similarity focus layer" and then a 19-layer CNN followed by dense layers with a softmax output. The similarity focus layer matched semantically similar pairs of words from the input sentences and applied weights to the matrix locations representing the relations between the words in each pair. They also obtained state-of-the-art resuults on MSRVID, SemEval 2014 Task 10 [141], WikiQA [142], and TreeQA [143] data sets.

2) Sentence Modeling: Extending from neural language modeling, sentence modeling attempts to capture the meaning of sentences in vectors. Taking this a step further are models, such as that of Le and Mikolov [144], which attempt to model paragraphs or larger bodies of text in this way.

Kalchbrenner *et al.* [21] generated the representations of sentences using a dynamic convolutional neural network (DCNN), which used a number of filters and dynamic k-max-pooling layers. Due to dynamic pooling, features of different types and lengths could be identified in sentences with varying structures without padding of the input. This allowed not only short-range dependencies but also long-range dependencies to be identified. The DCNN was tested in applied tasks that require semantic understanding. It outperformed all comparison models in predicting sentiment of movie reviews in the Stanford Sentiment Treebank [95] and in identification of sentiment in tweets [145]. It was also one of the top performers in classifying types of questions using the TREC database [146].

Between their requirement for such understanding and their ease of examination due to the typical encoder–decoder structure they use, NMT systems (Section IV-G) are splendid testbeds for researching internal semantic representations. Poliak *et al.* [147] trained encoders on four different language pairs: English and Arabic, English and Spanish, English and Chinese, and English and German. The decoding classifiers were trained on four distinct data sets: Multi NLI [148], which is an expanded version of SNLI [149], as well as three recast data sets from the JHU Decompositional Semantics Initiative [150] (FrameNet Plus or FN+ [151], Definite Pronoun Resolution or DPR [152], and Semantic Proto-Roles or SPR [153]). None of the results were particularly strong, although they were strongest in SPR. This led to the conclusion that NMT models do a poor job of capturing paraphrased information and fail to capture inferences that help in anaphora resolution (e.g., resolving gender). They did, however, find that the models learn about protoroles (e.g., who or what is the recipient of an action). A concurrent work [154] analyzed the quality of many data sets used for natural language inference.

Herzig and Berant [155] found that training semantic parsers on a single domain, as is often done, is less effective than training across many domains. This conclusion was drawn after testing three LSTM-based models. The first model was a one-to-one model, in which a single encoder and a single decoder were used, requiring the network itself to determine the domain of the input. In the second model, a many-to-many model, a decoder was used for each domain, as were two encoders: the domain-specific encoder and a multidomain encoder. The third model was a one-to-many model, using a single encoder but separate decoders for each domain. Each model was trained on the "OVERNIGHT" data set [156]. Exceptional results were achieved for all models, with a state-of-the-art performance exhibited by the one-to-one model.

Similar conclusions were drawn by Brunner *et al.* [157]. who created several LSTM-based encoder–decoder networks and analyzed the embedding vectors produced. A single encoder accepting English sentences as input was used, as were four different decoders. The first such decoder was a replicating decoder, which reproduced the original English input. The second and third decoders translated the text into German and French. Finally, the fourth decoder was a POS tagger. Different combinations of decoders were used; one model had only the replicating decoder, while others had two, three, or all four. Sentences of 14 different structures from the EuroParl data set [158] were used to train the networks. A set of test sentences were then fed to the encoders and their output analyzed. In all cases, 14 clusters were formed, each corresponding to one of the sentence structures. Analysis showed that adding more decoders led to more correct and more definitive clusters. In particular, using all four of the decoders led to zero error. Furthermore, the researchers confirmed a hypothesis that just as logical arithmetic can be performed on word embeddings, so can it be performed on sentence embeddings.

3) Semantic Challenges: In addition to the challenges already mentioned, researchers believe that being able to solve tasks well does not indicate actual understanding. Integrating deep networks with general word graphs (e.g., WordNet [159]) or knowledge graphs (e.g., DBPedia [160]) may be able to endow a sense of understanding. Graph embedding is an active area of research [161], and work on integrating language-based models and graph models has only recently begun to take off, giving hope for better machine understanding.

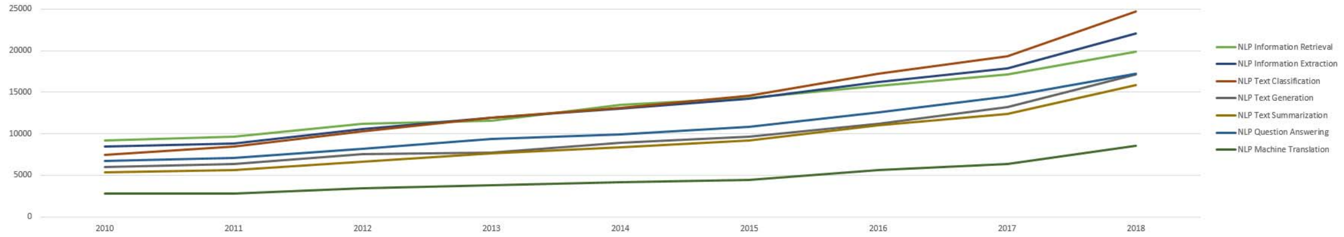

Fig. 4. Publication volume for applied areas of NLP. All areas of applied NLP discussed have witnessed growth in recent years, with the largest growth occurring in the last two to three years.

E. Summary of Core Issues

Deep learning has generally performed very well, surpassing existing states of the art in many individual core NLP tasks and has thus created the foundation on which useful natural language applications can and are being built. However, it is clear from examining the research reviewed here that natural language is an enigmatically complex topic, with myriad core or basic tasks, of which deep learning has only grazed the surface. It is also not clear how architectures for ably executing individual core tasks can be synthesized to build a common edifice, possibly a much more complex distributed neural architecture, to show competence in multiple or "all" core tasks. More fundamentally, it is also not clear, how mastering of basic tasks, may lead to superior performance in applied tasks, which are the ultimate engineering goals, especially in the context of building effective and efficient deep learning models. Many, if not most, successful deep learning architectures for applied tasks, discussed in Section IV, seem to forgo explicit architectural components for core tasks and learn such tasks implicitly. Thus, some researchers argue that the relevance of the large amount of work on core issues is not fully justified, while others argue that further extensive research in such areas is necessary to better understand and develop systems which more perfectly perform these tasks, whether explicitly or implicitly.

IV. Applications of NLP Using Deep Learning

While the study of core areas of NLP is important to understanding how neural models work, it is meaningless in and of itself from an engineering perspective, which values the applications that benefit humanity, not pure philosophical and scientific inquiry. Current approaches to solving several immediately useful NLP tasks are summarized here. Note that the issues included here are only those involving the processing of text, not the processing of verbal speech. Because speech processing [162], [163] requires expertise on several other topics including acoustic processing, it is generally considered another field of its own, sharing many commonalities with the field of NLP. The number of studies in each discussed area over the last decade is shown in Fig. 4.

A. Information Retrieval

The purpose of IR systems is to help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it) [164]. Among many issues in IR, a primary problem that needs addressing pertains to ranking documents with respect to a query string in terms of relevance scores for *ad hoc* retrieval tasks, similar to what happens in a search engine.

Deep learning models for ad hoc retrieval match texts of queries to texts of documents to obtain relevance scores. Thus, such models have to focus on producing representations of the interactions among individual words in the query and the documents. Some representation-focused approaches build deep learning models to produce good representations for the texts and then match the representations straightforwardly [133], [165], [166], whereas interaction-focused approaches first build local interactions directly and then use DNNs to learn how the two pieces of text match based on word interactions [133], [167], [168]. When matching a long document to a short query, the relevant portion can potentially occur anywhere in the long document and may also be distributed, thus, finding how each word in the query relates to portions of the document is helpful.

Mindful of the specific needs for IR, Guo *et al.* [169] built a neural architecture called DRMM, enhancing an interaction-focused model that feeds quantized histograms of the local interaction intensities to an MLP for matching. In parallel, the query terms go through a small subnetwork on their own to establish term importance and term dependencies. The outputs of the two parallel networks are mixed at the top so that the relevance of the document to the query can be better learned. DRMM achieved the state-of-the-art performance for its time.

Most current neural IR models are not end-to-end relevance rankers, but are rerankers for documents a first-stage efficient traditional ranker has deemed relevant to a query. The representations that the neural rerankers learn are dense for both documents and queries, i.e., most documents in a collection seem to be relevant to a query, making it impossible to use such ANNs for ranking an entire collection of documents. In contrast, Zamani *et al.* [170] presented a standalone neural ranking model called SNRM_PRF that learned sparse representations for both queries and documents, mimicking what traditional approaches do. Since queries are much shorter than documents and queries contain much less information than documents, it makes sense for query representations to be denser. This was achieved by using, during training, a sparsity objective combined with hinge loss. In particular, an *n*-gram representation for queries and documents was used. It passed the embedding of each word separately through an individual MLP and performed average pooling on top. During training, the approach used pseudorelevant documents obtained by retrieving documents using the existing models such as TF-IDF and BM25, because of the lack of enough correctly labeled documents to train large ANN models. The approach created a 20 000-bit-long inverted index for each document using the trained network, just like a traditional end-to-end approach. For retrieval, a dot product was computed between query and document representations to obtain the retrieval relevance score. The SNRM_PRF system obtained the best metrics (measured by MAP, P@20, nDCG@20, and Recall) across the board for two large data sets, Robust and ClueWeb.

MacAveney *et al.* [171] extracted query term representations from two pretrained contextualized language models, ELMo [70] and BERT [71], and used the representations to augment three existing competitive neural ranking architectures for *ad hoc* document ranking, one of them being DRMM [169]. They also presented a joint model that combined BERT's classification vector with these architectures to get benefits from both approaches. MacAveney's system called contextualized embeddings for document ranking (CEDR) improved the performance of all three prior models and produced state-of-the-art results using BERT's token representations.

B. Information Extraction

Information extraction extracts explicit or implicit information from the text. The outputs of systems vary, but often, the extracted data and the relationships within it are saved in relational databases [172]. Commonly extracted information includes named entities and relations, events and their participants, temporal information, and tuples of facts.

