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

DANIEL W. OTTER, University of Colorado Colorado Springs, USA
JULIAN R. MEDINA, University of Colorado Colorado Springs, USA
JUGAL K. KALITA, University of Colorado Colorado Springs, USA

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 survey 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 a number of 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.

ACM Reference Format:
Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2018. A Survey of the Usages of Deep Learning in Natural Language Processing. 1, 1 (July 2018), 35 pages.

1 INTRODUCTION

The field of natural language processing (NLP) encompasses a wide variety of topics which involve the computational processing and understanding of human languages. Beginning in the 1930s [Hutchins 2001], the field historically has focused on the use of human-crafted rules, ad-hoc processing, and mathematical logic. Since the 1980s, it has increasingly relied on data-driven computation involving statistics, probability, and machine learning [Jones 1994; Liddy 2001]. The first artificial neural networks (ANNs) were conceived in the 1940s [McCulloch and Pitts 1943] as computational devices to capture human intelligence. Recent increases in computational power and parallelization, harnessed by Graphical Processing Units (GPUs) [Coates et al. 2013; Raina et al. 2009], now allow for "deep learning", which utilizes ANNs, sometimes with billions of trainable parameters [Goodfellow et al. 2016]. Additionally, the contemporary availability of large datasets, facilitated by sophisticated data collection processes, enables the training of such deep architectures via their associated learning algorithms [Ciresan et al. 2011; LeCun et al. 2015; Schmidhuber 2015].

In recent years, researchers and practitioners in natural language processing have leveraged the power of modern artificial neural networks with many propitious results, beginning in large part with the pioneering work of Collobert et al. [2011]. Over the last half decade, the use of artificial neural networks and deep learning has upsurged considerably in the field [Goldberg 2017; Liu and Zhang 2018]. 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 survey provides a brief introduction to both natural language processing and deep neural networks, and then presents an extensive discussion on how deep learning is being used to solve current problems in NLP. This

Authors' addresses: Daniel W. Otter, University of Colorado Colorado Springs, College of Engineering and Applied Sciences, Department of Computer Science, 1420 Austin Bluffs Pkwy, Colorado Springs, Colorado, 80918, USA, dotter@uccs.edu; Julian R. Medina, University of Colorado Colorado Springs, College of Engineering and Applied Sciences, Department of Computer Science, 1420 Austin Bluffs Pkwy, Colorado Springs, Colorado, 80918, USA, jmedina5@uccs.edu; Jugal K. Kalita, University of Colorado Colorado Springs, College of Engineering and Applied Sciences, Department of Computer Science, 1420 Austin Bluffs Pkwy, Colorado Springs, Colorado, 80918, USA, jkalita@uccs.edu.

discussion will be found useful by readers who want to familiarize themselves quickly with the current state of the art before embarking upon further advanced research and practice themselves.

The topics of natural language processing and artificial neural networks, including deep learning, are introduced in Section 2. The ways in which deep learning has been used to solve problems in core areas of NLP are presented in Section 3. The section is broken down into several subsections, namely natural language modeling (3.1), morphology (3.2), parsing (3.3), and semantics (3.4). Applications of deep learning to more practical areas in natural language processing are then discussed in Section 4. Specifically discussed are information extraction (4.1), text classification (4.2), summarization (4.3), question answering (4.4), machine translation (4.5), and image and video captioning (4.6). Conclusions are then drawn in Section 5 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.

2 OVERVIEW OF NATURAL LANGUAGE PROCESSING AND DEEP LEARNING

Recent advances in artificial neural technology, especially increasingly deep learning, have impacted heavily on the field of machine learning, often defining the state of the art in the solutions to a variety of complex problems in numerous disparate domains. Natural language processing is no exception. In many areas of NLP, the use of deep learning has produced results that have easily surpassed those obtained by other machine learning and statistical methods used for many years prior. In this section, introduction is provided to some of the most significant issues in NLP that draw attention of researchers and practitioners, followed by a brisk explanation of the various artificial neural architectures that have been used in NLP.

2.1 Natural Language Processing

The field of natural language processing, also known as computational linguistics, is an area of inquiry that has been influenced by a number of other fields such as linguistics, psychology, philosophy, cognitive science, probability and statistics, and machine learning. It involves the engineering of computational models and processes to solve practical problems in understanding human languages. These solutions are then used to build useful software that can be embedded in various settings. Work in NLP can be divided into two broad sub-areas: core areas and applications, although it is sometimes difficult to distinguish clearly to which areas some issues belong. The core areas address fundamental problems such as language modeling, which underscores quantifying associations among naturally occurring words; morphological processing, or dealing with segmentation of meaningful components of words and identifying the true parts of speech of words as used; syntactic processing or parsing, which builds sentence diagrams as possible precursors to semantic processing; and semantic processing itself, which attempts to distill meaning of words, phrases, and higher level components in a piece of text. The application areas involve topics such as extraction of useful information (e.g. named entities and relations) from documents, translation of text between and among languages, summarization of written works, automatic answering of questions by inferring the most probable answers, classification and clustering of documents in corpora, and image and video captioning. Often one needs to handle one or more of the core issues successfully and apply those ideas and procedures in order to solve practical problems.

Early attempts at NLP were usually rule based, where rules were hand crafted using knowledge derived from various areas. Sometimes the rules were ad-hoc in that they were made up to solve specific problems expediently. A number of formalisms were developed to describe the syntax of natural languages, with a view toward the facilitation of parsing and semantic processing. Syntactic rules were usually paired with logic-based semantic rules to obtain semantic representations of sentences that were then used for solving practical problems. Early approaches to NLP were described in depth in widely used textbooks of the time [Allen 1995; Winograd 1983].

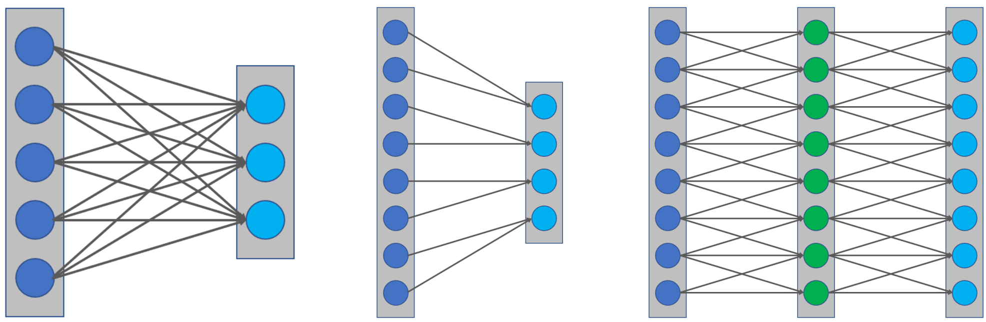
Rule-based processing is brittle, because when humans write or speak, often they do not obey all the niceties prescribed for "correct" language usage. In addition, with the proliferation of social media, texting on phones, and the democratization of writing on the Web, it has become necessary to process and understand informal language that deliberately flouts well-accepted rules of spelling and grammar. To satisfy the need to handle language beyond what is possible with rules written a priori, NLP began to transform slowly, starting in the 1980s, into a data-driven field predominantly using statistical and probabilistic computations along with machine learning algorithms. For example, parsing, which used to be rule based, became driven by statistical computation and machine learning, helped by corpora containing large numbers of parse trees, such as the Penn Treebank [Marcus et al. 1993]. Additionally, large corpora of carefully collected texts of various kinds have become available, ranging from large multi-lingual corpora of well-written formal and legal documents, to collections of pages obtained from Wikipedia, all the way to sets of highly informal texts such as tweets and other social media posts. Just like the rule-based approaches that preceded them, data-driven approaches to all aspects of NLP were discussed in several well-known textbooks [Charniak 1996; Jurafsky and Martin 2000; Manning et al. 1999].

Over time, a number of machine learning approaches such as naïve Bayes, k-nearest neighbors, hidden Markov models, conditional random fields, decision trees, random forests, and support vector machines were widely used in NLP. However, during the past several years, there has been a wholesale transformation in the field of NLP, and many of these approaches have been entirely replaced, or at least enhanced, by neural models, which are discussed in the following section.

2.2 Neural Networks and Deep Learning

An artificial neural network is a computing construct originally designed to mimic the architecture of the human brain [Fausett 1994; vanGerven and Bohte 2017]. While many modern ANNs employ a number of features that do not resemble the structure of the brain, they all use the same fundamental principles. Neural networks are composed of a number of interconnected nodes, or neurons, each receiving some number of inputs and supplying an output. The simplest neural networks are composed of two layers: input layers and neuron-consisting output layers. (Sometimes the input layers are not counted and these are considered single-layer networks.) Each node in the output layers performs a weighted sum computation on the values it receives from the input nodes and then generates an output using a simple nonlinear transformation function on the summation. The weights are associated with the edges between the nodes and are learned when the networks are shown large numbers of input and output pairs. Corrections to the weights are made in response to individual errors or losses the networks exhibit at the output nodes. While a number of learning methods such as the perceptron model [Rosenblatt 1958] and the Hebb Rule [Hebb 1949] have been used in the past, such corrections are usually made in modern networks using stochastic gradient descent, considering the derivatives of errors at the nodes.

Most networks incorporate additional layers of nodes between the input and output layers. These layers are known as hidden layers. When every node in a layer receives input from every node in the previous layer, that layer is said to be dense or fully connected. The main factors that distinguish different types of networks from each other are how the nodes are connected, and of late, 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 a feedforward neural networks (FFNNs). Conversely, networks in which one or more nodes receives input from itself or from a node that received input from it (either directly or indirectly) are referred to as feedback neural networks (FBNNs). While there is no clear consensus on exactly what defines a deep neural network (DNN), generally networks with multiple hidden layers are considered deep and those with many layers are considered very deep [Schmidhuber 2015]. Conversely, those with only a



(a) Fully-Connected ANN (b) Non-dense ANN (c) ANN with Hidden Layer

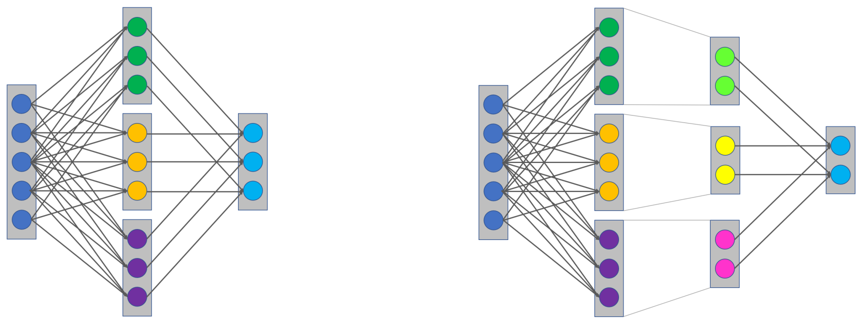
Fig. 1. Simple Neural Networks. Networks built until recently had very few layers. Early networks were fully connected, although non-dense networks are common these days.

single hidden layer (or no hidden layers) are said to be shallow. Feedforward networks with at least one hidden layer are often referred to as multilayer perceptrons (MLPs) [Rumelhart et al. 1985], although some definitions are more restrictive. Fig. 1 shows a few general ANN architectures.

