Записки научных
семинаров ПОМИ
Том 499, 2021 г.

E. Arkhangelskaya, S. Nikolenko

DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING: A SURVEY

Abstract. Over the last decade, deep learning has revolutionized machine learning. Neural network architectures have become the method of choice for many different applications; in this paper, we survey the applications of deep learning to natural language processing (NLP) problems. We begin by briefly reviewing the basic notions and major architectures of deep learning, including some recent advances that are especially important for NLP. Then we survey distributed representations of words, showing both how word embeddings can be extended to sentences and paragraphs and how words can be broken down further in character-level models. Finally, the main part of the survey deals with various deep architectures that have either arisen specifically for NLP tasks or have become a method of choice for them; the tasks include sentiment analysis, dependency parsing, machine translation, dialog and conversational models, question answering, and other applications. **Disclaimer**: this survey was written in 2016 and reflects the state of the art at the time. Although the field of deep learning moves very quickly, and all directions outlined here have already found many new developments, we hope that this survey can still be useful as an overview of already classical works in the field and a systematic introduction to deep learning for natural language processing.

§1. Introduction

In the mid-2000s, a deep learning revolution started in machine learning. In 2005–2006, research groups led by Geoffrey Hinton at the University of Toronto and Yoshua Bengio at the University of Montreal made breakthroughs in the training of *deep neural networks*, which has since turned the world of machine learning upside down. In many important problem

Key words and phrases: deep learning, natural language processing.

This research was supported by the St. Petersburg State University, research project "Artificial Intelligence and Data Science: Theory, Technology, Industrial and Interdisciplinary Research and Applications".

domains the best results are now obtained by deep neural networks: practical applications began with speech recognition and now extend to anything from self-driving cars to the game of Go.

While artificial neural networks had been around since before the field of artificial intelligence began, the first deep architectures appeared in the 1970s, and many architectures mentioned in this review stem from the 1980s and 1990s (for a comprehensive historical survey see [261]), due to exploding and vanishing gradient problems deep neural networks were very hard to train. Solutions proposed in the mid-2000s came in the form of *unsupervised pretraining*, where a network first trains on a large dataset without labeling and then can be fine-tuned for a specific problem starting from this initial approximation. The first successful deep architectures used for pretraining either deep Boltzmann machines [258, 260] or stacked autoencoders [20]. However, new techniques in regularization such as dropout [287], batch normalization [137], and better weight initialization [98, 171], combined with much larger datasets and much larger computational resources, in particular the ability to train neural networks on GPUs [24], have made unsupervised pretraining all but obsolete in most applications. We are now able to train deeper and more expressive neural architectures than ever before, especially if we have the data.

Moreover, by now deep learning has become very much an engineering field: thanks to the automatic differentiation libraries such as Theano [24] and TensorFlow [1] and libraries that implement various neural network components, layers, and optimization algorithms such as Keras [59][1], experimenting with new neural architectures in practice has transformed from a tedious error-prone affair into a relatively easy and exciting process. Most of the models outlined here adhere to modern reproducibility standards: software for training them is either publicly available or can be implemented relatively easily with modern libraries.

In this work, we overview the field of deep learning for natural language processing. It is a burgeoning field that brings new advances every month, perhaps every week. Although it started with more or less standard architectures (recurrent and convolutional neural networks), over the last few years it has began to branch out into several quite different directions, from recursive networks for syntactic parsing to attention-based models for machine translation and memory networks for question answering.

[1]Interestingly, Keras has become part of the recently released TensorFlow 1.0.

In this survey, we have attempted a general review of the deep learning applications to natural language processing (NLP). We have attempted to collect the most relevant references but the main purpose is not only to list references but also to present the key ideas behind these important new results. Hence, in the survey we concentrate on presenting deep models, architectures, and ideas in a unified and coherent way, paying the most attention to situations when the needs of natural language processing warranted a novel neural architecture that had not been proposed before.

Before proceeding further, let us mention a few general sources that can serve as starting points: previous reviews of the field of deep learning in natural language processing [101, 207], books and general reviews of the field of deep learning [17–19, 71, 72, 102, 178, 225, 261], and general sources on machine learning, probabilistic inference, and related fields of optimization [25, 27, 38, 166, 206, 208, 227, 245, 335]. We also note one very influential work on the subject: [68] was one of the first papers that showed that deep neural networks can be used to tackle most natural language problems in a general and unified fashion, and that it may well be that all one needs is large corpora of unlabeled data. They trained models that provided results close to current state of the art for a wide variety of natural language processing problems without any task-specific engineering, starting purely from large unlabeled text corpora. This work and its precursor [67] motivated much of further NLP research with its "data-driven" mentality.