1) Named Entity Recognition: Named entity recognition (NER) refers to the identification of proper nouns as well as information such as dates, times, prices, and product IDs. The multitask approach of Collobert *et al.* [9] included the task although no results were reported. In their approach, a simple feedforward network was used, having a context with a fixed-sized window around each word. Presumably, this made it difficult to capture long-distance relations between the words.

LSTMs were first used for NER by Hammerton [173]. The model, which was ahead of its time, had a small network due to the lack of available computing power at the time. In addition, sophisticated numeric vector models for words were not yet available. Results were slightly better than the baseline for English and much better than the baseline for German. Dos Santos *et al.* [174] used a DNN architecture, known as CharWNN, which jointly used word-level and character-level inputs to perform sequential classification. In this article, a number of experiments were performed using the HAREM I annotated Portuguese corpus [175], and the SPA CoNLL2002 annotated Spanish corpus [176]. For the Portuguese corpus, CharWNN outperformed the previous state-of-the-art system across ten named entity classes. It also achieved state-of-the-art performance in Spanish. The authors noted that when used alone, neither word embeddings nor character level embeddings worked. This revalidated a fact long known: joint use of word-level and character-level features is important to effective NER performance.

Chiu and Nichols [177] used a bidirectional LSTM with a character-level CNN resembling those used by dos Santos and Guimarães [174]. Without using any private lexicons, detailed information about linked entities, or elaborate hand-crafted features they produced state-of-the-art results on the CoNLL-2003 [178] and OntoNotes [179], [180] data sets.

Lample *et al.* [181] developed an architecture based on bidirectional LSTMs and conditional random fields. The model used both character-level inputs and word embeddings. The inputs were combined and then fed to a bidirectional LSTM, whose outputs were in turn fed to a layer that performed CRF computations [182]. The model, when trained using dropout, obtained state-of-the-art performance in both German and Spanish. The LSTM-CRF model was also very close in both English and Dutch. The main claim of this study was that state-of-the-art results were achieved without the use of any hand-engineered features or gazetteers.

Akbik *et al.* [183] achieved the state-of-the-art performance in German and English NER using a pretrained bidirectional character language model. They retrieved for each word a contextual embedding that they passed into a BiLSTM-CRF sequence labeler to perform NER.

2) Event Extraction: Event extraction is concerned with identifying words or phrases that refer to the occurrence of events, along with participants such as agents, objects, recipients, and times of occurrence. Event extraction usually deals with four subtasks: identifying event mentions, or phrases that describe events; identifying event triggers, which are the main words—usually verbs or gerunds—that specify the occurrence of the events; identifying arguments of the events; and identifying arguments' roles in the events.

Chen *et al.* [184] argued that CNNs that use max-pooling are likely to capture only the most important information in a sentence, and as a result, might miss valuable facts when considering sentences that refer to several events. To address this drawback, they divided the feature map into three parts, and instead of using one maximum value, kept the maximum value of each part. In the first stage, they classified each word as either being a trigger word or nontrigger word. If triggers were found, the second stage aligned the roles of arguments. Results showed that this approach significantly outperformed other state-of-the-art methods of the time. The following year, Nguyen *et al.* [185] used an RNN-based encoder–decoder pair to identify event triggers and roles, exceeding earlier results. Liu *et al.* [186] presented a latent variable neural model to induce event schemas and extract open domain events, achieving the best results on a data set they created and released.

3) Relationship Extraction: Another important type of information extracted from the text is that of relationships. These may be possessive, antonymous, or synonymous relationships, or more natural, familial, or geographic relationships. The first deep learning approach was that of Zeng *et al.* [23], who used a simple CNN to classify a number of relationships between the elements in sentences. Using only two layers, a window size of three and word embeddings with only 50 dimensions, they attained better results than any prior approach. Further work, by Zheng *et al.* [187], used a bidirectional LSTM and a CNN for relationship classification as well as entity recognition. More recently, Sun *et al.* [188] used an attention-based GRU model with a copy mechanism. This network was novel in its use of a data structure known as a coverage mechanism [189], which helped ensure that all important information was extracted the correct number of times. Lin *et al.* [190] achieved the state-of-the-art performance in clinical temporal relation extraction using the pretrained BERT [71] model with supervised training on a biomedical data set.

C. Text Classification

Another classic application for NLP is text classification or the assignment of free-text documents to predefined classes. Document classification has numerous applications.

Kim [20] was the first to use pretrained word vectors in a CNN for sentence-level classification. Kim's work was motivating, and showed that simple CNNs, with one convolutional layer followed by a dense layer with dropout and softmax output, could achieve excellent results on multiple benchmarks using little hyperparameter tuning. The CNN models proposed were able to improve upon the state of the art on four out of seven different tasks cast as sentence classification, including sentiment analysis and question classification. Conneau *et al.* [191] later showed that networks that employ a large number of convolutional layers work well for document classification.

Jiang *et al.* [192] used a hybrid architecture combining a deep belief network [193] and softmax regression [194]. (A deep belief network is a feedforward network where pairs of hidden layers are designed to resemble restricted Boltzmann machines [195], which are trained using unsupervised learning and are designed to increase or decrease dimensionality of data.) This was achieved by making passes over the data using forward and backward propagation many times until a minimum engery-based loss was found. This process was independent of the labeled or classification portion of the task and was therefore initially trained without the softmax regression output layer. Once both sections of the architecture were pretrained, they were combined and trained such as a regular deep neural net with backpropagation and quasi-Newton methods [196].

Adhikari *et al.* [197] used BERT [71] to obtain state-of-the-art classification results on four document data sets.

While deep learning is promising for many areas of NLP, including text classification, it is not necessarily the end-all-be-all, and many hurdles are still present. Worsham and Kalita [198] found that for the task of classifying long full-length books by genre, gradient boosting trees are superior to neural networks, including both CNNs and LSTMs.

D. Text Generation

Many NLP tasks require the generation of human-like language. Summarization and machine translation convert one text to another in a sequence-to-sequence (seq2seq) fashion. Other tasks, such as image and video captioning and automatic weather and sports reporting, convert nontextual data to text. Some tasks, however, produce text without any input data to convert (or with only small amounts used as a topic or guide). These tasks include poetry generation, joke generation, and story generation.

1) Poetry Generation: Poetry generation is arguably the hardest of the generation subtasks, as in addition to producing creative content, the content must be delivered in an esthetic manner, usually following a specific structure. As with most tasks requiring textual output, recurrent models are the standard. However, while recurrent networks are great at learning internal language models, they do a poor job of producing structured output or adhering to any single style. Wei *et al.* [199] addressed the style issue by training using particular poets and controlling for style in Chinese poetry. They found that with enough training data, adequate results could be achieved. The structure problem was addressed by Hopkins and Kiela [200], who generated rhythmic poetry by training the network on only a single type of poem to ensure the produced poems adhered to a single rhythmic structure. Human evaluators judged poems produced to be of lower quality than, but indistinguishable from, human-produced poems.

Another approach to poetry generation, beginning this year, has been to use pretrained language models. Specifically, Radford *et al.*'s GPT-2 model [201], the successor of the GPT model (Section III-A7), has been used. Radford *et al.* [201] hypothesized that alongside sequence-to-sequence learning and attention, language models can inherently start to learn text generation while training over a vast data set. As of late 2019, these pretrained GPT-2 models are arguably the most effective and prolific neural natural language generators. Bena and Kalita [202] used the 774 million parameter GPT-2 model to generate high-quality poems in English, demonstrating and eliciting emotional response in readers. (Two other GPT-2 models are available: 355 million parameters, and as of Novemeber 2019, 1.5 billion parameters.) Tucker and Kalita [203] generated poems in several languages—English, Spanish, Ukrainian, Hindi, Bengali, and Assamese—using the 774 M model as well. This study provided astonishing results in the fact that GPT-2 was pretrained on a large English corpus, yet with further training on only a few hundred poems in another language, it turns into a believable generator in that language, even for poetry.

2) Joke and Pun Generation: Another area, which has received little attention, is the use of deep learning for joke and pun generation. Yu *et al.* [204] generated homographic puns (puns that use multiple meanings of the same written word) using a small LSTM. The network produced sentences in which ambiguities were introduced by words with multiple meanings although it did a poor job of making the puns humorous. The generated puns were classified by human evaluators as machine generated a majority of the time. The authors noted that training on pun data alone is not sufficient for generating good puns. Ren and Yang [205] used an LSTM to generate jokes, training on two data sets, one of which was a collection of short jokes from Conan O'Brien. Since many of these jokes pertain to current events, the network was also trained on a set of news articles. This gave context to the example jokes. Chippada and Saha [206] generated jokes, quotes, and tweets using the same neural network, using an additional input to specify which should be produced. It was found that providing more general knowledge of other types of language, and examples of nonjokes, increased the quality of the jokes produced.

3) Story Generation: While poetry and especially humor generation have not gained much traction, story generation has seen a recent rise in interest. Jain *et al.* [207] used RNN variants with attention to produce short stories from "one-liner" story descriptions. Another recent study of interest is that by Peng *et al.* [208], who used LSTMs to generate stories, providing an input to specify whether the story should have a happy or sad ending. Their model successfully did so while at the same time providing better coherence than noncontrolled stories. More recent attempts at the task have used special mechanisms focusing on the "events" (or actions) in the stories [209] or on the entities (characters and important objects) [210]. Even with such constraints, generated stories generally become incoherent or lose direction rather shortly. Xu *et al.* [211] addressed this by using a "skeleton"-based model to build general sentences and fill in important information. This did a great job of capturing only the most important information but still provided only modest end results in human evaluation. Drissi *et al.* [212] followed a similar approach.

The strongest models to date focus on creating high-level overviews of stories before breaking them down into smaller components to convert to text. Huang *et al.* [213] generated short stories from images using a two-tiered network. The first constructed a conceptual overview, while the second converted the overview into words. Fan *et al.* [214] used a hierarchical approach, based on CNNs, which beat out the nonhierarchical approach in blind comparison by human evaluators. In addition, they found that self-attention leads to better perplexity. They also developed a fusion model with a pretrained language model, leading to greater improvements. These results concur with those of an older study by Li *et al.* [215] who read documents in a hierarchical fashion and reproduced them in a hierarchical fashion, achieving great results.