2.2.1 Convolutional Neural Networks. Some networks contain layers in which the nodes receive input from only some of the nodes in previous layers, and therefore are not fully connected. One such type of network is the convolutional neural network (CNN) [LeCun et al. 1989, 1998], built upon Fukashima's [1980; 1982] neocognitron. Deriving their name from the convolution operation in mathematics and signal processing, convolutional neural networks use an assortment of different fully-enumerated functions, known as filters, to augment the data in varying ways, allowing for simultaneous analysis of different features in the data [Krizhevsky 2014; LeCun et al. 1995]. The structure is inspired by the biological neural networks (BNNs) found in the visual cortex [Hubel and Wiesel 1962]. A single neuron receives signals from a number of co-located rods and cones (visual receptors in the eye) in an area known as that neuron's receptive field. The neuron, in essence, performs a function on the received signals to identify if a particular low-level feature is present in its receptive field. Likewise, a number of other neurons with identical or similar receptive fields perform other operations on the stimuli to identify other features. Further neurons receive signals from these neurons as well as neurons with different receptive fields, and use this information to identify higher level features. Consequently, CNNs are used extensively in image and video processing. A simple CNN architecture is show in Fig. 2a. Due to some of the advantages they provide, they have also found use in a number of other fields, including speech and natural language processing [Dos Santos and Gatti 2014; Kalchbrenner et al. 2014; Kim 2014; Zeng et al. 2014].

Two of the largest advantages of convolutional networks are their ability to share weights and their ability to account for poorly processed data [LeCun et al. 1995]. Since many nodes in a CNN are looking for the same features as other neurons in their layer, just in different receptive fields, it makes sense that they would use the same functions (i.e., the same weights for their inputs). Therefore, for example, instead of having to train one million neurons to identify one hundred different features in ten thousand different regions, one only has to train one hundred different neurons, which can then be used ten thousand times each. This reduces the scope of the problem immensely (in this example by a factor of ten thousand), and with it the training time.

Another appealing feature of CNNs is their ability to account for data that may not be uniformly or systematically formatted. Since CNNs are capable of learning features that may be present in different regions of the input data, rather than relying on certain features being present in



(a) Basic CNN (b) CNN with Pooling

Fig. 2. Convolutional Neural Networks. These networks have three distinct filters that operate on the input layer, each with a receptive field size of three. In (b), k-max pooling is exhibited, where $k = 2$ with a pool size of three.

particular locations, less preprocessing is required. This allows CNNs to identify objects in images no matter where in the image the object occurs or what proportion of the frame is occupied by the object. Likewise, it allows linguistic features to be identified no matter where in a sample text they occur. This allows for processing of diverse forms of word morphology, sentence syntax, and other characteristics that may not always follow the same structure [Kalchbrenner et al. 2014].

Often, it is not important precisely where certain features occur, but rather whether or not they appear in particular localities. Therefore, pooling operations, such as that show in Figure 2b, can be used to minimize the size of feature maps (the outputs of the convolutional filters). The most common pooling operation is k-max pooling, in which the k largest values within the pool are retained and all others discarded. The sizes of such pools are generally small in order to prevent the loss of too much precision. As pooling can significantly reduce the number of locations to be examined by the following layers, its use can dramatically decrease the number of parameters in a network, leading to substantial improvements in training times. Convolutional neural networks usually consist of multiple convolutional and pooling layers followed by several fully-connected layers.

2.2.2 Recursive Neural Networks. Another notable type of artificial neural network used in natural language processing is the recursive neural network (RvNN) [Goller and Kuchler 1996; Kawato et al. 1987]. Much like convolutional networks, recursive networks 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 in NLP, 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 of the tree [Socher et al. 2011]. Since nodes in RvNNs are dependent on their previous results, and therefore feed back to themselves, RvNNs are not considered to be feedforward.

2.2.3 Recurrent Neural Networks. A simple type of recursive neural network that is used heavily in NLP is the recurrent neural network (RNN) [Elman 1990; Fausett 1994]. An RNN, in its simplest form, has an edge on a node, usually in a hidden layer, that feeds back to itself, as seen in Figure 3a. To demonstrate easily the creation of a chain like structure, which allows the network to remember the previous input in ordered sequences of input, an RNN can be visualized in an unrolled manner

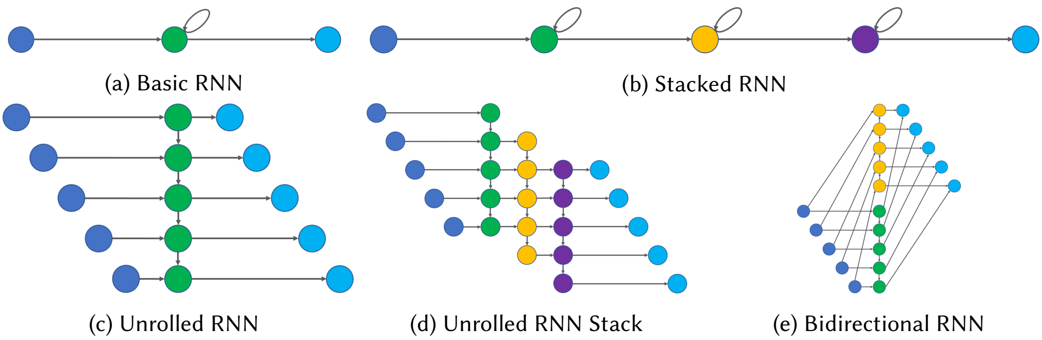


Fig. 3. Recurrent Neural Networks. These networks can be "unrolled" to appear as multiple different nodes. Note that (a) and (c) are identical, as are (b) and (d). If an entire data sequence is known prior to computation beginning, RNNs can also operate over the reverse sequence of the data. Bidirectional networks such as (e) utilize two RNNs working in opposite directions and then combine the outputs.

[Medsker and Jain 2001], such as shown in Figure 3c. Since much of NLP is dependent on the order of words or other elements such as phonemes or sentences, it is extremely useful to have memory of the previous elements when processing new ones [Mikolov et al. 2011a, 2010, 2011b].

Sometimes, backwards dependencies exist, i.e., correct processing of some words may depend on words that follow. These dependencies may be better captured if sentences are analyzed in the backwards direction. Thus, it may be beneficial to look at sentences in both directions, forwards and backwards, using two RNN cells, and combine their outputs. This arrangement of RNNs is called a bidirectional RNN and is shown in Figure 3e. Since an RNN's memory of data early in the chain gradually decays, later outputs may be erroneous if they rely heavily on early information. Examining chains in both directions counteracts this, as the RNNs focus separately on early data and late data. Note that all information must be present prior to computation in order to view it in reverse, preventing the use of bidirectional RNNs in some live applications such as speech processing programs.

It has also been noticed that sometimes it leads to a better final representation if instead of having one RNN cell, there is a sequence of RNN cells, one after the other. This may allow the effect of an input to linger longer than a single RNN cell, allowing for longer-term effects. This setup of sequential RNN cells is called an RNN stack [El Hihi and Bengio 1996; Schmidhuber 1992]. Of course, there can be more than two stacked RNN cells as well, as shown in Figures 3b and 3d.

2.2.4 Long Short-Term Memory Networks. The internal workings of nodes in RvNNs (RNNs included) may vary. Single neurons may be used or complex networks may be used. One highly engineered RNN is the long short-term memory (LSTM) network [Greff et al. 2017; Hochreiter and Schmidhuber 1997]. In LSTMs, the recursive nodes are composed of several individual neurons (or in some variants small ANNs) connected in a manner designed to retain 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. Furthermore, each of the elements of the results are remembered equally. Oftentimes, it is important to remember fully information from the distant past, while at the same time, other very recent information may be of zero importance. By using LSTM blocks in RNNs to form LSTM networks, this important information can be retained almost indefinitely while irrelevant information can be forgotten. However, if information is disregarded, it cannot be recovered, even if its presence is desired later. Current research looks to correct this problem, possibly by retaining larger amounts of data and

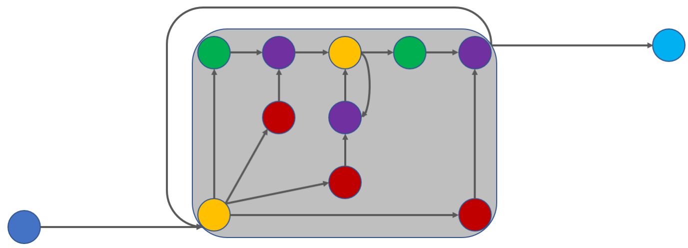


Fig. 4. Long Short-Term Memory Network. In this network, the input is concatenated with the previous output and is then distributed to an activation node and to three sigmoid nodes followed by gate nodes. Depending on the outputs of the sigmoid nodes, the gate nodes choose whether or not to allow information to pass through. These gates control i) the input, ii) the recurrent internal state, and iii) the output.

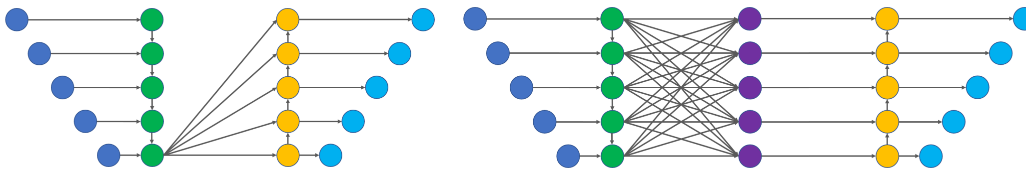
simply altering which parts of it particular examples are attentive to [Xu et al. 2015; Yang et al. 2016]. In another line of work, a slightly simpler variant of the LSTM, called the Gated Recurrent Unit (GRU), has been shown to perform as well as or better than standard LSTMs in many NLP tasks [Cho et al. 2014; Chung et al. 2014].

2.2.5 Attention Mechanisms, Residual Connections, and Dropout. A number of other forms of artificial neural networks exist [Schmidhuber 2015], however, those listed above are the most commonly used in current natural language processing research. Furthermore, many neural networks, including some mentioned in the discussions that follow, contain features from multiple types of networks. In order to understand some of these networks, one must understand some additional components and methods used in neural networks. One such component is the attention mechanism.

For many tasks such as machine translation, text summarization, or image or video captioning, the output is in textual form. (This differs from many tasks in which text is input and a classification decision must be made.) 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 text based on this vector. Since the vector must be of this specified length in order to be fed into the decoder, the encoder must also generate a vector of this length. While RNNs produce an output at each time step, the number of time steps is variable. However, ideally speaking, the final output will account for all inputs. This output will be a fixed size, and therefore can be passed into the decoder. The problem with this scheme, which is shown in Figure 5a, is that the RNN is forced to encode an entire sequence to a finite length vector, without regards to whether or not any of the inputs are more important than others.