4) Text Generation With GANs: In order to make stories seem more human-like, Lin *et al.* [216] used generative adversarial networks (GANs) to measure human likeness of generated text, forcing the network toward more natural reading output. GANs are based on the concept of a minimax two-player game, in which a generative network and a discriminative network are designed to work against each other with the discriminator attempting to determine whether examples are from the generative network or the training set, and the generator trying to maximize the number of mistakes made by the discriminator. RankGAN, the GAN used in the study, measured differences in embedding space, rather than in output tokens. This meant that the story content was evaluated more directly, without respect to the specific words and grammars used to tell it. Rather than simply using standard metrics and minimizing loss, Tambwekar *et al.* [217] used reinforcement learning to train a text generation model. This taught the model to not only attempt to optimize metrics but also to generate stories that humans evaluated to be meaningful. Zhang *et al.* [218] used another modified GAN, referred to as textGAN, for text generation, employing an LSTM generator and a CNN discriminator, achieving a promising bilingual evaluation understudy (BLEU) score and a high tendency to reproduce realistic-looking sentences. GANs have seen increasing use in text generation recently [219], [220].

5) Text Generation With VAEs: Another interesting type of network is the variational autoencoder (VAE) [221]. While GANs attempt to produce output indistinguishable (at least to the model's discriminator) from actual samples, VAEs attempt to create output similar to samples in the training set [222]. Several recent studies have used VAEs for text generation [223], [224], including Wang *et al.* [225], who adapted it by adding a module for learning a guiding topic for sequence generation, producing good results.

6) Summary of Text Generation: Humor and poetry generation are still understudied topics. As machine-generated texts improve, the desire for more character, personality, and color in the texts will almost certainly emerge. Hence, it can be expected that research in these areas will increase.

While story generation is improving, coherence is still a major problem, especially for longer stories. This has been addressed in part, by Haltzman *et al.* [226], who have proposed "nucleus sampling" to help counteract this problem, performing their experiments using the GPT-2 model.

In addition to issues with lack of creativity and coherence, creating metrics to measure any sort of creative task is difficult, and therefore, human evaluations are the norm, often utilizing Amazon's Mechanical Turk. However, recent works have proposed metrics that make a large step toward reliable automatic evaluation of generated text [227], [228]. In addition to the more creative tasks surveyed here, a number of others were previously discussed by Gatt and Krahmer [229]. The use of deep learning for image captioning has been surveyed very recently [230], [231], and tasks that generate text given textual inputs are discussed in Sections IV-E–IV-G.

E. Summarization

Summarization finds elements of interest in documents in order to produce an encapsulation of the most important content. There are two primary types of summarization: extractive and abstractive. The first focuses on sentence extraction, simplification, reordering, and concatenation to relay the important information in documents using text taken directly from the documents. Abstractive summaries rely on expressing documents' contents through generation-style abstraction, possibly using words never seen in the documents [48].

Rush *et al.* [39] introduced deep learning to summarization, using an FFNN. The language model used an encoder and a generative beam search decoder. The initial input was given directly to both the language model and the convolutional attention-based encoder, which determined contextual importance surrounding the summary sentences and phrases. The performance of the model was comparable to other state-of-the-art models of the time.

As in other areas, attention mechanisms have improved the performance of encoder–decoder models. Krantz and Kalita [232] compared various attention models for abstractive summarization. A state-of-the-art approach developed by Paulus *et al.* [40] used a multiple intratemporal attention encoder mechanism that considered not only the input text tokens but also the output tokens used by the decoder for previously generated words. They also used similar hybrid cross-entropy loss functions to those proposed by Ranzato *et al.* [233], which led to decreases in training and execution by orders of magnitude. Finally, they recommended using strategies seen in reinforcement learning to modify gradients and reduce exposure bias, which has been noted in models trained exclusively via supervised learning. The use of attention also boosted accuracy in the fully convolutional model proposed by Gehring *et al.* [234], who implemented an attention mechanism for each layer.

Zhang *et al.* [235] proposed an encoder–decoder framework, which generated an output sequence based on an input sequence in a two-stage manner. They encoded the input sequence using BERT [71]. The decoder had two stages. In the first stage, a transformer-based decoder generated a draft output sequence. In the second stage, they masked each word of the draft sequence and fed it to BERT, and then by combining the input sequence and the draft representation generated by BERT, they used a transformer-based decoder to predict the refined word for each masked position. Their model achieved state-of-the-art performance on the CNN/Daily Mail and New York Times data sets.

F. Question Answering

Similar to summarization and information extraction, question answering (QA) gathers relevant words, phrases, or sentences from a document. QA coherently returns this information in response to a request. Current methods resemble those of summarization.

Wang *et al.* [41] used a gated attention-based recurrent network to match the question with an answer-containing passage. A self-matching attention mechanism was used to refine the machine representation by mapping the entire passage. Pointer networks were used to predict the location and boundary of an answer. These networks used attention-pooling vector representations of passages, as well as the words being analyzed, to model the critical tokens or phrases necessary.

Multicolumn CNNs were used by Dong *et al.* [236] to automatically analyze questions from multiple viewpoints. Parallel networks were used to extract pertinent information from input questions. Separate networks were used to find context information and relationships and to determine which forms of answers should be returned. The output of these networks was combined and used to rank possible answers.

Santoro *et al.* [237] used relational networks (RNs) for summarization. First proposed by Raposo *et al.* [238], RNs are built upon an MLP architecture, with a focus on relational reasoning, i.e., defining relationships among entities in the data. These feedforward networks implement a similar function among all pairs of objects in order to aggregate correlations among them. For input, the RNs took final LSTM representations of document sentences. These inputs were further paired with a representation of the information request given [237].

BERT [71] achieved state of the art in QA experiments on the SQuAD 1.1 and SQuAD 2.0 data sets. Yang *et al.* [239] demonstrated an end-to-end QA system that integrates BERT with the open-source Anserini IR toolkit. This system can identify answers from a large corpus of Wikipedia articles in an end-to-end fashion, obtaining the best results on a standard benchmark test collection.

G. Machine Translation

Machine translation is the quintessential application of NLP. It involves the use of mathematical and algorithmic techniques to translate the documents in one language to another. Performing effective translation is intrinsically onerous even for humans, requiring proficiency in areas such as morphology, syntax, and semantics, as well as an adept understanding and discernment of cultural sensitivities, for both of the languages (and associated societies) under consideration [48].

The first attempt at NMT was that by Schwenk [240], although neural models had previously been used for the similar task of transliteration, converting certain parts of text, such as proper nouns, into different languages [241]. Schwenk used a feedforward network with seven-word inputs and outputs, padding and trimming when necessary. The ability to translate from a sentence of one length to a sentence of another length came about with the introduction of encoder–decoder models.

The first use of such a model, by Kalchbrenner and Blumson [242], stemmed from the success of continuous recurrent representations in capturing syntax, semantics, and morphology [243] in addition to the ability of RNNs to build robust language models [29]. This original NMT encoder–decoder model used a combination of generative convolutional and recurrent layers to encode and optimize a source language model and cast this into a target language. The model was quickly reworked and further studied by Cho *et al.* [244], and numerous novel and effective advances to this model have since been made [38], [245]. Encoder–decoder models have continuously defined the state of the art, being expanded to contain dozens of layers, with residual connections, attention mechanisms, and even residual attention mechanisms allowing the final decoding layer to attend to the first encoding layer [246]. State-of-the-art results have also been achieved by using numerous convolutional layers in both the encoder and decoder, allowing information to be viewed in several hierarchical layers rather than a multitude of recurrent steps [234]. Such derived models are continually improving, finding answers to the shortcomings of their predecessors and overcoming any need for hand engineering [247]. Recent progress includes effective initialization of decoder hidden states, use of conditional gated attentional cells, removal of bias in embedding layers, use of alternative decoding phases, factorization of embeddings, and test time use of the beam search algorithm [248], [249].

The standard initialization for the decoder state is that proposed by Bahdanau *et al.* [38], using the last backward encoder state. However, as noted by Britz *et al.* [247], using the average of the embedding or annotation layer seems to lead to the best translations. Gated recurrent cells have been the gold standard for sequence-to-sequence tasks, a variation of which is a conditional GRU (cGRU) [248], most effectively utilized with an attention mechanism. A cGRU cell consists of three key components: two GRU transition blocks and an attention mechanism between them. These three blocks combine the previous hidden state, along with the attention context window to generate the next hidden state. Altering the decoding process [38] from *look* at input, *generate* output token, *update* hidden representation to a process of *look, update*, and *generate* can simplify the final decoding. Adding further source attributes, such as morphological segmentation labels, POS tags, and syntactic dependency labels, improves models, and concatenating or factorizing these with embeddings increases robustness further [248], [250]. For remembering long-term dependencies, vertically stacked recurrent units have been the standard, with the optimum number of layers having been determined to be roughly between 2 and 16 [247], depending on the desired input length as well as the presence and density of residual connections. At test time, a beam search algorithm can be used beside the final softmax layer for considering multiple target predictions in a greedy fashion, allowing the best predictions to be found without looking through the entire hypothesis space [249].

In a direction diverging from previous work, Vaswani *et al.* [42] and Ahmed *et al.* [251] proposed discarding the large number of recurrent and convolutional layers and instead focusing exclusively on attention mechanisms to encode a language globally from input to output. Preferring such "self-attention" mechanisms over traditional layers is motivated by the following three principles: reducing the complexity of computations required per layer, minimizing sequential training steps, and, finally, abating the path length from input to output and its handicap on the learning of the long-range dependencies that are necessary in many sequencing tasks [252]. Apart from increased accuracy across translation tasks, self-attention models allow more parallelization throughout architectures, decreasing the training times and minimizing necessary sequential steps. At time of writing, the state-of-the-art model generating the best results for English to German and English to French on the International Workshop on Spoken Language Translation (IWSLT) 2014 test corpus [253] is that of Medina and Kalita [254], which modified the model proposed by Vaswani to use parallel self-attention mechanisms, rather than stacking them as was done in the original model. In addition to improving BLEU scores [255], this also reduced training times. Ghazvininejad *et al.* [256] recently applied BERT to the machine translation task using constant-time models. They were able to achieve relatively competitive performance in a fraction of the time. Lample and Conneau [257] attained state-of-the-art results, performing unsupervised machine translation using multiple languages in their language model pretraining.