A robust solution to this is that of *attention*. The first noted use of an attention mechanism was that of Bahdanau et al. [2014]. This approach 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 Figure 5b. Many variants of the mechanism have been introduced, popular ones including convolutional [Rush et al. 2015], intra-temporal [Paulus et al. 2017], gated [Wang et al. 2017], and self-attention [Vaswani et al. 2017].

Deep neural networks are usually trained via backpropagation [Rumelhart et al. 1985]. This method computes the error at the output nodes, and makes updates to weights on all edges connecting to them from lower level nodes. The backward propagation of error and correction of edge weights happens layer by layer in a similar manner, all the way back to the input layer.



(a) Basic RNN Encoder–Decoder Pair

(b) Encode–Decoder Pair with Attention Mechanism

Fig. 5. 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.

In increasingly deep networks, the gradients used to correct for error often vanish or explode by the time early layers are reached. [Bengio et al. 1994]. This can be somewhat mitigated by choosing activation functions, such as the Rectified Linear Unit (ReLU) [Nair and Hinton 2010], which do not exhibit regions that are arêtically steep or have bosonically small gradients. Also in response to this issue, as well as others [He et al. 2016], 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 [Srivastava et al. 2015] and DenseNets [Huang et al. 2017].

Another important method used in training ANNs is *dropout*. In dropout, some 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. For simplicity in training, this is usually implemented via dropout layers, which are simply single feedforward layers (generally fully connected) in which all the dropped nodes are contained. Dropout rates between 0 and 1 designate the proportion of nodes dropped during each training batch.

3 DEEP LEARNING IN CORE AREAS OF NATURAL LANGUAGE PROCESSING

The core issues of NLP are those issues that are inherently present in any computational linguistic system. In order 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 four main areas: language modeling, morphology, parsing and semantics.

Language modeling can be viewed in two ways. First, it is the task of determining which words follow which. By extension, however, this can be viewed as determining what words mean, as individual words are only weakly meaningful if not meaningless, 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 how they evolve through the use of prefixes and suffixes, compounds, and other intraword devices, in order to display tense, gender, plurality, and a number of other linguistic constructs. Parsing considers the interactions between words. Specifically, it considers which words modify each other, and in which ways, forming constituents (i.e., phrases of various kinds), leading to a sentential structure. Encapsulating all of these is the area of semantics, which is the study of what words mean as a collective. It must take into account the meanings of the individual words and how they relate to and modify others, as well as the context these words appear in and some degree of world knowledge, i.e., "common sense".

One can easily see that there is a significant amount of overlap between each of the areas discussed. Therefore, many of the 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.

3.1 Language Modeling

Arguably, the most important task in current natural language processing is that of language modeling. Language modeling (LM) is an essential piece of almost any application of NLP, from speech recognition to machine translation. In essence, language modeling is the process of creating a model to predict words or simple linguistic components given previous words or components, usually with associated probabilities [Jurafsky and Martin 2000]. This is helpful in speech recognition as it can help resolve ambiguity between two similar sounding words or phrases. Likewise, it is useful for applications in which a user types input, particularly touch screen keyboards, 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 diverse tasks such as machine translation or text summarization, in which the program must validate input and/or produce output in the form of natural language. Using prediction, such programs are able to generate more clear and human-sounding sentences and validate the opposite.

3.1.1 Early Language Modeling. Historically, language modeling has been a statistical task [Charniak 1996; Jurafsky and Martin 2000; Manning et al. 1999]. For example, probabilities of word occurrences were calculated using data from large corpora. Initially, the bag-of-words approach was used, in which words were simply predicted by their frequencies of occurrence in the training corpus, possibly with smoothing of counts. A later development, that of the n-gram model, attempted to solve the problem by determining frequencies of words following each other. Inevitably, this significantly increased the size of the model and computation required to construct it, as instead of simply having vectors of size v, where v was the size of the vocabulary, the models consisted of matrices of size $v \times v$. However, as many words rarely, if ever, follow each other in natural usage, the matrices were sparse. This observation allowed for significant reduction in the sizes of the models.

The n-gram approach does not simply look at the probabilities of words following single other words, but instead at the probabilities of words following a set of $n-1$ prior words. Hence, a bag-of-words model would be considered a unigram or 1-gram model, a model using only one previous word a 2-gram or bigram model, and a model using the two previous words a trigram model. Due to the decreasing likelihood of a set of words having co-occurred in the training corpus as the size of n increases, a number of techniques were developed to allow for reasonably long n-grams to be used. [Jurafsky and Martin 2000] Typical models would use unigrams through 5-grams.

3.1.2 Neural Language Modeling. A major problem with statistical language models was their inability to deal well with synonyms or out-of-vocabulary (OOV) words. While they could predict the re-occurrence of word sets that were present in the training corpus, they had difficulty predicting new sets of words; for example, a set of words from the training corpus in which one word was replaced with a synonym or a similar word with a different meaning (e.g. *car* and *truck* or *red* and *blue*). Progress was made in solving these problems with the introduction of the neural language model [Bengio et al. 2003]. While much of NLP took another decade to begin to utilize ANNs heavily, the LM community immediately took advantage of them, and continued to develop sophisticated models, many of which were summarized by DeMulder et al. [2015]. Not only do they allow for the prediction of synonymous words, they also allow for modeling the relationships between words [Mikolov et al. 2013a,b].

The word vectors with numeric components, obtained by language modeling techniques are called embeddings. 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, mathematics 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.

3.1.3 Evaluation of Language Models. While neural networks have certainly made breakthroughs in the LM field, it is often rather hard to quantify improvements. In some applications such as speech recognition or machine translation, metrics such as word error rate (WER) can be used. (Word error rate is the fraction of actual output words not correctly matching the desired output words.) However, typically 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 [Chen et al. 1998; Clarkson and Robinson 2001; Iyer et al. 1997]. 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 LMs trained on the same datasets, but when they are trained on different vocabularies, the metric becomes less meaningful. Luckily, there are several benchmark datasets that are used widely in the field, allowing for reasonable comparison. Two such datasets are the Penn Treebank (PTB) [Marcus et al. 1993], and more recently, in order to allow for comparison of large models that require extensive amounts of training data, the Billion Word Benchmark [Chelba et al. 2013].

3.1.4 Memory Networks and Attention Mechanisms in Language Modeling. Daniluk et al. [2017] tested several different networks using several variations of attention mechanisms. The first network trained had a simple attention mechanism, which was not fully connected, having a window length of five. The second and third networks were more novel, particularly within the context of language modeling. The researchers hypothesized that using a single value to predict the next token, to encode information for the attentional unit, and to decode the information in the attentional unit hinders a network, as these are distinct tasks, and it is difficult to train a single parameter to perform all three tasks simultaneously. Therefore, in the second network, they designed each node to have two outputs: one used 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. These networks were respectively dubbed "Key-Value Attention" and "Key-Value-Predict Attention" networks.

Tests on a self-made 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. This showed that the use of attention mechanisms, particularly those with several distinct parameters, is useful in language modeling. 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, it was decided to test a fourth network which 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 than have often been suggested.

Another recent study was done on the usage of residual memory networks (RMNs) for language modeling [Beneš et al. 2017]. While the authors made no mention as to whether or not they tested the use of multiple residual connections, they found that residual connections skipping two layers were most effective, followed closely by those skipping a single layer. Using these findings they designed recurrent networks with residual connections skipping two units at a time. Rather than having such a connection present at every layer, they had such repetitions reoccur in blocks. A residual connection was present between the first layer and the fourth, 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 in terms of performance, however, wide networks were found to be harder to train. The networks were trained on the Penn Treebank and compared to the results of RNNs and LSTMs. It was found that RMNs are capable of outperforming LSTMs of similar size.

3.1.5 Convolutional Neural Networks in Language Modeling. A convolutional neural network that was used recently in language modeling was one in which the pooling layers were replaced with fully-connected layers [Pham et al. 2016]. These layers allowed the feature maps to be reduced to lower dimensional spaces just as the pooling layers usually do. However, whereas any references to location of such features are lost in pooling layers, fully-connected layers somewhat retain this information. Several variations of CNNs were tested, all incorporating this characteristic. In total, three different architectures were implemented: a multilayer perceptron CNN (MLPConv) in which the filters were not simply linear, but instead small MLPs [Lin et al. 2013]; a multilayer CNN (ML-CNN) in which multiple convolutional layers are 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). All of the networks were tested on three different datasets: the Penn Treebank, Europarl-NC [Bojar et al. [n. d.]], and ukWaC [Baroni et al. 2009]. The results of this study showed that stacking convolutional layers was actually detrimental in language modeling, but that both MLPConv and COM were effective at reducing the perplexity on each of the datasets tested. Furthermore, combining MLPConv with the varying kernel sizes of COM provided even better results, achieving a perplexity of ninety-one on the Penn Treebank. Analysis was performed by the researchers, showing that the networks learned specific patterns of words, such as, "as . . . as". Lastly, 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 found to be of some significance as well. Words as far as sixteen positions away were considered.

3.1.6 Character Aware Neural Language Models. While most CNNs used in NLP receive words (or word embeddings) as input, some recent networks have analyzed character level input instead. One such network is that of Kim et al. [2016], which unlike previous networks that accepted character level input [Botha and Blunsom 2014], accepted only such input, rather than combining it with word embeddings. A CNN was used to process the character level input in order to provide representations of the words. In a similar manner as 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) [Srivastava et al. 2015] and an LSTM. They trained the network on the English Penn Treebank, as well as on datasets for Czech, German, Spanish, French, Russian, and Arabic. The Arabic data was from the News-Commentary corpus and all of the other data was from the 2013 ACL Workshop on Machine Translation (both the small and large datasets). For every non-English language except Russian, the network outperformed previously published results [Botha and Blunsom 2014] in both the large and small datasets. For Russian, better results were achieved on the small dataset, but not the large. On the Penn Treebank, results were produced on par with the existing state of the art [Zaremba et al. 2014], achieving a perplexity of 78.9 (compared to 78.4). However, the network had only 19 million trainable parameters, which is considerably lower than others, and was handicapped by the number of words marked as unknown in the Penn Treebank. Since the network focused on morphological similarities produced by character level analysis, it was more capable than previous models of handling words that were rarely seen in the corpus. Analysis showed that without the use of highway layers, many words had nearest neighbors that were orthographically similar (spelled similarly), but not necessarily semantically similar. The addition of highway layers (which were used in the highest performing network) seemed to solve this problem. Additionally, 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 an extremely robust model.

Jozefowicz et al. [2016] tested a number of architectures including some mentioned above, and a few others producing character level outputs [Chelba et al. 2013; Ji et al. 2015; Shazeer et al. 2015; Williams et al. 2015]. Whereas many of these models had only been tested on small scale language modeling, this study tested them on large scale language modeling, 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 [Chelba et al. 2013]), was a large LSTM using a character level CNN as an input network. The best performance, however, was achieved using an ensemble (group of parallel networks whose outputs are averaged) of ten LSTMs. The perplexity of this ensemble, 23.7, far surpassed the previous state-of-the-art ensemble [Shazeer et al. 2015], which had a perplexity of 41.0.