Several of the recent state-of-the-art models were examined by Chen *et al.* [258]. The models were picked apart to determine which features were truly responsible for their strength and to provide a fair comparison. Hybrid models were then created using this knowledge, and incorporating the best parts of each previous model, outperforming the previous models. In addition to creating two models with both a self-attentive component and a recurrent component (in one model, they were stacked, in the other parallel), they determined four techniques that they believe should always be employed, as they are crucial to some models, at best, and neutral to all models examined, at worst. These are label smoothing, multi-head attention, layer normalization, and synchronous training. Another study, by Denkowski and Neubig [259], examined a number of other techniques, recommending three: using Adam optimization, restarting multiple times, with learning rate annealing; performing subword translation; and using an ensemble of decoders. Furthermore, they tested a number of common techniques on models that were strong to begin and determined that three of the four provided no additional benefits to, or actually hurt, the model, those three being lexicon bias (priming the outputs with directly translated words), pretranslation (using translations from another model, usually of lower quality, as additional input), and dropout. They did find, however, that data bootstrapping (using phrases that are parts of training examples as additional independent smaller samples) was advantageous even to models that are already high performing. They recommended that future developments be tested on top-performing models in order to determine their realm of effectiveness.

In addition to studies presenting recommendations, one study has listed a number of challenges facing the field [260]. While neural machine translation models are superior to other forms of statistical machine translation models (as well as rule-based models), they require significantly more data, perform poorly outside of the domain in which they are trained, fail to handle rare words adequately, and do not do well with long sentences (more than about 60 words). Furthermore, attention mechanisms do not perform as well as their statistical counterparts for aligning words, and beam searches used for decoding only work when the search space is small. Surely, these six drawbacks will be, or in some cases, will continue to be, the focus of much research in the coming years. In addition, as mentioned in Section III-D2, NMT models still struggle with some semantic concepts, which will also be a likely area of focus in the upcoming years. While examining some of these failings of NMT can help, predicting the future of research and development in the field is nearly impossible.

New models and methods are being reported daily with far too many advancements to survey, and state-of-the-art practices are becoming outdated in a matter of months. Notable recent advancements include using caching to provide networks with greater context than simply the individual sentences being translated [261], the ability to better handle rare words [262], [263], and the ability to translate to and from understudied languages, such as those that are polysynthetic [264]. In addition, work has been conducted on the selection, sensitivity, and tuning of hyperparameters [265], denoising of data [266], and a number of other important topics surrounding NMT. Finally, a new branch of machine translation has been opened up by groundbreaking research: multilingual translation.

A fairly recent study [267] showed that a single, simple (but large) neural network could be trained to convert a number (up to at least 12) of different languages to each other, automatically recognizing the source language and simply needing an input token to identify the output language. Furthermore, the model was found to be capable of understanding, at least somewhat, multilingual input, and of producing mixed outputs when multiple language tokens are given, sometimes even in languages related to, but not actually, those selected. This suggests that DNNs may be capable of learning universal representations for information, independent of language, and even more, that they might possibly be capable of learning some etymology and relationships between and among families of different languages.

H. Summary of Deep Learning NLP Applications

Numerous other applications of NLP exist including grammar correction, as seen in word processors, and author mimicking, which, given sufficient data, generates text replicating the style of a particular writer. Many of these applications are infrequently used, understudied, or not yet exposed to deep learning. However, the area of sentiment analysis should be noted, as it is becoming increasingly popular and utilizing deep learning. In large part a semantic task, it is the extraction of a writer's sentiment—their positive, negative, or neutral inclination toward some subject or idea [268]. Applications are varied, including product research, futures prediction, social media analysis, and classification of spam [269], [270]. The current state of the art uses an ensemble, including both LSTMs and CNNs [271].

This section has provided a number of select examples of the applied usages of deep learning in NLP. Countless studies have been conducted in these and similar areas, chronicling the ways in which deep learning has facilitated the successful use of natural language in a wide variety of applications. Only a minuscule fraction of such work has been referred to in this survey.

While more specific recommendations for practitioners have been discussed in some individual sections, the current trend in state-of-the-art models in all application areas is to use pretrained stacks of transformer units in some configuration, whether in encoder–decoder configurations or just as encoders. Thus, self-attention, which is the mainstay of transformer, has become the norm, along with cross attention between the encoder and decoder units, if decoders are present. In fact, in many recent articles, if not most, transformers have begun to replace LSTM units that were preponderant just a few months ago. Pretraining of these large transformer models has also become the accepted way to endow a model with generalized knowledge of language. Models such as BERT, which have been trained on corpora of billions of words, are available for download, thus providing a practitioner with a model that possesses a great amount of general knowledge of language already. A practitioner can further train it with one's own general corpora, if desired, but such training is not always necessary, considering the enormous sizes of the pretraining that downloaded models have received. To train a model to perform a certain task well, the last step that a practitioner must go through is to use available downloadable task-specific corpora or build one's own task-specific corpus. This last training step is usually supervised. It is also recommended that if several tasks are to be performed, multitask training is used wherever possible.

V. Conclusion

Early applications of NLP included a well-acclaimed but simpleminded algebra word problem solver program called STUDENT [272], as well as interesting but severely constrained conversational systems such as Eliza, which acted as a "psychotherapist" [273], and another that conversed about manipulating blocks in a microworld [274]. Nowadays, highly advanced applications of NLP are ubiquitous. These include Google's and Microsoft's machine translators, which translate more or less competently from a language to scores of other languages, as well as a number of devices which process voice commands and respond in like. The emergence of these sophisticated applications, particularly in deployed settings, acts as a testament to the impressive accomplishments that have been made in this domain over the last sixty or so years. Without a doubt, incredible progress has taken place, particularly in the last several years.

As has been shown, this recent progress has a clear causal relationship with the remarkable advances in ANNs. Considered an "old" technology just a decade ago, these machine learning constructs have ushered in progress at an unprecedented rate, breaking performance records in myriad tasks in miscellaneous fields. In particular, deep neural architectures have instilled models with higher performance in natural language tasks, in terms of "imperfect" metrics. Consolidating the analysis of all the models surveyed, a few general trends can be surmized. Both convolutional and recurrent specimens had contributed to the state of the art in the recent past; however, of very late, stacks of attention-powered transformer units as encoders and often decoders have consistently produced superior results across the rich and varying terrain of the NLP field. These models are generally heavily pretrained on general language knowledge in an unsupervised or supervised manner and somewhat lightly trained on specific tasks in a supervised fashion. Second, attention mechanisms alone, without recurrences or convolutions, seem to provide the best connections between encoders and decoders. Third, forcing networks to examine different features (by performing multiple tasks) usually improves results. Finally, while highly engineering networks usually optimizes results, there is no substitute for cultivating networks with large quantities of high-quality data, although pretraining on large generic corpora seems to help immensely. Following from this final observation, it may be useful to direct more research effort toward pretraining methodologies, rather than developing highly specialized components to squeeze the last drops of performance from complex models.

While the numerous stellar architectures being proposed each month are highly competitive, muddling the process of identifying a winning architecture, the methods of evaluation used add just as much complexity to the problem. Data sets used to evaluate new models are often generated specifically for those models and are then used only several more times, if at all, although consolidated data sets encompassing several tasks such as GLUE [275] have started to emerge. As the features and sizes of these data sets are highly variable, this makes comparison difficult. Most subfields of NLP, as well as the field as a whole, would benefit from extensive, large-scale discussions regarding the necessary contents of such data sets, followed by the compilation of such sets. In addition to high variability in evaluation data, there are numerous metrics used to evaluate performance on each task. Oftentimes, comparing similar models is difficult because different metrics are reported for each. Agreement on particular sets of metrics would go a long way toward ensuring clear comparisons in the field.

Furthermore, metrics are usually only reported for the best case, with few mentions of average cases and variability, or of worst cases. While it is important to understand the possible performance of new models, it is just as important to understand the standard performance. If models produce highly variable results, they may take many attempts to train to the cutting-edge levels reported. In most cases, this is undesirable, and models that can be consistently trained to relatively high levels of performance are preferable. While increasingly large numbers of randomized parameters do reduce variation in performance, some variance will always exist, necessitating the reporting of more than just best case metrics.

One final recommendation for future work is that it is directed toward a wider variety of languages than it is at present. Currently, the vast majority of research in NLP is conducted on the English language, with another sizeable portion using Mandarin Chinese. In translation tasks, English is almost always either the input or output language, with the other end usually being one of a dozen major European or Eastern Asian languages. This neglects entire families of languages, as well as the people who speak them. Many linguistic intricacies may not be expressed in any of the languages used and, therefore, are not captured in current NLP software. Furthermore, there are thousands of languages spoken throughout the world, with at least 80 spoken by more than 10 million people, meaning that current research excludes an immense segment of humankind. Collection and validation of data in underanalyzed languages, as well as testing NLP models using such data, will be a tremendous contribution to not only the field of NLP but also to human society as a whole.

Due to the small amounts of data available in many languages, the authors do not foresee the complete usurpation of traditional NLP models by deep learning any time in the near future. Deep learning models (and even shallow ANNs) are extremely data hungry. Contrastingly, many traditional models require only relatively small amounts of training data. However, looking further forward, it can be anticipated that deep learning models will become the norm in computational linguistics, with pretraining and transfer learning playing highly impactful roles. Collobert *et al.* [9] sparked the deep learning revolution in NLP, although one of the key contributions of their work—that of a single unified model—was not realized widely. Instead, neural networks were introduced into traditional NLP tasks and are only now reconnecting. In the field of parsing, for example, most models continue to implement nonneural structures, simply using ANNs on the side to make the decisions that were previously done using rules and probability models. While more versatile and general architectures are obviously becoming more and more of a reality, understanding the abstract concepts handled by such networks is important to understanding how to build and train better networks. Furthermore, as abstraction is a hallmark of human intelligence, understanding of the abstractions that take place inside an ANN may aid in the understanding of human intelligence and the processes that underlie it. Just as human linguistic ability is only a piece of our sentience, so is linguistic processing just a small piece of artificial intelligence. Understanding how such components are interrelated is important in constructing more complete AI systems, and creating a unified NLP architecture is another step toward making such a system a reality.