3.2 Morphology

Morphology is concerned with finding segments within single words, including roots and stems, prefixes, suffixes, and—in some languages—infixes. Affixes (prefixes, suffixes, or infixes) are used to overtly modify stems for gender, number, person, et cetera. Furthermore, different stems derived from the same roots usually convey separate but related concepts.

Luong et al. [2013] constructed a language model that was morphologically aware. An RvNN was used to model the morphological structure of English words. A neural language model was then placed on top of the RvNN. The model was trained on the WordSim-353 dataset [Finkelstein et al. 2001] and segmentation was performed using Morfessor [Creutz and Lagus 2007]. An additional dataset of rare words was generated, as most available datasets do a poor job of modeling these words. Two models were constructed—one using context and one not. It was found that the model that was insensitive to context over-accounted 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 datasets [Huang et al. 2012; Miller and Charles 1991; Rubenstein and Goodenough 1965], significantly outperforming previous embedding models on all.

A good morphological analyzer is often important in the context of larger linguistic systems and other NLP tasks. As such, one recent study by Belinkov et al. [2017] examined the extent to which morphology was learned and utilized by a variety of neural machine translation models. In this study, 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 [Cettolo 2016; Cettolo et al. 2012]. The decoders were then replaced with part-of-speech (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.

The effects of character-level models have also been further examined in machine translation [Chung et al. 2016], as well as in a number of other fields such as semantics [Rajana et al. 2017] and parsing [Yu and Vu 2017]. Their use has, however, been most prevalent in language modeling [Botha and Blunsom 2014; Kim et al. 2016], as previously mentioned in Section 3.1.6.

Morita et al. [2015] 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 both an automatically labeled [Kawahara and Kurohashi 2006] corpus and on a new manually labeled corpus which they presented. 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 [Kawahara et al. 2002] and the Kyoto University Web Document Leads Corpus [Hangyo et al. 2012], outperforming all baselines on all tasks.

3.3 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 [Jurafsky and Martin 2000]. In constituency parsing, phrasal constituents are extracted from a sentence in a hierarchical fashion. Phrases are identified, which in turn form larger phrases, eventually culminating in complete sentences. Dependency parsing on the other hand looks solely at the relationships between pairs of individual words.

Most recent uses of deep learning in natural language parsing have been in the area of 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 [Jurafsky and Martin 2000]. Without such an approach, an immense number of trees would be produced, many of them being totally illogical.

More popular in recent years than graph-based approaches have been transition-based approaches. In these approaches, only one parse tree is usually constructed. 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. These arcs can either be right-arcs or left-arcs, depending on whether the top word (which is further right in the sentence) is dependent on the bottom word (which is further left), or the bottom word is dependent on the top. 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 [Nivre 2003, 2004], all dependents are connected to a word before the word is connected to its parent. In the *arc-eager* approach [Nivre 2003, 2004], 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 [Nivre et al. 2009], the arc-standard approach is modified to allow swapping of positions on the stack. This makes the graphing of non-projective edges possible.

3.3.1 Early Neural Parsing. Compared to many areas of NLP, parsing has only recently been introduced to the power of deep learning. One early application of deep learning to natural language processing, that of Scoher et al. [2013a; 2013b], included the use of RNNs with probabilistic context-free grammars (PCFGs) [Chi and Geman 1998; Fujisaki et al. 1991; Jelinek et al. 1992], allowing for semantically different phrases to be parsed differently, even if their part-of-speech tags were identical. 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 [2014]. Such performance was achieved on the Penn Treebank 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. [2015a] 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 [Petrov and McDonald 2012] and the Question Treebank [Judge et al. 2006] as opposed to the Wall Street Journal portion of the Penn Treebank [Marcus et al. 1993]), showing that neural models can generalize between domains. Embeddings were first used in dependency parsing by Stenetorp [2013], laying the path for most recent attempts at natural language parsing. 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 dataset [Surdeanu et al. 2008]), by the time it reached the end of a sentence, it seemed to have difficulty remembering phrases from early in the sentence.

3.3.2 Transition-Based Dependency Parsing. The basis for most major recent research in parsing is the work of Chen and Manning [2014], who pushed the state of the art in both UAS and LAS on both English and Chinese datasets, scoring 92.2% UAS on the English Penn Treebank. They accomplished this by using a simple feedforward neural network 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 that were being heavily used. While they achieved state-of-the-art results, they noted the simplicity of their model as well as a number of areas in which it could be improved upon.

The first of such areas was that of the search mechanism employed by the model. Chen and Manning used a simple greedy search, which was replaced by Zhou et al. [2015] with a beam search. By doing this, they achieved a significant improvement, nearing the performance of the new state of the art. In addition to using beam search, Weiss et al. [2015] 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. Furthermore, they were able to train on significantly more examples than typical by utilizing tri-training [Li et al. 2014], a process in which potential data samples are fed to two other parsers, and those on which both of the parsers agree upon are used for training the primary parser. This approach became the state of the art with respective UAS and LAS scores of 94.26% and 92.41% on the Penn Treebank. Alberti et al. [2015] further extended this work by testing several similar models on the Wall Street Journal dataset, the Web English Treebank dataset, and the Question Treebank data, as well as datasets from CoNLL 2009 [Hajič et al. 2009] for a number of languages. State-of-the-art performance was achieved in both UAS and LAS on the three English datasets as well as for most of the languages tested from CoNLL 2009.

Another model was produced using an LSTM instead of a feedforward network [Dyer et al. 2015]. 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 UAS and LAS scores of 93.1% and 90.9%, respectively on the Stanford Dependency Treebank [De Marneffe and Manning 2008], as well as state-of-the-art results on the

CTB5 Chinese dataset [Xue et al. 2005]. Lastly, Andor et al. [2016] used a feedforward network with global normalization on a number of tasks including part-of-speech tagging, sentence compression, and dependency parsing. State-of-the-art results were obtained on all tasks, including reaching a UAS score of 94.61% on the Wall Street Journal dataset. Notably, their model required significantly less computation than comparable models to achieve these results.

At the time of writing of this survey, the state of the art is defined by another such transition-based dependency parser [Wang et al. 2018]. Much like the work of Stenentorp [2013], this model uses an alternative algorithm to produce directed acyclic graphs, rather than simple trees. In addition to the typical stack and buffer used in transition-based parsing, the algorithm employs a deque. This allows for the representation of multi-parented words, which although rare in English, are common in many natural languages. Furthermore, it allows for multiple children of the *ROOT* label. In addition to producing said graphs, this work is novel in its use of two new LSTM-based techniques: Bi-LSTM Subtraction and Incremental Tree-LSTM. Bi-LSTM Subtraction builds off of previous work [Cross and Huang 2016; Wang et al. 2017] 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 [Tai et al. 2015], modified for directed acyclic graphs instead of trees, by connecting children to parents incrementally, rather than connecting all children to a parent simultaneously. Both techniques were found to improve performance, and the model exercising both achieved the best published scores for fourteen of the sixteen evaluation metrics used on SemEval-2015 Task 18 (English) [Oepen et al. 2015] and SemEval-2016 Task 9 (Chinese) [Che et al. 2012].

3.3.3 Generative Dependency and Constituent Parsing. While transition-based dependency parsing still remains successful, currently defining the state of the art, improvements have recently been attained by applying deep learning to constituency parsing and generative models. Dyer et al. [2016] proposed a model that used recurrent neural network grammars for parsing and language modeling. Whereas 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. While not attaining state-of-the-art results in all of parsing, 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.

The first model to achieve state-of-the-art results using a deep learning approach to generative parsing was that of Choe and Charniak [2016]. Parsing was treated as a language modeling problem, and an LSTM was used to assign probabilities to the parse trees. Using this approach, UAS and LAS scores of 95.9% and 94.1%, respectively, were achieved on the Penn Treebank. Fried et al. [2017] considered both of these models and tested them to determine whether the power of these 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 still. 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, with an F1 score on the Penn Treebank of 93.4%. Finally, an ensemble of eight such models (using two parsers) was constructed and achieved the best results (94.25% F1 on Penn Treebank) prior to the model proposed by Wang et al. [2018].

3.4 Semantics

Semantic processing involves understanding the meaning of words, phrases, sentences, or documents at some level. The term *meaning* is difficult to define, and linguists and philosophers have been debating about it for centuries. Word embeddings, such as Word2Vec [Mikolov et al. 2013a,b] and GloVe [Pennington et al. 2014], claim to capture meanings of words, following the Distributional Hypothesis of Meaning [Harris 1954]. 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. Such composition is necessary for many tasks such as summarization, question answering and video captioning. In this section, neural semantic processing research is separated into two distinct areas: Work focusing on comparing the semantic similarity of two portions of text, and work focusing on capturing and transferring meaning in high level constituents of language, particularly sentences.

3.4.1 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. A series of workshops called SemEval have been conducted annually from 1998 to date to encourage research in approaches to semantic computing. Comparing the semantic similarity of two sentences has been a staple contest in SemEval contests, and papers published in other venues.

Hu et al. [2014] proposed two CNNs to perform a semantic comparison task. The first model, ARC-I, inspired by Bordes et al. [2014], 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. Three "multiple choice" experiments were performed, testing English sentence completion, responding to Chinese tweets, and comparing English sentence meaning. The approach outperformed a number of existing models.

Building on prior work [Hu et al. 2014; Kalchbrenner et al. 2014; Socher et al. 2011], Yin and Schütze [2015] 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. Following Hu et al., they used a "Siamese" network, however, it was modified by the use of the Dynamic CNNs proposed by Kalchbrenner et al. (described in Section 3.4.2). Additionally, the feature maps from each level were used in the comparison process, rather than simply the top-level feature maps. They achieved state-of-the-art results on the Microsoft Research Paraphrase Corpus (MSRP) [Das and Smith 2009; Dolan et al. 2004].

He et al. [2015] also used a CNN to compare the semantic similarity of sentences. The network resembled that of Yin and Schütze, using two CNNs in parallel. The constructed feature maps were then compared using a "similarity measurement layer" followed by a fully-connected layer and then a log-softmax output layer. The windows used in the convolutional layers ranged in length from one to four. The network was trained and evaluated on three different datasets: MSRP, the Sentences Involving Compositional Knowledge (SICK) dataset [Marelli et al. 2014], and the Microsoft Video Paraphrase Corpus (MSRVID) [Agirre et al. 2012]. State-of-the-art results were achieved on the first and the third, and nearly matched on the second. Ablation studies showed that a number of elements such as POS tags, multiple pooling operations, and multiple similarity metrics, as well as several other features, all contributed to the performance of the model.

The state-of-the-art network that generated superior results to the model proposed by He et al. on the SICK dataset was a model concocted using an RvNN with LSTM-like nodes [Tai et al. 2015], referred to by its designers as a Tree-LSTM. Two different variations were examined (a constituency-based model and a dependency-based model) and tested on both the SICK dataset and the Stanford Sentiment Treebank [Socher et al. 2013b]. The constituency-based model achieved state-of-the-art results on the Stanford Sentiment Treebank and the dependency-based model achieved state-of-the-art results on SICK.