This goal will also be aided by further advances in computational equipment. While GPUs have significantly improved the ability to train deep networks, they are only a step in the right direction [276]. The next step is the wider availability of chips designed specifically for this purpose, such as Google's Tensor Processing Unit (TPU), Microsoft's Catapult, and Intel's Lake Crest [277]. Ultimately, ANNs implemented in traditional von Neumann style computers may not be able to reach their full potential. Luckily, another old line of work in computer science and engineering has seen a resurgance in recent years: neuromorphic computing. With neuromorphic chips, which implement neural structures at the hardware level, expected much more widely in the coming years [278], the continuation of deep learning and the longevity of its success can be highly anticipated, ensuring the opportunity for sustained progress in NLP.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers of this survey for their valuable comments, critiques, and recommendations. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] K. S. Jones, "Natural language processing: A historical review," in *Current Issues in Computational Linguistics: In Honour of Don Walker*. Dordrecht, The Netherlands: Springer, 1994, pp. 3–16.

[2] E. D. Liddy, "Natural language processing," in *Encyclopedia of Library and Information Science*, 2nd ed. New York, NY, USA: Marcel Decker, Inc., 2001.

[3] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, "Deep learning with cots HPC systems," in *Proc. ICML*, 2013, pp. 1337–1345.

[4] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. ICML*, 2009, pp. 873–880.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[8] D. C. Ciresan *et al.*, "Flexible, high performance convolutional neural networks for image classification," in *Proc. IJCAI*, 2011, vol. 22, no. 1, p. 1237.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[10] Y. Goldberg, "Neural network methods for natural language processing," *Synth. Lect. Hum. Lang. Technol.*, vol. 10, no. 1, pp. 1–309, 2017.

[11] Y. Liu and M. Zhang, "Neural network methods for natural language processing," *Comput. Linguistics*, vol. 44, no. 1, pp. 193–195, Mar. 2018.

[12] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[13] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," UCSD, La Jolla, CA, USA, Tech. Rep. ICS-8506, 1985.

[14] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[16] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[17] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognit.*, vol. 15, no. 6, pp. 455–469, Jan. 1982.

[18] Y. LeCun *et al.*, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10. Cambridge, MA, USA: MIT Press, 1995.

[19] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014, *arXiv:1404.5997*. [Online]. Available: http://arxiv.org/abs/1404.5997

[20] Y. Kim, "Convolutional neural networks for sentence classification," 2014, *arXiv:1408.5882*. [Online]. Available: http://arxiv.org/abs/1408.5882

[21] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," 2014, *arXiv:1404.2188*. [Online]. Available: http://arxiv.org/abs/1404.2188

[22] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proc. COLING*, 2014, pp. 69–78.

[23] D. Zeng *et al.*, "Relation classification via convolutional deep neural network," in *Proc. COLING*, 2014, pp. 2335–2344.

[24] M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural-network model for control and learning of voluntary movement," *Biol. Cybern.*, vol. 57, no. 3, pp. 169–185, Oct. 1987.

[25] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in *Proc. IEEE Int. Conf Neural Netw.*, vol. 1, Jun. 1996, pp. 347–352.

[26] R. Socher, E. Huang, J. Pennin, C. Manning, and A. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Proc. NIPS*, 2011, pp. 801–809.

[27] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[28] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.

[29] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, vol. 2, 2010, p. 3.

[30] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE ICASSP*, May 2011, pp. 5528–5531.

[31] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2011, pp. 196–201.

[32] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Comput.*, vol. 4, no. 2, pp. 234–242, Mar. 1992.

[33] S. El Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Proc. NIPS*, 1996, pp. 493–499.

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

[35] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[36] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*. [Online]. Available: http://arxiv.org/abs/1409.1259

[37] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[38] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[39] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," 2015, *arXiv:1509.00685*. [Online]. Available: http://arxiv.org/abs/1509.00685

[40] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*. [Online]. Available: http://arxiv.org/abs/1705.04304

[41] W. Wang, N. Yang, F. Wei, B. Chang, and M. Zhou, "Gated self-matching networks for reading comprehension and question answering," in *Proc. ACL*, vol. 1, 2017, pp. 189–198.

[42] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 6000–6010.

[43] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[44] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 807–814.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Jun. 2016, pp. 770–778.

[46] R. Kumar Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015, *arXiv:1505.00387*. [Online]. Available: http://arxiv.org/abs/1505.00387

[47] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE CVPR*, Jul. 2017, vol. 1, no. 2, pp. 4700–4708.

[48] D. Jurafsky and J. Martin, *Speech & Language Processing*. London, U.K.: Pearson Education, 2000.

[49] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[50] W. De Mulder, S. Bethard, and M.-F. Moens, "A survey on the application of recurrent neural networks to statistical language modeling," *Comput. Speech Lang.*, vol. 30, no. 1, pp. 61–98, Mar. 2015.

[51] R. Iyer, M. Ostendorf, and M. Meteer, "Analyzing and predicting language model improvements," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 1997, pp. 254–261.

[52] S. F. Chen, D. Beeferman, and R. Rosenfeld, "Evaluation metrics for language models," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., Jan. 2008. [Online]. Available: https://kilthub.cmu.edu/articles/Evaluation_Metrics_For_Language_Models/6605324, doi: 10.1184/R1/6605324.v1.

[53] P. Clarkson and T. Robinson, "Improved language modelling through better language model evaluation measures," *Comput. Speech Lang.*, vol. 15, no. 1, pp. 39–53, Jan. 2001.

[54] M. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: The Penn Treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[55] C. Chelba *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," 2013, *arXiv:1312.3005*. [Online]. Available: http://arxiv.org/abs/1312.3005

[56] M. Daniluk, T. Rocktäschel, J. Welbl, and S. Riedel, "Frustratingly short attention spans in neural language modeling," 2017, *arXiv:1702.04521*. [Online]. Available: http://arxiv.org/abs/1702.04521

[57] K. Beneš, M. K. Baskar, and L. Burget, "Residual memory networks in language modeling: Improving the reputation of feed-forward networks," in *Proc. InterSpeech*, Aug. 2017, pp. 284–288.

[58] N.-Q. Pham, G. Kruszewski, and G. Boleda, "Convolutional neural network language models," in *Proc. EMNLP*, 2016, pp. 1153–1162.

[59] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: http://arxiv.org/abs/1312.4400

[60] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *Proc. AAAI*, 2016, pp. 2741–2749.

[61] J. Botha and P. Blunsom, "Compositional morphology for word representations and language modelling," in *Proc. ICML*, 2014, pp. 1899–1907.

[62] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, *arXiv:1409.2329*. [Online]. Available: http://arxiv.org/abs/1409.2329

[63] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," 2016, *arXiv:1602.02410*. [Online]. Available: https://arxiv.org/abs/1602.02410

[64] Y. Ji, T. Cohn, L. Kong, C. Dyer, and J. Eisenstein, "Document context language models," 2015, *arXiv:1511.03962*. [Online]. Available: http://arxiv.org/abs/1511.03962

[65] N. Shazeer, J. Pelemans, and C. Chelba, "Sparse non-negative matrix language modeling for skip-grams," in *Proc. InterSpeech*, 2015, pp. 1428–1432.

[66] W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson, "Scaling recurrent neural network language models," in *Proc. IEEE ICASSP*, Apr. 2015, pp. 5391–5395.

[67] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781

[68] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, 2013, pp. 3111–3119.

[69] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. (2018). *Improving Language Understanding by Generative Pre-Training*. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[70] M. E. Peters *et al.*, "Deep contextualized word representations," 2018, *arXiv:1802.05365*. [Online]. Available: http://arxiv.org/abs/1802.05365

[71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805

[72] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," 2019, *arXiv:1901.11504*. [Online]. Available: http://arxiv.org/abs/1901.11504

[73] X. Liu, Y. Shen, K. Duh, and J. Gao, "Stochastic answer networks for machine reading comprehension," 2017, *arXiv:1712.03556*. [Online]. Available: http://arxiv.org/abs/1712.03556

[74] X. Liu, K. Duh, and J. Gao, "Stochastic answer networks for natural language inference," 2018, *arXiv:1804.07888*. [Online]. Available: http://arxiv.org/abs/1804.07888

[75] T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," in *Proc. ACL*, 2019, pp. 3428–3448.

[76] T. Luong, R. Socher, and C. Manning, “Better word representations with recursive neural networks for morphology,” in *Proc. CoNLL*, 2013, pp. 104–113.

[77] L. Finkelstein *et al.*, “Placing search in context: The concept revisited,” in *Proc. Int. Conf. World Wide Web*, 2001, pp. 406–414.

[78] M. Creutz and K. Lagus, “Unsupervised models for morpheme segmentation and morphology learning,” *ACM Trans. Speech Lang. Process.*, vol. 4, no. 1, pp. 1–34, Jan. 2007.

[79] G. A. Miller and W. G. Charles, “Contextual correlates of semantic similarity,” *Lang. Cognit. Processes*, vol. 6, no. 1, pp. 1–28, Jan. 1991.

[80] H. Rubenstein and J. B. Goodenough, “Contextual correlates of synonymy,” *Commun. ACM*, vol. 8, no. 10, pp. 627–633, Oct. 1965.

[81] E. Huang, R. Socher, C. Manning, and A. Ng, “Improving word representations via global context and multiple word prototypes,” in *Proc. ACL*, vol. 1, 2012, pp. 873–882.

[82] Y. Belinkov, N. Durrani, F. Dalvi, H. Sajjad, and J. Glass, “What do neural machine translation models learn about morphology?” 2017, *arXiv:1704.03471*. [Online]. Available: http://arxiv.org/abs/1704.03471

[83] M. Cettolo, C. Girardi, and M. Federico, “WIT3: Web inventory of transcribed and translated talks,” in *Proc. Conf. Eur. Assoc. Mach. Transl.*, 2012, pp. 261–268.

[84] M. Cettolo, “An Arabic–Hebrew parallel corpus of TED talks,” 2016, *arXiv:1610.00572*. [Online]. Available: http://arxiv.org/abs/1610.00572

[85] H. Morita, D. Kawahara, and S. Kurohashi, “Morphological analysis for unsegmented languages using recurrent neural network language model,” in *Proc. EMNLP*, 2015, pp. 2292–2297.