He et al. presented another model [2016], which outperformed that of Tai et al. on SICK. The model formed a matrix of the two sentences being compared before applying a "similarity focus layer" and then passing the information into a nineteen-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. In addition to testing on the SICK dataset, they tested on MSRVID, SemEval 2014 Task 10 [Agirre et al. 2014], WikiQA [Yang et al. 2015], and TreeQA [Wang et al. 2007]. State-of-the-art results were attained on all of the above.

3.4.2 Sentence Modeling. While the comparison of the meanings of sentences and phrases is useful, it in large part relies on being able to calculate the meanings of individual sentences and phrases. This is a common theme in all of NLP. Extending from neural language modeling, which attempts to capture the meaning of words in vectors, 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 [2014], which attempt to model paragraphs or larger bodies of text in this way. While these are intriguing, there is still much work to be done on the phrase and sentence levels, which will be the focus of the remainder of this section.

Kalchbrenner et al. [2014] generated representations of sentences using a dynamic convolutional neural network (DCNN), which used a number of different filters and dynamic k-max pooling layers to determine which features detected by those filters are most important. Due to the 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 several applied tasks that require thorough semantic understanding. It outperformed all comparison models in predicting sentiment of movie reviews in the Stanford Sentiment Treebank [Socher et al. 2013a] and in identification of sentiment in tweets [Go et al. 2009]. It was also one of the top performers in classifying types of questions using the TREC database [Li and Roth 2002].

Undoubtedly, the area of NLP space most dependent on good semantic understanding is that of machine translation. Between their requirement for such understanding and their ease of examination due to the typical encoder–decoder structure they use, neural machine translation (NMT) systems are splendid testbeds for researching internal semantic representations. One study that employed this, the results of which were published during the writing of this survey, was that conducted by Poliak et al. [2018a]. Encoders were trained on four different language pairs: English and Arabic, English and Spanish, English and Chinese, and English and German. Building on a number of previous studies, the decoding classifiers were trained on four distinct datasets: Multi-NLI [Williams et al. 2017], which is an expanded version of SNLI [Nangia et al. 2017], as well as three recast datasets from the JHU Decompositional Semantics Initiative [White et al. [n. d.]] (FrameNet Plus or FN+ [Pavlick et al. 2015], Definite Pronoun Resolution or DPR [Rahman and Ng 2012], and Semantic Proto-Roles or SPR [Reisinger et al. 2015]). None of the results were particularly strong, although they were strongest in SPR. This led the researchers 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, plurality, etc.). They did, however, find that the models may learn a reasonable amount about proto-roles (e.g. who or what is the recipient of an action). A concurrent work [Poliak et al. 2018b] analyzed the quality of many datasets used for natural language inference.

Herzig and Berant [2017] 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 constructing and testing three different LSTM-based models. The first model was a one-to-one model, in which a single encoder and 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" dataset [Wang et al. 2015]. 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 in a recently published study by Brunner et al. [2018]. This study also has some of the most significant implications for the entire field of natural language processing since Mikolov's arithmetic with word embeddings [Mikolov et al. 2013b]. In this study, several LSTM-based encoder–decoder networks were created, and the embedding vectors produced were analyzed. A single encoder accepting English sentences as input was used, as were four different decoders. The first such decoder was a replicating decoder, which attempted to reproduce the original English input. The second and third decoders attempted to translate 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, in one case, all four. Sentences of fourteen different structures from the EuroParl dataset [Koehn 2005] were used to train the networks. A set of test sentences were then fed to the encoders and their output analyzed. In all cases, fourteen clusters were formed, each corresponding to one of the sentence structures. Analyzing the spacing between the clusters and the number of misclassified results showed that adding more decoders led to more correct and more definitive clusters. In particular, using all four of the decoders led to a zero error rate. Furthermore, the researchers tested and confirmed a hypothesis regarding these sentence embeddings. They found that just as logical arithmetic can be performed on word embeddings, so can it be performed on sentence embeddings. For example, taking the vector embedding for, "This example works," and subtracting the vector for, "This example fails," produced a vector which when added to the embedding for, "Another attempt fails," resulted in a vector nearly identical to the embedding of, "Another attempts work." Note that while this sentence is incorrect, it is very close to the desired result. Furthermore, it differs from the desired sentence only in terms of anaphora, which was noted above to be the most difficult semantic construct to process. While this is acknowledged to be a well-controlled environment, it shows that NLP models are capable of capturing a great deal of meaning and paves the way for a new line of research, and potentially another revolution in the field.

3.5 Summary of Core Issues

Deep learning approaches have generally performed very well in creating the foundation on which useful natural language applications can and are being built. This section has presented only a few selected efforts at using deep learning to tackle such problems. Topics not discussed in this paper due to lack of space include part-of-speech tagging, spelling correction, word sense disambiguation, co-reference resolution, discourse and conversational issues, and a host of others. A section is now furnished in which are summarized several application areas in which deep learning has contributed.

4 APPLICATIONS OF NATURAL LANGUAGE PROCESSING USING DEEP LEARNING

While the study of core areas of natural language processing is important to understanding how neural models work, it is meaningless in and of itself from an engineering perspective, which values applications that benefit humanity, not pure philosophical and scientific inquiry. The true usefulness of such study comes from the application of the gained understanding to real world problems. Current approaches to solving several immediately useful problems in natural language processing 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 requires expertise on several other topics such as acoustic processing, it is generally considered another field in and of itself, sharing many commonalities with the field of NLP. A number of studies in speech processing have been previously summarized [Graves et al. 2013; Hinton et al. 2012].

4.1 Information Extraction

Information extraction is the process of using algorithms to extract explicit or implicit information from text. The outputs of systems using these algorithms vary across implementations, but often the extracted data and the relationships within it are saved in relational databases [Cowie and Lehnert 1996]. Early methods included the use of simplistic information classification, pattern matching, and grammar methods to create rule-based approaches [Andersen et al. 1992; Salton and Harman 2003]. Current information retrieval systems use machine learning algorithms of various kinds—supervised and unsupervised. Commonly extracted information includes named entities and relations, events and their participants, temporal information, and tuples of facts.

4.1.1 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. Until recently, support vector machines [Vapnik 2013] and conditional random fields [Lafferty et al. 2001] were the most commonly used approaches for NER. The multi-task approach of Collobert et al. [2011] 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 words. Furthermore, it used word embeddings as input, rendering it unable to leverage explicit character level features.

Long short-term memory was first used for named entity recognition by Hammerton [2003]. The model, which was ahead of its time, had a small network due to the lack of available computing power at the time. Additionally, sophisticated numeric vector models for words were not yet available. Results obtained were slightly better than the baseline for English and much better than the baseline for German. Dos Santos et al. [2015] used a deep neural network architecture, known as CharWNN, which jointly used word-level and character-level inputs to perform sequential classification. Before being tested for NER, this exact network, with the same hyper-parameters and without any hand-crafted features, was used effectively for language-independent POS tagging [Dos Santos and Gatti 2014]. Dos Santos et al. [2015] performed a number of experiments using the HAREM I annotated Portuguese corpus [Santos et al. 2006], and the SPA CoNLL2002 annotated Spanish corpus [Carreras et al. 2002]. For the Portuguese corpus, CharWNN outperformed the previous state-of-the-art system by 7.9 points in the F_1 score across ten named entity classes. It also achieved state-of-the-art performance on the Spanish corpus. The authors noted that when used alone, neither word embeddings nor character level embeddings worked as well as when used jointly. This revalidated a fact long-known in the NLP community: Joint use of word-level and character-level features is important to effective performance of named entity recognition.

Chiu and Nichols [2015] were also inspired by the work of Collobert et alii. They used a bidirectional LSTM along with a character-level CNN resembling those used by dos Santos et al. [2015] and Labeau et al. [2015], who used it for German POS-tagging. Without using any private lexicons, detailed information about linked entities, or elaborate hand-crafted features, they were able to achieve F_1 scores of 91.62 and 86.28, respectively, on the CoNLL-2003 [Tjong Kim Sang and

De Meulder 2003] and OntoNotes [Hovy et al. 2006; Pradhan et al. 2013] datasets, forwarding the state of the art.

Lample et al. [2016] 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 conditional random field (CRF) computations [Lafferty et al. 2001]. 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 any hand-engineered features or gazetteers.

4.1.2 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, and recipients, as well as times when the events happen. Event extraction usually deals with four sub-tasks: identifying event mentions, or phrases that describe events; identifying event triggers, which are the main words—usually verbs or gerunds, sometimes infinitives—that specify the occurrence of the events; identifying arguments of the events; and identifying arguments' roles in the events. Almost all event extraction work uses supervised machine learning and depends on feature engineering, whereby clues obtained from lexical, syntactic, or knowledge-based analysis are used as features. These features are used with a classification algorithm to identify certain words as event triggers, identify the classes of events the triggers represent, et cetera. Early work by Ahn [2006] used lexicon-, syntax-, and knowledge-based features along with a memory-based nearest neighbor classifier [Daelemans et al. 2007] and a maximum entropy classifier [Daumé III 2004]. Details of many pre-ANN approaches to event extraction were discussed by Kalita [2016]. These approaches, some of which achieve high performance, are feature-driven, and as a result suffer from the drawbacks of requiring the generation of suitable feature sets and not being generalizable. In addition, because there are usually several stages, the errors tend to accumulate as they propagate from one to another.

Chen et al. [2015] proposed an architecture that automatically extracted lexical-level and sentence-level features without using complex NLP tools. They 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. They called this a dynamic multi-pooling convolutional neural network (DMCNN). In the first stage, it classified each word in a sentence as either being a trigger word or non-trigger word. If triggers were found, the second stage aligned the roles of arguments. The experimental results showed that this approach significantly outperformed other state-of-the-art methods of the time. The following year, Nguyen et al. [2016] used an RNN-based encoder–decoder pair to identify event triggers and roles, exceeding the results of Chen et alii.

4.1.3 Relationship Extraction. Another important type of information extracted from text is that of relationships. These may be possessive relationships, antonymous or synonymous relationships, or more natural relationships such as those that are familial or geographic. Note that this task is in large part a semantic task, particularly when considered in the context of neural models. Hence, discussion of underlying theory is not provided here, only a brief description of relevant approaches. One such approach is that of Zheng et al. [2017], who used a bidirectional LSTM and a CNN for entity recognition as well as relationship classification. This study exhibited the usefulness of CNNs for the task. Sun et al. [2018] 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 [Tu et al. 2016], which helped to ensure that all important information was extracted and that it was not extracted multiple times.

4.2 Text Classification

Another classic application for natural language processing is text classification, or the assignment of free-text documents to predefined classes. Document classification has numerous applications. Many techniques have produced valuable results on this task—the Rocchio algorithm used in information retrieval [Rocchio 1971], support vector machines [Joachims 1998], decision trees [Mehta et al. 1996], and the focus of this section, deep neural networks, among them.

Kim [2014] was the first to use pretrained word vectors in a CNN for sentence-level classification tasks. Kim's work with classification was motivating, and showed that simple CNNs, with one convolutional layer followed by a softmax layer with dropout, could achieve excellent results on multiple benchmarks using little hyperparameter tuning. The CNN models proposed by Kim were able to improve upon the state of the art on 4 out of 7 different tasks cast as sentence classification, including sentiment analysis and question classification. Conneau et al. [2017] later showed that networks that employ a large number of convolutional neural networks work well for document classification. Instead of using various sized convolutions, their architecture used only convolutions with a window size of three.