[86] D. Kawahara and S. Kurohashi, “Case frame compilation from the Web using high-performance computing,” in *Proc. LREC*, 2006, pp. 1344–1347.

[87] D. Kawahara, S. Kurohashi, and K. Hasida, “Construction of a Japanese relevance-tagged corpus,” in *Proc. LREC*, 2002, pp. 2008–2013.

[88] M. Hangyo, D. Kawahara, and S. Kurohashi, “Building a diverse document leads corpus annotated with semantic relations,” in *Proc. Pacific–Asia Conf. Lang., Inf., Comput.*, 2012, pp. 535–544.

[89] M. Dehouck and P. Denis, “A framework for understanding the role of morphology in universal dependency parsing,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 2864–2870.

[90] A. More *et al.*, “CONLL-UL: Universal morphological lattices for universal dependency parsing,” in *Proc. 11th Int. Conf. Lang. Resour. Eval.*, 2018, pp. 3847–3853.

[91] J. Nivre, “An efficient algorithm for projective dependency parsing,” in *Proc. Int. Workshop Parsing Technol.*, 2003, pp. 149–160.

[92] J. Nivre, “Incrementality in deterministic dependency parsing,” in *Proc. Workshop Incremental Parsing Bringing Eng. Cognition Together (IncrementParsing)*, 2004, pp. 50–57.

[93] J. Nivre, M. Kuhlmann, and J. Hall, “An improved oracle for dependency parsing with online reordering,” in *Proc. 11th Int. Conf. Parsing Technol. (IWPT)*, 2009, pp. 73–76.

[94] R. Socher *et al.*, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proc. EMNLP*, 2013, pp. 1631–1642.

[95] R. Socher, J. Bauer, A. Y. Ng, and C. D. Manning, “Parsing with compositional vector grammars,” in *Proc. ACL*, vol. 1, Aug. 2013, pp. 455–465.

[96] T. Fujisaki, F. Jelinek, J. Cocke, E. Black, and T. Nishino, “A probabilistic parsing method for sentence disambiguation,” in *Current Issues in Parsing Technology*. Boston, MA, USA: Springer, 1991, pp. 139–152.

[97] F. Jelinek, J. Lafferty, and R. Mercer, “Basic methods of probabilistic context free grammars,” in *Speech Recognition and Understanding*. Berlin, Germany: Springer, 1992, pp. 345–360.

[98] P. Le and W. Zuidema, “The inside-outside recursive neural network model for dependency parsing,” in *Proc. EMNLP*, 2014, pp. 729–739.

[99] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, “Grammar as a foreign Language,” in *Proc. NIPS*, 2015, pp. 2773–2781.

[100] S. Petrov and R. McDonald, “Overview of the 2012 shared task on parsing the Web,” in *Proc. Notes 1st Workshop Syntactic Anal. Non-Canonical Lang.*, vol. 59, 2012, pp. 1–8.

[101] J. Judge, A. Cahill, and J. Van Genabith, “Questionbank: Creating a corpus of parse-annotated questions,” in *Proc. COLING*, 2006, pp. 497–504.

[102] P. Stenetorp, “Transition-based dependency parsing using recursive neural networks,” in *Proc. Deep Learn. Workshop Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2013. [Online]. Available: https://pontus.stenetorp.se/res/pdf/stenetorp2013transition.pdf

[103] M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre, “The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies,” in *Proc. CONLL*, 2008, pp. 159–177.

[104] D. Chen and C. Manning, “A fast and accurate dependency parser using neural networks,” in *Proc. EMNLP*, 2014, pp. 740–750.

[105] H. Zhou, Y. Zhang, S. Huang, and J. Chen, “A neural probabilistic structured-prediction model for transition-based dependency parsing,” in *Proc. ACL IJCNLP*, vol. 1, 2015, pp. 1213–1222.

[106] D. Weiss, C. Alberti, M. Collins, and S. Petrov, “Structured training for neural network transition-based parsing,” 2015, *arXiv:1506.06158*. [Online]. Available: http://arxiv.org/abs/1506.06158

[107] Z. Li, M. Zhang, and W. Chen, “Ambiguity-aware ensemble training for semi-supervised dependency parsing,” in *Proc. ACL*, vol. 1, 2014, pp. 457–467.

[108] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, “Transition-based dependency parsing with stack long short-term memory,” 2015, *arXiv:1505.08075*. [Online]. Available: http://arxiv.org/abs/1505.08075

[109] M.-C. de Marneffe and C. D. Manning, “The Stanford typed dependencies representation,” in *Proc. Coling, Workshop Cross-Framework Cross-Domain Parser Eval. (CrossParser)*, 2008, pp. 1–8.

[110] N. Xue, F. Xia, F.-D. Chiou, and M. Palmer, “The Penn Chinese TreeBank: Phrase structure annotation of a large corpus,” *Natural Lang. Eng.*, vol. 11, no. 2, pp. 207–238, Jun. 2005.

[111] D. Andor *et al.*, “Globally normalized transition-based neural networks,” 2016, *arXiv:1603.06042*. [Online]. Available: http://arxiv.org/abs/1603.06042

[112] Y. Wang, W. Che, J. Guo, and T. Liu, “A neural transition-based approach for semantic dependency graph parsing,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5561–5568.

[113] J. Cross and L. Huang, “Incremental parsing with minimal features using bi-directional LSTM,” 2016, *arXiv:1606.06406*. [Online]. Available: http://arxiv.org/abs/1606.06406

[114] K. Sheng Tai, R. Socher, and C. D. Manning, “Improved semantic representations from tree-structured long short-term memory networks,” 2015, *arXiv:1503.00075*. [Online]. Available: http://arxiv.org/abs/1503.00075

[115] S. Oepen *et al.*, “SemEval 2015 task 18: Broad-coverage semantic dependency parsing,” in *Proc. 9th Int. Workshop Semantic Eval. (SemEval)*, 2015, pp. 915–926.

[116] W. Che, Y. Shao, T. Liu, and Y. Ding, “SemEval-2016 task 9: Chinese semantic dependency parsing,” in *Proc. 10th Int. Workshop Semantic Eval. (SemEval)*, 2016, pp. 378–384.

[117] W.-T. Yih, X. He, and C. Meek, “Semantic parsing for single-relation question answering,” in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2014, pp. 643–648.

[118] J. Krishnamurthy, P. Dasigi, and M. Gardner, “Neural semantic parsing with type constraints for semi-structured tables,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1516–1526.

[119] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, “Recurrent neural network grammars,” 2016, *arXiv:1602.07776*. [Online]. Available: http://arxiv.org/abs/1602.07776

[120] D. K. Choe and E. Charniak, “Parsing as language modeling,” in *Proc. EMNLP*, 2016, pp. 2331–2336.

[121] D. Fried, M. Stern, and D. Klein, “Improving neural parsing by disentangling model combination and reranking effects,” 2017, *arXiv:1707.03058*. [Online]. Available: http://arxiv.org/abs/1707.03058

[122] T. Dozat and C. D. Manning, “Simpler but more accurate semantic dependency parsing,” 2018, *arXiv:1807.01396*. [Online]. Available: http://arxiv.org/abs/1807.01396

[123] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, “Deep semantic role labeling with self-attention,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4929–4936.

[124] L. Duong, H. Afshar, D. Estival, G. Pink, P. Cohen, and M. Johnson, “Active learning for deep semantic parsing,” in *Proc. 56th Annu. Meeting Assoc. for Comput. Linguistics*, vol. 2, 2018, pp. 43–48.

[125] J. Nivre, “Towards a universal grammar for natural language processing,” in *Proc. Int. Conf. Intell. Text Process. Comput. Linguistics*. Cham, Switzerland: Springer, 2015, pp. 3–16.

[126] D. Zeman *et al.*, “CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies,” in *Proc. CoNLL Shared Task, Multilingual Parsing Raw Text Universal Dependencies*, 2018, pp. 1–21.

[127] D. Hershcovich, O. Abend, and A. Rappoport, “Universal dependency parsing with a general transition-based DAG parser,” 2018, *arXiv:1808.09354*. [Online]. Available: http://arxiv.org/abs/1808.09354

[128] T. Ji, Y. Liu, Y. Wang, Y. Wu, and M. Lan, "AntNLP at CoNLL 2018 shared task: A graph-based parser for universal dependency parsing," in *Proc. CoNLL Shared Task, Multilingual Parsing Raw Text Universal Dependencies*, 2018, pp. 248–255.

[129] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, "Universal dependency parsing from scratch," 2019, *arXiv:1901.10457*. [Online]. Available: http://arxiv.org/abs/1901.10457

[130] Y. Liu, Y. Zhu, W. Che, B. Qin, N. Schneider, and N. A. Smith, "Parsing tweets into universal dependencies," 2018, *arXiv:1804.08228*. [Online]. Available: http://arxiv.org/abs/1804.08228

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

[132] Z. S. Harris, "Distributional structure," *Word*, vol. 10, nos. 2–3, pp. 146–162, 1954.

[133] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Proc. NIPS*, 2014, pp. 2042–2050.

[134] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Mach. Learn.*, vol. 94, no. 2, pp. 233–259, Feb. 2014.

[135] W. Yin and H. Schütze, "Convolutional neural network for paraphrase identification," in *Proc. NAACL, HLT*, 2015, pp. 901–911.

[136] B. Dolan, C. Quirk, and C. Brockett, "Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources," in *Proc. COLING*, 2004, p. 350.

[137] H. He, K. Gimpel, and J. Lin, "Multi-perspective sentence similarity modeling with convolutional neural networks," in *Proc. EMNLP*, 2015, pp. 1576–1586.

[138] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, "SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment," in *Proc. 8th Int. Workshop Semantic Eval. (SemEval)*, 2014, pp. 1–8.

[139] E. Agirre, M. Diab, D. Cer, and A. Gonzalez-Agirre, "SemEval-2012 task 6: A pilot on semantic textual similarity," in *Proc. Joint Conf. Lexical Comput. Semantics*, vol. 1, 2012, pp. 385–393.

[140] H. He and J. Lin, "Pairwise word interaction modeling with deep neural networks for semantic similarity measurement," in *Proc. NAACL, HLT*, 2016, pp. 937–948.