A novel approach proposed by Jiang [2018] used a hybrid architecture combining a deep belief network [Hinton et al. 2006] and softmax regression [Sutton and Barto 1998]. A deep belief network is a feedforward network where pairs of hidden layers are designed to resemble the restricted Boltzmann machines (RBMs) proposed by Smolensky [1986]. Restricted Bolzmann Machines are trained using unsupervised learning and are designed to increase or decrease dimensionality of data. This is achieved by making passes over the data using forward and backward propagation many times until a minimum error is found over an energy-loss-based function. 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 then combined and trained like a regular deep neural net with backpropagation and quasi-Newton methods [Fletcher 2013].

4.3 Summarization

Summarization is the task of finding elements or characteristics of interest from documents in order to produce an encapsulation of the most important information. There are two primary types of summarization techniques: extractive and abstractive. The first focuses on sentence extraction, simplification, reordering, and concatenation to relay the important information contained in documents using text taken directly from the documents. A large number of extractive summarization algorithms have been proposed over the years. These include frequency-based approaches [Edmundson 1969; Luhn 1958]; machine-learning (naive Bayes) -based algorithms [Kupiec et al. 1995]; and graph-based algorithms, computing centrality measure [Radev et al. 2004] and relative importance of sentences [Erkan and Radev 2004] using the PageRank algorithm [Page et al. 1999]. Abstractive summaries rely on expressing documents' contents through generation-style abstraction, possibly using words never seen in the documents [Jurafsky and Martin 2000]. Historically, abstractive summarization algorithms have included graph-based algorithms [Ganesan et al. 2010] as well as parse tree and graph-to-graph transformations [Liu et al. 2015]. More recently—and with increasing success—deep learning methods have been used for abstractive summarization. Deep learning approaches generally use recurrent encoder–decoder architectures.

Breaking rank was the application to summarization of the feedforward neural language model by Rush et al. [2015]. The language model used an attention-based 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 performance of encoder–decoder models. A state-of-the-art approach developed by Paulus et al. [2017] used a multiple intra-temporal 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. [2015], 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. [2017], who implemented an attention mechanism for each layer.

4.4 Question Answering

Similar to summarization and information extraction, question answering (QA) gathers specific points of data, phrases, or paragraphs. However, QA is different in that it returns this information in a coherent fashion in response to a request. This problem was historically broken into the following subtasks: question classification, passage retrieval, and answer extraction [Ezzeldin and Shaheen 2012]. Question classification determines the type of information requested and the format of the response that should be returned. In passage retrieval, extractive summarization is often used to retrieve, simplify, and combine information in an intelligent order to create a response. Very early approaches to answer extraction used a variety of methods such as simple parse-match methods, which parse the given requests and match these same tokens to sentences or phrases in retrieved documents [Phillips 1960]. Other researchers used simple first-order logic languages combined with theorem proving models such as the one proposed by Green [1969]. Current models and methods resemble those used in summarization.

Wang et al. [2017] used a gated attention-based recurrent network to match the question with an answer-containing passage. A self-matching attention mechanism was utilized to refine the machine representation by mapping the entire passage. Finally, 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 for the comprehension required. This mechanism helped shift the focus from significant words throughout passages to surrounding context windows.

Multicolumn convolutional neural networks (MCCNNs) were used by Dong et al. [2015] 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 then combined and used to rank possible answers. Training on question-answer pairs, the model, without extensive tuning, learned the logical forms automatically.

A recent novel approach was the use of relational networks (RNs). First proposed by Raposo et al. [2017], RNs are built upon an MLP architecture, where the focus is on relational reasoning, i.e. defining relationships among entities in the data. These feed-forward 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 [Santoro et al. 2017]. The RN considered all permutations to determine if there were any relationships among the sentences in a given document, or between those sentences and the presented question.

4.5 Machine Translation

Machine translation (MT) is the quintessential application of natural language processing. It involves the use of mathematical and algorithmic techniques to translate documents in one language to another. Performing effective translation is intrinsically onerous even for humans, requiring expert proficiency and prowess 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 [Jurafsky and Martin 2000]. Historically, this work included statistical approaches to phrase [Koehn et al. 2003], syntax [Yamada and Knight 2001], and word-based translation [Brown et al. 1993], each having its own limitations. Recent attempts at this task use neural machine translation (NMT), which uses end-to-end deep neural networks. The first use of such a model [Kalchbrenner and Blunsom 2013] stemmed from the success of continuous recurrent representations in capturing syntax, semantics, and morphology [Collobert and Weston 2008] in addition to the ability of recurrent neural networks to build robust language models [Mikolov et al. 2010]. This original NMT 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. Numerous novel and effective advances to this model have since been made [Bahdanau et al. 2014; Sutskever et al. 2014], as derived models are continually improving, finding answers to the shortcomings of their predecessors and overcoming any need for hand engineering [Britz et al. 2017]. 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, large-scale stacking of recurrent layers, and test time use of the beam search algorithm [Klein et al. 2017; Sennrich et al. 2017].

The standard initialization for the decoder state is that proposed by Bahsanau et al. [2014], using the last backward encoder state. However, as noted by Britz et al. [2017], 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), proposed by Sennrich et al. [2017], 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 [Bahdanau et al. 2014] from *Look, Generate, Update* (Look at input, generate output token, update hidden representation) to a process of *Look, Update, Generate* can simplify the final decoding implementation. Adding further source attributes such as morphological segmentation labels, part-of-speech tags, and syntactic dependency labels has been shown to improve models, and concatenating or factorizing these with embeddings has been shown to increase robustness further [Sennrich et al. 2017; Sennrich and Haddow 2016]. 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 two and sixteen [Britz et al. 2017], 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 [Klein et al. 2017].

In a direction diverging from previous work, some researchers have 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 [Ahmed et al. 2017; Vaswani et al. 2017]. 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 lastly, abating the path length from input to output and its handicap on the learning of the long-range dependencies which are necessary in many sequencing tasks [Hochreiter et al. 2001]. Apart from increased accuracy across translation tasks, self-attention models allow more parallelization throughout architectures, decreasing the possible training times and minimizing necessary sequential steps.

4.6 Image and Video Captioning

Image captioning is unique in that it combines the fields of natural language processing and computer vision, encoding information from images and decoding it into text. While the encoding process is out of the scope of this survey, the decoding process, as well as, to some degree, holistic models are briefly discussed here.

The automatic captioning of images has been attempted for many years, gaining popularity following the work of Duygulu et al. [2002]. The introduction of neural networks to the field has led to significant advancements in recent years. The first neural models were template based [Kulkarni et al. 2013; Yang et al. 2016] and search based [Devlin et al. 2015a,b]. In the prior, aspects of images were recognized and the different words describing these aspects were then combined in standardized grammatical templates. In the latter, images were compared with those in a database and were each assigned the caption of the most similar image.

The epitome of neural captioning models, Google's Neural Image Caption (NIC) algorithm [Vinyals et al. 2015b], used a deep CNN, trained for image classifications purposes, as the encoder and an LSTM network for decoding. Karpathy et al. [2015] developed an algorithm capturing alignments between image regions and corresponding segments of words. Hendricks et al. [2016] addressed the ability to include unseen words in captions with the Deep Compositional Captioning (DCC) model, composed of a lexical classifier and a language model working together. Xu et al. [2015] introduced image attention, feeding the decoder with dynamic vector representations of the most visually salient attributes. You et al. [2016] developed a semantic attentional model by combining top-down and bottom-up image features in a feedback process injected to the LSTM-based language model. Lu et al. [2017] proposed an adaptive attentional model that included a sentinel gate within the LSTM architecture in the decoder, dictating whether to consider the encoding or to generate the next word using the language model alone. Rennie et al. [2017] proposed the use of reinforcement learning to train a deep neural model for image captioning.

To the best knowledge of the authors, the highest performer on the vision-to-language problem, at the time of writing, uses regional attention and scene-specific contexts [Fu et al. 2017]. The algorithm segments a query image into spatial regions at a multi-scale level. Given a binary classifier, the semantic concepts captured in the image regions are considered salient for the algorithm when they are semantically meaningful (they relate to high level concepts), primitive, and contextually rich (they have dependency on other nearby regions). Simultaneously, a scene context vector is computed for the whole image. The addition of the scene vector (what the image topic is about) provides extra support to the LSTM architecture as it helps in choosing likely words given a certain image context, or in rejecting unlikely words given the same context.

An extension of image captioning is video captioning [Wu et al. 2016]. The recently emerging task is treated nearly identically, analyzing a number of frames rather than a single image [Ballas et al. 2015]. The frames are combined between the standard encoder and decoder using (a) recurrent layer(s) [Donahue et al. 2015; Venugopalan et al. 2014]. Recent approaches use hierarchical recurrent networks to perform this task [Baraldi et al. 2017; Pan et al. 2016; Yu et al. 2016]. These networks implement recurrent layers over several frames, and then implement another such layer over the final outputs of the previous layer, reducing the number of connections to a fraction inversely proportional to the number of frames captured by each recurrent implementation in the previous layer. This significantly decreases the number of units through which gradients must be backpropagated. One novel approach [Guo et al. 2016] utilized attention along with 3D convolutions. Future work will surely integrate acoustic processing for better captioning and more extensive tasks such as summarizing movies.

4.7 Summary of Deep Learning NLP Applications

Numerous other applications of natural language processing 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 towards some subject or idea [Jurafsky and Martin 2017]. Applications are varied, including product research, futures prediction, social media analysis, and classification of spam [Etter et al. 2018; Zheng et al. 2018]. The current state of the art uses an ensemble including both LSTMs and CNNs [Cliche 2017].

This section has provided a number of select examples of the applied usages of deep learning in natural language processing. 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. Initial work in many application domains often comes from academia, but then the ideas are frequently adopted, as well as expanded upon, in industry by organizations big and small, established and startup. Published academic work, as well as commercial work, is likely to continue apace, if not faster, in the years to come.

5 CONCLUSIONS

Early applications of natural language processing included a well-acclaimed but simpleminded algebra word problem solver program called STUDENT [Bobrow 1964], as well as interesting but severely constrained conversational systems such as Eliza, which acted as a "psycho-therapist" [Weizenbaum 1966]), and another that conversed about manipulating blocks in a microworld [Winograd 1971]. Nowadays, highly advanced applications of state-of-the-art NLP programs 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 Artificial Neural Networks. 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 greater "understanding" of natural languages. Both convolutional and recurrent specimen contribute to the state of the art in the field, however, it is not clear which produces superior results across the rich and varying terrain of the NLP field. Consolidating the analysis of all the models surveyed, a few general trends can be surmised. First, recurrent networks (particularly those with explicit memory, such as LSTMs and GRUs) with attention mechanisms are the strongest decoders. Second, the best decoders tend to be those that implement CNNs capped by RNNs, with the convolutional aspect seeming slightly more important. Third, while highly engineering networks usually optimizes results, there is no substitute for cultivating networks with large quantities of high quality data. Following from this final observation, it may be useful to direct more research effort toward training methodologies, rather than developing expensive, 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. Many datasets used to evaluate new models are generated specifically for that model and are then used only several more times, if at all. As the features and sizes of these datasets are highly variable, this makes comparison difficult, particularly over time. Most subfields of NLP, as well as the field as a whole, would be behooved by extensive, large-scale discussions regarding the necessary contents of such datasets, 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 due to the fact that 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 research. 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 be 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 language or the 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 eighty 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 natural language processing, but 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. Collobert et al. [2011] 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 non-neural structures, simply using ANNs on the side to make the decisions that were previously done using rules and probability models. While more complete NLP 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 [Schuman et al. 2017]. 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 [Hennessy and Patterson 2017]. Ultimately, artificial neural networks 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 coming years [Monroe 2014], the continuation of deep learning and the longevity of its success can be highly anticipated, ensuring the opportunity for sustained progress in natural language processing.

REFERENCES

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 Task 10: Multilingual Semantic Textual Similarity. In *International Workshop on Semantic Evaluation*. 81–91.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Joint Conf on Lexical and Computational Semantics-Volume 1*. 385–393.

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted Transformer Network for Machine Translation. *arXiv preprint arXiv:1711.02132* (2017).

David Ahn. 2006. The stages of event extraction. In *Workshop on Annotating and Reasoning about Time and Events*. 1–8.

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-based Parsing and Tagging with Neural Networks. In *Conf on Empirical Methods in Natural Language Processing*. 1354–1359.

James Allen. 1995. *Natural Language Understanding*. Pearson.

Peggy Andersen, Philip Hayes, Alison Huettner, Linda Schmandt, Irene Nirenburg, and Steven Weinstein. 1992. Automatic Extraction of Facts from Press Releases to Generate News Stories. In *Conf on Applied Natural Language Processing*. 170–177.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042* (2016).

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, Trevor Darrell, Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, et al. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In *IEEE Conf on Computer Vision and Pattern Recognition*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473* (2014).

Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. 2015. Delving Deeper into Convolutional Networks for Learning Video Representations. *arXiv preprint arXiv:1511.06432* (2015).

Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2017. Hierarchical boundary-aware neural encoder for video captioning. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3185–3194.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-crawled Corpora. *Language Resources and Evaluation* 43, 3 (2009), 209–226.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do Neural Machine Translation Models Learn about Morphology? *arXiv preprint arXiv:1704.03471* (2017).

Karel Beneš, Murali Karthick Baskar, and Lukáš Burget. 2017. Residual Memory Networks in Language Modeling: Improving the Reputation of Feed-Forward Networks. *Interspeech 2017* (2017), 284–288.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3 (2003), 1137–1155.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5, 2 (1994), 157–166.

Daniel G Bobrow. 1964. Natural Language Input for a Computer Problem Solving System. (1964).

Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina. [n. d.]. Proceedings of the Tenth Workshop on Statistical Machine Translation.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A Semantic Matching Energy Function for Learning with Multi-relational Data. *Machine Learning* 94, 2 (2014), 233–259.

Jan Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *International Conference on Machine Learning*. 1899–1907.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *arXiv preprint arXiv:1703.03906* (2017).

Peter Brown, Vincent Della Pietra, Stephen Della Pietra, and Robert Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics* 19, 2 (1993), 263–311.

Gino Brunner, Yuyi Wang, Roger Wattenhofer, and Michael Weigelt. 2018. Natural Language Multitasking: Analyzing and Improving Syntactic Saliency of Hidden Representations. *arXiv preprint arXiv:1801.06024* (2018).

Xavier Carreras, Lluis Marquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *Conf on Natural Language Learning*. 1–4.

Mauro Cettolo. 2016. An Arabic-Hebrew parallel corpus of TED talks. *arXiv preprint arXiv:1610.00572* (2016).

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Conf of European Association for Machine Translation*. 261–268.

Eugene Charniak. 1996. *Statistical language learning*. MIT press.

Wanxiang Che, Meishan Zhang, Yanqiu Shao, and Ting Liu. 2012. Semeval-2012 task 9: Chinese semantic dependency parsing. In *First Joint Conf on Lexical and Computational Semantics*. 378–384.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005* (2013).

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 740–750.

Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. 1998. Evaluation metrics for language models. (1998).

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Annual Meeting of the ACL*, Vol. 1. 167–176.

Zhiyi Chi and Stuart Geman. 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics* 24, 2 (1998), 299–305.

Jason Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *arXiv preprint arXiv:1511.08308* (2015).

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Conf on Empirical Methods in Natural Language Processing*. 2331–2336.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147* (2016).

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

Dan Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. 2011. Flexible, high performance convolutional neural networks for image classification. In *International Joint Conf on Artificial Intelligence*, Vol. 22. Barcelona, Spain, 1237.

Philip Clarkson and Tony Robinson. 2001. Improved language modelling through better language model evaluation measures. *Computer Speech & Language* 15, 1 (2001), 39–53.

Mathieu Cliche. 2017. BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs. *arXiv preprint arXiv:1704.06125* (2017).

Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Ng Andrew. 2013. Deep learning with COTS HPC systems. In *International Conf on Machine Learning*. 1337–1345.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*. 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12 (2011), 2493–2537.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Conf of the European Chapter of the ACL*, Vol. 1. 1107–1116.

Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Commun. ACM* 39, 1 (1996), 80–91.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing* 4, 1 (2007), 3.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional LSTM. *arXiv preprint arXiv:1606.06406* (2016).

Walter Daelemans, Jakub Zavrel, Ko Van Der Sloot, and Antal Van Den Bosch. 2007. Timbl: Tilburg memory-based learner. *Version* 6 (2007), 07–03.

Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. *arXiv preprint arXiv:1702.04521* (2017).

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Joint Annual Meeting of the ACL and International Joint Conference on Natural Language Processing*. 468–476.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. *Paper available at http://pub. hal3. name# daume04cg-bfgs, implementation available at http://hal3. name/megam* 198 (2004), 282.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Coling Workshop on Cross-framework and Cross-domain Parser Evaluation*. 1–8.

Wim De Mulder, Steven Bethard, and Marie-Francine Moens. 2015. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language* 30, 1 (2015), 61–98.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015a. Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809* (2015).

Jacob Devlin, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick. 2015b. Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467* (2015).

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *International Conf on Computational Linguistics*. 350.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2625–2634.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Annual Meeting of the ACL and International Joint Conf on Natural Language Processing*, Vol. 1. 260–269.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts.. In *COLING*. 69–78.

Pinar Duygulu, Kobus Barnard, Joao FG de Freitas, and David A Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *European conference on computer vision*. Springer, 97–112.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075* (2015).

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776* (2016).

Harold P Edmundson. 1969. New methods in automatic extracting. *J. ACM* 16, 2 (1969), 264–285.

Salah El Hihi and Yoshua Bengio. 1996. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*. 493–499.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive Science* 14, 2 (1990), 179–211.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22 (2004), 457–479.

Michael Etter, Elanor Colleoni, Laura Illia, Katia Meggiorin, and Antonino D'Eugenio. 2018. Measuring organizational legitimacy in social media: Assessing citizens' judgments with sentiment analysis. *Business & Society* 57, 1 (2018), 60–97.

Ahmed Magdy Ezzeldin and Mohamed Shaheen. 2012. A survey of Arabic question answering: challenges, tasks, approaches, tools, and future trends. In *International Arab Conf on Information Technology*. 1–8.

Laurene Fausett. 1994. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *International Conference on World Wide Web*. 406–414.

Roger Fletcher. 2013. *Practical methods of optimization*. John Wiley & Sons.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving Neural Parsing by Disentangling Model Combination and Reranking Effects. *arXiv preprint arXiv:1707.03058* (2017).

Kun Fu, Junqi Jin, Runpeng Cui, Fei Sha, and Changshui Zhang. 2017. Aligning where to see and what to tell: image captioning with region-based attention and scene-specific contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 12 (2017), 2321–2334.

Tetsu Fujisaki, Fred Jelinek, John Cocke, Ezra Black, and T Nishino. 1991. A probabilistic parsing method for sentence disambiguation. In *Current issues in parsing technology*. 139–152.

Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Bioological Cybernetics* 36 (1980), 193–202.

Kunihiko Fukushima and Sei Miyake. 1982. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition* 15, 6 (1982), 455–469.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *International Conference on Computational Linguistics*. 340–348.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* (2017).

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1, 12 (2009).

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies* 10, 1 (2017), 1–309.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conf on Neural Networks*, Vol. 1. 347–352.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT Press, Cambridge.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conf on Acoustics, Speech and Signal Processing*. 6645–6649.

Cordell Green. 1969. Theorem proving by resolution as a basis for question-answering systems. *Machine Intelligence* 4 (1969), 183–205.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* 28, 10 (2017), 2222–2232.

Zhao Guo, Lianli Gao, Jingkuan Song, Xing Xu, Jie Shao, and Heng Tao Shen. 2016. Attention-based LSTM with semantic consistency for videos captioning. In *ACM Multimedia Conference*. 357–361.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Conf on Computational Natural Language Learning: Shared Task*. 1–18.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Conf on Natural language learning at HLT-NAACL 2003-Volume 4*. 172–175.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Pacific Asia Conf on Language, Information, and Computation*. 535–544.

Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Conf on Empirical Methods in Natural Language Processing*. 1576–1586.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Conf of North American Chapter of ACL: Human Language Technologies*. 937–948.

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

Donald O. Hebb. 1949. The organization of behavior: A neuropsychological theory. (1949).

John Hennessy and David Patterson. 2017. *Computer architecture: a quantitative approach*. Elsevier.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. *arXiv preprint arXiv:1702.01569* (2017).

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29, 6 (2012), 82–97.

Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. (2001).

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Human Language Technology Conf, Companion Volume*. 57–60.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*. 2042–2050.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1. 3.

Richard Huang, Eric Hand Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Annual Meeting of the ACL: Volume 1*. 873–882.

David Hubel and Torsten Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology* 160, 1 (1962), 106–154.

W John Hutchins. 2001. Machine translation over fifty years. *Histoire Epistémologie Langage* 23, 1 (2001), 7–31.

Rukmini Iyer, Mari Ostendorf, and Marie Meteer. 1997. Analyzing and predicting language model improvements. In *IEEE Workshop on Automatic Speech Recognition and Understanding*. 254–261.

Frederick Jelinek, John Lafferty, and Robert Mercer. 1992. Basic methods of probabilistic context free grammars. In *Speech Recognition and Understanding*. Springer, 345–360.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *arXiv preprint arXiv:1511.03962* (2015).

Mingyang Jiang, Yanchun Liang, Xiaoyue Feng, Xiaojing Fan, Zhili Pei, Yu Xue, and Renchu Guan. 2018. Text classification based on deep belief network and softmax regression. *Neural Computing and Applications* 29, 1 (2018), 61–70.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European Conf on Machine Learning*. 137–142.