[141] E. Agirre *et al.*, "SemEval-2014 task 10: Multilingual semantic textual similarity," in *Proc. 8th Int. Workshop Semantic Eval. (SemEval)*, 2014, pp. 81–91.

[142] Y. Yang, W.-T. Yih, and C. Meek, "WikiQA: A challenge dataset for open-domain question answering," in *Proc. EMNLP*, 2015, pp. 2013–2018.

[143] M. Wang, N. A. Smith, and T. Mitamura, "What is the jeopardy model? A quasi-synchronous grammar for QA," in *Proc. Joint EMNLP CoNLL*, 2007, pp. 22–32.

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

[145] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Project Rep., Stanford, CA, USA, Tech. Rep. CS224N, 2009, vol. 1, no. 12.

[146] X. Li and D. Roth, "Learning question classifiers," in *Proc. COLING*, vol. 1, 2002, pp. 1–7.

[147] A. Poliak, Y. Belinkov, J. Glass, and B. Van Durme, "On the evaluation of semantic phenomena in neural machine translation using natural language inference," 2018, *arXiv:1804.09779*. [Online]. Available: http://arxiv.org/abs/1804.09779

[148] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," 2017, *arXiv:1704.05426*. [Online]. Available: http://arxiv.org/abs/1704.05426

[149] N. Nangia, A. Williams, A. Lazaridou, and S. R. Bowman, "The RepEval 2017 shared task: Multi-genre natural language inference with sentence representations," 2017, *arXiv:1707.08172*. [Online]. Available: http://arxiv.org/abs/1707.08172

[150] A. S. White, P. Rastogi, K. Duh, and B. Van Durme, "Inference is everything: Recasting semantic resources into a unified evaluation framework," in *Proc. IJCNLP*, vol. 1, 2017, pp. 996–1005.

[151] E. Pavlick, T. Wolfe, P. Rastogi, C. Callison-Burch, M. Dredze, and B. Van Durme, "FrameNet+: Fast paraphrastic tripling of FrameNet," in *Proc. ACL*, vol. 2, 2015, pp. 408–413.

[152] A. Rahman and V. Ng, "Resolving complex cases of definite pronouns: The winograd schema challenge," in *Proc. Joint EMNLP CoNLL*, 2012, pp. 777–789.

[153] D. Reisinger, R. Rudinger, F. Ferraro, C. Harman, K. Rawlins, and B. Van Durme, "Semantic proto-roles," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 475–488, Dec. 2015.

[154] A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. Van Durme, "Hypothesis only baselines in natural language inference," 2018, *arXiv:1805.01042*. [Online]. Available: http://arxiv.org/abs/1805.01042

[155] J. Herzig and J. Berant, "Neural semantic parsing over multiple knowledge-bases," 2017, *arXiv:1702.01569*. [Online]. Available: http://arxiv.org/abs/1702.01569

[156] Y. Wang, J. Berant, and P. Liang, "Building a semantic parser overnight," in *Proc. ACL IJCNLP*, vol. 1, 2015, pp. 1332–1342.

[157] G. Brunner, Y. Wang, R. Wattenhofer, and M. Weigelt, "Natural language multitasking: Analyzing and improving syntactic saliency of hidden representations," 2018, *arXiv:1801.06024*. [Online]. Available: http://arxiv.org/abs/1801.06024

[158] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *Proc. MT Summit*, vol. 5, 2005, pp. 79–86.

[159] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[160] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A nucleus for a Web of open data," in *The Semantic Web*. Berlin, Germany: Springer, 2007, pp. 722–735.

[161] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, Dec. 2017.

[162] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[163] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[164] T. Kenter, A. Borisov, C. Van Gysel, M. Dehghani, M. de Rijke, and B. Mitra, "Neural networks for information retrieval," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 1403–1406.

[165] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for Web search using clickthrough data," in *Proc. ACM CIKM*, 2013, pp. 2333–2338.

[166] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for Web search," in *Proc. 23rd Int. Conf. World Wide Web (WWW Companion)*, 2014, pp. 373–374.

[167] Z. Lu and H. Li, "A deep architecture for matching short texts," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 1367–1375.

[168] L. Pang, Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng, "Text matching as image recognition," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2793–2799.

[169] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2016, pp. 55–64.

[170] H. Zamani, M. Dehghani, W. B. Croft, E. Learned-Miller, and J. Kamps, "From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2018, pp. 497–506.

[171] S. MacAvaney, A. Yates, A. Cohan, and N. Goharian, "CEDR: Contextualized embeddings for document ranking," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2019, pp. 1101–1104.

[172] J. Cowie and W. Lehnert, "Information extraction," *Commun. ACM*, vol. 39, no. 1, pp. 80–91, 1996.

[173] J. Hammerton, "Named entity recognition with long short-term memory," in *Proc. HLT-NAACL*, vol. 4, 2003, pp. 172–175.

[174] C. Nogueira dos Santos and V. Guimarães, "Boosting named entity recognition with neural character embeddings," 2015, *arXiv:1505.05008*. [Online]. Available: http://arxiv.org/abs/1505.05008

[175] D. Santos, N. Seco, N. Cardoso, and R. Vilela, "Harem: An advanced ner evaluation contest for portuguese," in *Proc. LREC*, Genoa, Italy, 2006, pp. 1986–1991.

[176] X. Carreras, L. Màrquez, and L. Padró, "Named entity extraction using AdaBoost," in *Proc. CoNLL*, 2002, pp. 1–4.

[177] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," 2015, *arXiv:1511.08308*. [Online]. Available: http://arxiv.org/abs/1511.08308

[178] E. F. Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. HLT-NAACL*, vol. 4, 2003, pp. 142–147.

[179] E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel, "Ontonotes: The 90% solution," in *Proc. Hum. Lang. Technol. Conf. Companion*, 2006, pp. 57–60.

[180] S. Pradhan *et al.*, "Towards robust linguistic analysis using ontonotes," in *Proc. CoNLL*, 2013, pp. 143–152.

[181] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," 2016, *arXiv:1603.01360*. [Online]. Available: http://arxiv.org/abs/1603.01360

[182] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Mach. Learn.* San Mateo, CA, USA: Morgan Kaufmann, 2001, pp. 282–289.

[183] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. COLING*, 2018, pp. 1638–1649.

[184] Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao, "Event extraction via dynamic multi-pooling convolutional neural networks," in *Proc. ACL*, vol. 1, 2015, pp. 167–176.

[185] T. H. Nguyen, K. Cho, and R. Grishman, "Joint event extraction via recurrent neural networks," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 300–309.

[186] X. Liu, H. Huang, and Y. Zhang, "Open domain event extraction using neural latent variable models," 2019, *arXiv:1906.06947*. [Online]. Available: http://arxiv.org/abs/1906.06947

[187] S. Zheng *et al.*, "Joint entity and relation extraction based on a hybrid neural network," *Neurocomputing*, vol. 257, pp. 59–66, Sep. 2017.

[188] M. Sun, X. Li, X. Wang, M. Fan, Y. Feng, and P. Li, "Logician: A unified End-to-End neural approach for open-domain information extraction," in *Proc. 11th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2018, pp. 556–564.

[189] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," 2016, *arXiv:1601.04811*. [Online]. Available: http://arxiv.org/abs/1601.04811

[190] C. Lin, T. Miller, D. Dligach, S. Bethard, and G. Savova, "A bert-based universal model for both within-and cross-sentence clinical temporal relation extraction," in *Proc. Clin. NLP Workshop*, 2019, pp. 65–71.

[191] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, vol. 1, 2017, pp. 1107–1116.

[192] M. Jiang *et al.*, "Text classification based on deep belief network and softmax regression," *Neural Comput. Appl.*, vol. 29, no. 1, pp. 61–70, Jan. 2018.

[193] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[194] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.

[195] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," Dept. Comput. Sci., Univ. Colorado Boulder, Boulder, CO, USA, Tech. Rep. CU-CS-321-86, 1986.

[196] R. Fletcher, *Practical Methods of Optimization*. Hoboken, NJ, USA: Wiley, 2013.

[197] A. Adhikari, A. Ram, R. Tang, and J. Lin, "DocBERT: BERT for document classification," 2019, *arXiv:1904.08398*. [Online]. Available: http://arxiv.org/abs/1904.08398

[198] J. Worsham and J. Kalita, "Genre identification and the compositional effect of genre in literature," in *Proc. COLING*, 2018, pp. 1963–1973.

[199] J. Wei, Q. Zhou, and Y. Cai, "Poet-based poetry generation: Controlling personal style with recurrent neural networks," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 156–160.

[200] J. Hopkins and D. Kiela, "Automatically generating rhythmic verse with neural networks," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2017, pp. 168–178.

[201] 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.

[202] B. Bena and J. Kalita, "Introducing aspects of creativity in automatic poetry generation," in *Proc. Int. Conf. NLP*, 2019.

[203] S. Tucker and J. Kalita, "Genrating believable poetry in multiple languages using GPT-2," Univ. Colorado, Colorado Springs, CO, USA, Tech. Rep., 2019.

[204] Z. Yu, J. Tan, and X. Wan, "A neural approach to pun generation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2018, pp. 1650–1660.

[205] H. Ren and Q. Yang, "Neural joke generation," Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, Final Project Rep. Course CS224n, 2017.

[206] B. Chippada and S. Saha, "Knowledge amalgam: Generating jokes and quotes together," 2018, *arXiv:1806.04387*. [Online]. Available: http://arxiv.org/abs/1806.04387

[207] P. Jain, P. Agrawal, A. Mishra, M. Sukhwani, A. Laha, and K. Sankaranarayanan, "Story generation from sequence of independent short descriptions," 2017, *arXiv:1707.05501*. [Online]. Available: http://arxiv.org/abs/1707.05501

[208] N. Peng, M. Ghazvininejad, J. May, and K. Knight, "Towards controllable story generation," in *Proc. 1st Workshop Storytelling*, 2018, pp. 43–49.

[209] L. J. Martin *et al.*, "Event representations for automated story generation with deep neural nets," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 868–875.

[210] E. Clark, Y. Ji, and N. A. Smith, "Neural text generation in stories using entity representations as context," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2018, pp. 2250–2260.