Karen Sparck Jones. 1994. Natural language processing: a historical review. In *Current Issues in Computational Linguistics: in Honour of Don Walker*. Springer, 3–16.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. (2016).

John Judge, Aoife Cahill, and Josef Van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *International Conf on Computational Linguistics*. 497–504.

Dan Jurafsky and James Martin. 2000. *Speech & language processing*. Pearson Education.

Dan Jurafsky and James Martin. 2017. *Speech & language processing (3rd. Edition Draft)*. Pearson Education.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Conf on Empirical Methods in Natural Language Processing*. 1700–1709.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

Jugal Kalita. 2016. Detecting and Extracting Events from Text Documents. *arXiv preprint arXiv:1601.04012* (2016).

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3128–3137.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *International Conf on Language Resources and Evaluation*. 1344–1347.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese Relevance-tagged Corpus.. In *LREC*.

Mitsuo Kawato, Kazunori Furukawa, and Ryoji Suzuki. 1987. A hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics* 57, 3 (1987), 169–185.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-Aware Neural Language Models.. In *AAAI*. 2741–2749.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810* (2017).

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, Vol. 5. 79–86.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Conf of the North American Chapter of the ACL on Human Language Technology-Volume 1*. 48–54.

Alex Krizhevsky. 2014. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997* (2014).

Girish Kulkarni, Visruth Premraj, Vicente Ordonez, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2013. Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2891–2903.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *International ACM SIGIR Conf on Research and Development in Information Retrieval*. 68–73.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained POS tagging. In *Conf on Empirical Methods in Natural Language Processing*. 232–237.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Conf on Empirical Methods in Natural Language Processing*. 729–739.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conf on Machine Learning*. 1188–1196.

Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks* 3361, 10 (1995).

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation* 1, 4 (1989), 541–551.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *International Conf on Computational linguistics-Volume 1*. 1–7.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Annual Meeting of the ACL*, Vol. 1. 457–467.

Elizabeth D Liddy. 2001. Natural language processing. (2001).

Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *Conf of the North American Chapter of the ACL: Human Language Technologies*. 1077–1086.

Yang Liu and Meng Zhang. 2018. Neural Network Methods for Natural Language Processing. (2018).

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *IEEE Conf on Computer Vision and Pattern Recognition*, Vol. 6.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2, 2 (1958), 159–165.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Conf on Computational Natural Language Learning*. 104–113.

Christopher Manning, Hinrich Schütze, et al. 1999. *Foundations of statistical natural language processing*. Vol. 999. MIT Press.

Mitchell Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19, 2 (1993), 313–330.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *International Workshop on Semantic Evaluation*. 1–8.

Warren S McCulloch and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 4 (1943), 115–133.

LR Medsker and LC Jain. 2001. *RECURRENT NEURAL NETWORKS*.

Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. 1996. SLIQ: A fast scalable classifier for data mining. In *International Conf on Extending Database Technology*. 18–32.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011a. Strategies for training large scale neural network language models. In *IEEE Workshop on Automatic Speech Recognition and Understanding*. 196–201.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Annual Conf of the International Speech Communication Association*, Vol. 2. 3.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011b. Extensions of recurrent neural network language model. In *IEEE International Conf on Acoustics, Speech and Signal Processing (ICASSP)*. 5528–5531.

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

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes* 6, 1 (1991), 1–28.

Don Monroe. 2014. Neuromorphic computing gets ready for the (really) big time. *Commun. ACM* 57, 6 (2014), 13–15.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Conf on Empirical Methods in Natural Language Processing*. 2292–2297.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conf on Machine Learning*. 807–814.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. *arXiv preprint arXiv:1707.08172* (2017).

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Conf of the North American Chapter of ACL: Human Language Technologies*. 300–309.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *International Workshop on Parsing Technologies*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. 50–57.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *International Conf on Parsing Technologies*. 73–76.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *International Workshop on Semantic Evaluation*. 915–926.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016. Hierarchical recurrent neural encoder for video representation with application to captioning. In *IEEE Conf on Computer Vision and Pattern Recognition*. 1029–1038.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A Deep Reinforced Model for Abstractive Summarization. *arXiv preprint arXiv:1705.04304* (2017).

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. FrameNet+: Fast paraphrastic tripling of FrameNet. In *Annual Meeting of the ACL*, Vol. 2. 408–413.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conf on Empirical Methods in Natural Language Processing*. 1532–1543.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-canonical Language*, Vol. 59.

Ngoc-Quan Pham, German Kruszewski, and Gemma Boleda. 2016. Convolutional neural network language models. In *Conf on Empirical Methods in Natural Language Processing*. 1153–1162.

Anthony Valiant Phillips. 1960. A Question-answering Routine. (1960).

Adam Poliak, Yonatan Belinkov, James Glass, and Benjamin Van Durme. 2018a. On the Evaluation of Semantic Phenomena in Neural Machine Translation Using Natural Language Inference. *arXiv preprint arXiv:1804.09779* (2018).

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018b. Hypothesis Only Baselines in Natural Language Inference. *arXiv preprint arXiv:1805.01042* (2018).

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Conf on Computational Natural Language Learning*. 143–152.

Dragomir Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management* 40, 6 (2004), 919–938.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 777–789.

Rajat Raina, Anand Madhavan, and Andrew Y Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In *International Conf on Machine Learning*. 873–880.

Sneha Rajana, Chris Callison-Burch, Marianna Apidianaki, and Vered Shwartz. 2017. Learning Antonyms with Paraphrases and a Morphology-Aware Neural Network. In *Joint Conf on Lexical and Computational Semantics*. 12–21.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* (2015).

David Raposo, Adam Santoro, David Barrett, Razvan Pascanu, Timothy Lillicrap, and Peter Battaglia. 2017. Discovering objects and their relations from entangled scene representations. *arXiv preprint arXiv:1702.05068* (2017).

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Transactions of the ACL* 3 (2015), 475–488.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, Vol. 1. 3.

Joseph John Rocchio. 1971. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing* (1971), 313–323.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65, 6 (1958), 386.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM* 8, 10 (1965), 627–633.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. 1985. *Learning internal representations by error propagation.* Technical Report. California Univ San Diego.

Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* (2015).

Gerard Salton and Donna Harman. 2003. *Information retrieval.* John Wiley and Sons Ltd.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems.* 4974–4983.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008* (2015).

Diana Santos, Nuno Seco, Nuno Cardoso, and Rui Vilela. 2006. Harem: An advanced ner evaluation contest for portuguese. In *International Conf on Language Resources and Evaluation Genoa Italy.*

Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Computation* 4, 2 (1992), 234–242.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. 2017. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* (2017).

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357* (2017).

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892* (2016).

Noam Shazeer, Joris Pelemans, and Ciprian Chelba. 2015. Sparse non-negative matrix language modeling for skip-grams. In *Interspeech 2015.* 1428–1432.

Paul Smolensky. 1986. *Information processing in dynamical systems: Foundations of harmony theory.* Technical Report. University of Colorado, Boulder.

Richard Socher, John Bauer, Christopher Manning, et al. 2013a. Parsing with compositional vector grammars. In *Annual Meeting of the ACL*, Vol. 1. 455–465.

Richard Socher, Eric Huang, Jeffrey Pennin, Christopher Manning, and Andrew Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems.* 801–809.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Conf on Empirical Methods in Natural Language Processing.* 1631–1642.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).

Pontus Stenetorp. 2013. Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning.*

Mingming Sun, Xu Li, Xin Wang, Miao Fan, Yue Feng, and Ping Li. 2018. Logician: A Unified End-to-End Neural Approach for Open-Domain Information Extraction. In *ACM International Conf on Web Search and Data Mining.* 556–564.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proc of Twelfth Conf on Computational Natural Language Learning.* 159–177.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems.* 3104–3112.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction.* Vol. 1. MIT press Cambridge.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Conf on Natural Language Learning at HLT-NAACL 2003-Volume 4.* 142–147.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* (2016).

M vanGerven and SM Bohte (Eds.). 2017. Artificial neural networks as models of neural information processing, Frontiers in Computational Neuroscience. (2017).

Vladimir Vapnik. 2013. *The nature of statistical learning theory.* Springer science & business media.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems.* 6000–6010.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2014. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729* (2014).

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a foreign language. In *Advances in Neural Information Processing Systems.* 2773–2781.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *IEEE Conf on Computer Vision and Pattern Recognition*. 3156–3164.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Joint Conf on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Annual Meeting of the ACL*, Vol. 1. 189–198.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Annual Meeting of the ACL and the 7th International Joint Conf on Natural Language Processing*, Vol. 1. 1332–1342.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A Neural Transition-Based Approach for Semantic Dependency Graph Parsing. (2018).

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158* (2015).

Joseph Weizenbaum. 1966. ELIZAâĂŤA computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 1 (1966), 36–45.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. [n. d.]. Inference is Everything: Recasting Semantic Resources into a Unified Evaluation Framework. In *International Joint Conf on Natural Language Processing*, Vol. 1. 996–1005.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426* (2017).

Will Williams, Niranjani Prasad, David Mrva, Tom Ash, and Tony Robinson. 2015. Scaling recurrent neural network language models. In *IEEE International Conf on Acoustics, Speech and Signal Processing*. 5391–5395.

Terry Winograd. 1971. *Procedures as a representation for data in a computer program for understanding natural language.* Technical Report. Massachusetts Institute of Technology.

Terry Winograd. 1983. *Language as a Cognitive Process: Volume 1: Syntax.* Addison-Wesley Pub. Co., Reading, MA.

Zuxuan Wu, Ting Yao, Yanwei Fu, and Yu-Gang Jiang. 2016. Deep Learning for Video Classification and Captioning. *arXiv preprint arXiv:1609.06782* (2016).

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conf on Machine Learning*. 2048–2057.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering* 11, 2 (2005), 207–238.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Annual Meeting on ACL*. 523–530.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Conf on Empirical Methods in Natural Language Processing*. 2013–2018.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *IEEE Conference on Computer Vision and Pattern Recognition*. 21–29.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Conf of the North American Chapter of ACL: Human Language Technologies*. 901–911.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *IEEE Conf on Computer Vision and Pattern Recognition*. 4651–4659.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *IEEE Conf on Computer Vision and Pattern Recognition*. 4584–4593.

Xiang Yu and Ngoc Thang Vu. 2017. Character composition model with convolutional neural networks for dependency parsing on morphologically rich languages. *arXiv preprint arXiv:1705.10814* (2017).

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation Classification via Convolutional Deep Neural Network.. In *COLING*. 2335–2344.

Lijuan Zheng, Hongwei Wang, and Song Gao. 2018. Sentimental feature selection for sentiment analysis of Chinese online reviews. *International Journal of Machine Learning and Cybernetics* 9, 1 (2018), 75–84.

Suncong Zheng, Yuexing Hao, Dongyuan Lu, Hongyun Bao, Jiaming Xu, Hongwei Hao, and Bo Xu. 2017. Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing* 257 (2017), 59–66.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Annual Meeting of the ACL and International Joint Conf on Natural Language Processing*, Vol. 1. 1213–1222.