[211] J. Xu, X. Ren, Y. Zhang, Q. Zeng, X. Cai, and X. Sun, "A skeleton-based model for promoting coherence among sentences in narrative story generation," 2018, *arXiv:1808.06945*. [Online]. Available: http://arxiv.org/abs/1808.06945

[212] M. Drissi, O. Watkins, and J. Kalita, "Hierarchical text generation using an outline," in *Proc. Int. Conf. NLP*, 2018, pp. 180–187.

[213] Q. Huang, Z. Gan, A. Celikyilmaz, D. Wu, J. Wang, and X. He, "Hierarchically structured reinforcement learning for topically coherent visual story generation," 2018, *arXiv:1805.08191*. [Online]. Available: http://arxiv.org/abs/1805.08191

[214] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," 2018, *arXiv:1805.04833*. [Online]. Available: http://arxiv.org/abs/1805.04833

[215] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," 2015, *arXiv:1506.01057*. [Online]. Available: http://arxiv.org/abs/1506.01057

[216] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3155–3165.

[217] P. Tambwekar, M. Dhuliawala, L. J. Martin, A. Mehta, B. Harrison, and M. O. Riedl, "Controllable neural story plot generation via reinforcement learning," 2018, *arXiv:1809.10736*. [Online]. Available: http://arxiv.org/abs/1809.10736

[218] Y. Zhang *et al.*, "Adversarial feature matching for text generation," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 4006–4015.

[219] L. Chen *et al.*, "Adversarial text generation via feature-mover's distance," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4666–4677.

[220] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5141–5148.

[221] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114

[222] C. Doersch, "Tutorial on variational autoencoders," 2016, *arXiv:1606.05908*. [Online]. Available: http://arxiv.org/abs/1606.05908

[223] I. V. Serban *et al.*, "A hierarchical latent variable encoder-decoder model for generating dialogues," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3295–3301.

[224] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1587–1596.

[225] W. Wang *et al.*, "Topic-guided variational autoencoders for text generation," 2019, *arXiv:1903.07137*. [Online]. Available: http://arxiv.org/abs/1903.07137

[226] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2019, *arXiv:1904.09751*. [Online]. Available: http://arxiv.org/abs/1904.09751

[227] E. Clark, A. Celikyilmaz, and N. A. Smith, "Sentence Mover's similarity: Automatic evaluation for multi-sentence texts," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2748–2760.

[228] T. B. Hashimoto, H. Zhang, and P. Liang, "Unifying human and statistical evaluation for natural language generation," 2019, *arXiv:1904.02792*. [Online]. Available: http://arxiv.org/abs/1904.02792

[229] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *J. Artif. Intell. Res.*, vol. 61, pp. 65–170, Jan. 2018.

[230] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–36, Feb. 2019.

[231] X. Liu, Q. Xu, and N. Wang, "A survey on deep neural network-based image captioning," *Vis. Comput.*, vol. 35, no. 3, pp. 445–470, Mar. 2019.

[232] J. Krantz and J. Kalita, "Abstractive summarization using attentive neural techniques," in *Proc. Int. Conf. NLP*, 2018, pp. 1–9.

[233] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 2015, *arXiv:1511.06732*. [Online]. Available: http://arxiv.org/abs/1511.06732

[234] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," 2017, *arXiv:1705.03122*. [Online]. Available: http://arxiv.org/abs/1705.03122

[235] H. Zhang, J. Xu, and J. Wang, "Pretraining-based natural language generation for text summarization," 2019, *arXiv:1902.09243*. [Online]. Available: http://arxiv.org/abs/1902.09243

[236] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over freebase with multi-column convolutional neural networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2015, pp. 260–269.

[237] A. Santoro *et al.*, "A simple neural network module for relational reasoning," in *Proc. NIPS*, vol. 2017, pp. 4974–4983.

[238] D. Raposo, A. Santoro, D. Barrett, R. Pascanu, T. Lillicrap, and P. Battaglia, "Discovering objects and their relations from entangled scene representations," 2017, *arXiv:1702.05068*. [Online]. Available: http://arxiv.org/abs/1702.05068

[239] W. Yang *et al.*, "End-to-End open-domain question answering with BERTserini," 2019, *arXiv:1902.01718*. [Online]. Available: http://arxiv.org/abs/1902.01718

[240] H. Schwenk, "Continuous space translation models for phrase-based statistical machine translation," in *Proc. COLING*, 2012, pp. 1071–1080.

[241] T. Deselaers, S. Hasan, O. Bender, and H. Ney, "A deep learning approach to machine transliteration," in *Proc. 4th Workshop Stat. Mach. Transl. (StatMT)*, 2009, pp. 233–241.

[242] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proc. EMNLP*, 2013, pp. 1700–1709.

[243] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. ICML*, 2008, pp. 160–167.

[244] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: http://arxiv.org/abs/1406.1078

[245] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[246] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*. [Online]. Available: http://arxiv.org/abs/1609.08144

[247] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," 2017, *arXiv:1703.03906*. [Online]. Available: http://arxiv.org/abs/1703.03906

[248] R. Sennrich *et al.*, "Nematus: A toolkit for neural machine translation," 2017, *arXiv:1703.04357*. [Online]. Available: http://arxiv.org/abs/1703.04357

[249] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "OpenNMT: Open-source toolkit for neural machine translation," 2017, *arXiv:1701.02810*. [Online]. Available: http://arxiv.org/abs/1701.02810

[250] R. Sennrich and B. Haddow, "Linguistic input features improve neural machine translation," 2016, *arXiv:1606.02892*. [Online]. Available: http://arxiv.org/abs/1606.02892

[251] K. Ahmed, N. Shirish Keskar, and R. Socher, "Weighted transformer network for machine translation," 2017, *arXiv:1711.02132*. [Online]. Available: http://arxiv.org/abs/1711.02132

[252] S. Hochreiter, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. Piscataway, NJ, USA: IEEE Press, 2001.

[253] M. Cettolo, J. Niehues, S. Stüker, L. Bentivogli, and M. Federico, "Report on the 11th IWSLT evaluation campaign, IWSLT 2014," in *Proc. Int. Workshop Spoken Lang. Transl.*, Hanoi, Vietnam, 2014, pp. 2–17.

[254] J. Richard Medina and J. Kalita, "Parallel attention mechanisms in neural machine translation," 2018, *arXiv:1810.12427*. [Online]. Available: http://arxiv.org/abs/1810.12427

[255] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. ACL*, 2002, pp. 311–318.

[256] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," 2019, *arXiv:1904.09324*. [Online]. Available: http://arxiv.org/abs/1904.09324

[257] G. Lample and A. Conneau, "Cross-lingual language model pretraining," 2019, *arXiv:1901.07291*. [Online]. Available: http://arxiv.org/abs/1901.07291

[258] M. Xu Chen *et al.*, "The best of both worlds: Combining recent advances in neural machine translation," 2018, *arXiv:1804.09849*. [Online]. Available: http://arxiv.org/abs/1804.09849

[259] M. Denkowski and G. Neubig, "Stronger baselines for trustable results in neural machine translation," 2017, *arXiv:1706.09733*. [Online]. Available: http://arxiv.org/abs/1706.09733

[260] P. Koehn and R. Knowles, "Six challenges for neural machine translation," 2017, *arXiv:1706.03872*. [Online]. Available: http://arxiv.org/abs/1706.03872

[261] S. Kuang, D. Xiong, W. Luo, and G. Zhou, "Modeling coherence for neural machine translation with dynamic and topic caches," in *Proc. COLING*, 2018, pp. 596–606.

[262] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," 2014, *arXiv:1410.8206*. [Online]. Available: http://arxiv.org/abs/1410.8206

[263] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2015, *arXiv:1508.07909*. [Online]. Available: http://arxiv.org/abs/1508.07909

[264] M. Mager, E. Mager, A. Medina-Urrea, I. Meza, and K. Kann, "Lost in translation: Analysis of information loss during machine translation between polysynthetic and fusional languages," 2018, *arXiv:1807.00286*. [Online]. Available: http://arxiv.org/abs/1807.00286

[265] M. Ott, M. Auli, D. Grangier, and M. Ranzato, "Analyzing uncertainty in neural machine translation," 2018, *arXiv:1803.00047*. [Online]. Available: http://arxiv.org/abs/1803.00047

[266] W. Wang, T. Watanabe, M. Hughes, T. Nakagawa, and C. Chelba, "Denoising neural machine translation training with trusted data and online data selection," 2018, *arXiv:1809.00068*. [Online]. Available: http://arxiv.org/abs/1809.00068

[267] M. Johnson *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," 2016, *arXiv:1611.04558*. [Online]. Available: http://arxiv.org/abs/1611.04558

[268] D. Jurafsky and J. Martin, *Speech & Language Processing*, 3rd ed. London, U.K.: Pearson Education, 2017.

[269] L. Zheng, H. Wang, and S. Gao, "Sentimental feature selection for sentiment analysis of Chinese online reviews," *Int. J. Mach. Learn. Cybern.*, vol. 9, no. 1, pp. 75–84, Jan. 2018.

[270] M. Etter, E. Colleoni, L. Illia, K. Meggiorin, and A. D'Eugenio, "Measuring organizational legitimacy in social media: Assessing citizens' judgments with sentiment analysis," *Bus. Soc.*, vol. 57, no. 1, pp. 60–97, Jan. 2018.

[271] M. Cliche, "BB_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs," 2017, *arXiv:1704.06125*. [Online]. Available: http://arxiv.org/abs/1704.06125

[272] D. G. Bobrow, "Natural language input for a computer problem solving system," Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. MAC-TR-1, 1964.

[273] J. Weizenbaum, "ELIZA—A computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966.

[274] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," MIT, Cambridge, MA, USA, Tech. Rep. MAC-TR-84, 1971.

[275] 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," 2018, *arXiv:1804.07461*. [Online]. Available: http://arxiv.org/abs/1804.07461

[276] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," 2017, *arXiv:1705.06963*. [Online]. Available: http://arxiv.org/abs/1705.06963

[277] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam, The Netherlands: Elsevier, 2017.

[278] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Commun. ACM*, vol. 57, no. 6, pp. 13–15, Jun. 2014.