This survey is organized as follows. In Section 2, we review the basic notions of artificial neural networks and introduce the deep learning revolution: Section 2.1 covers the basics of neural architectures and backpropagation on acyclic computational graphs, Section 2.2 presents convolutional neural networks, Section 2.3 discusses recurrent neural networks, including modern architectures such as LSTM and GRU, and Section 2.4 considers regularization, gradient descent modifications, and other practical issues of training modern deep networks. Section 3 presents the foundations of modern natural language processing: the basic models of word embeddings (Section 3.1), their extensions to distributional semantics and embeddings of larger chunks of text (Section 3.2), character-level representations and models (Section 3.3), and other approaches to word embeddings (Section 3.4).

Section 4, the main part of this survey, considers several important natural language processing problems that are especially interesting both for modern approaches to NLP and as a source of new architectures for deep

learning. Section 4.1 introduces different kinds of NLP problems and considers the basic question of how to evaluate the results, which is often far from obvious in NLP, Section 4.2 considers sentiment analysis and syntactic parsing, presenting Socher's recursive neural networks for these problems, Section 4.3 considers dependency parsing and presents stack LSTMs, Section 4.4 deals with machine translation and introduces attention-based models, Section 4.5 treats dialog and conversational models, presenting the HRED encoder-decoder architecture, Section 4.6 surveys question answering and memory networks used for it, Section 4.7 presents neural topic models and hybrids of topic models and word embeddings, and Section 4.8 reviews a few other applications. We conclude with Section 5.

Disclaimer added at publication. This survey was written in the end of 2016 and reflects the state of the art at the time. Unfortunately, long review times and a couple of rejections have resulted in it being published only now, in the beginning of 2019. Although the field of deep learning moves at exponentially increasing speed, and all directions outlined here have already found many new developments, we hope that this survey can still be useful as an overview of the already classical works in the field and a systematic introduction to deep learning for natural language processing. We refer to, e.g., [70] for a more recent and more detailed overview of the field.

§2. Neural networks and deep learning

In this section, we briefly review the basic building blocks of deep learning, surveying basic building blocks that modern deep models are constructed from, basic training algorithms and important advances made over the last decade in this field.

2.1. Backpropagation: taming the gradients. Artificial neural networks are computational graphs composed of individual artificial neurons; historically, the first artificial neuron with a training algorithm, coming after first ideas of neural networks [211], was Rosenblatt's linear perceptron [251,252] that basically computes a linear combination of its inputs $\boldsymbol{x}$ with the weights $\boldsymbol{w}$ that can be trained by gradient descent or by explicitly finding the optimum by regression analysis.

Since it does not make sense to compose linear perceptrons (a composition of linear functions is still linear), to construct a neural network one uses nonlinear artificial neurons that are basically linear perceptrons

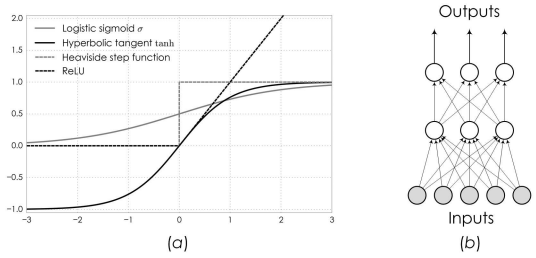

Figure 1. Neural network fundamentals: (a) nonlinear activation functions; (b) two-layer feedforward neural network.

with their output passed through a nonlinear activation function. Popular activation functions include:

- the logistic sigmoid $\sigma(a) = 1/(1+e^{-a})$, a natural choice for binary classification problems (since it is related to the likelihood of Bernoulli trials),
- the hyperbolic tangent $\tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$,
- the rectified linear unit (ReLU) $g(a) = \max(0, a)$, used historically by the first convolutional networks [88, 89], recently motivated by biological considerations [99], and also obviously very simple computationally, and
- other activation functions; various forms of nonlinear activation functions are shown on Figure 1a.

In any case, to train a single unit that operates as $f(a) = g(\boldsymbol{w}^\top \boldsymbol{x})$ with some nonlinear activation function g and error function

$$E_P(\boldsymbol{w}) = -\sum_{n \in \mathcal{M}} g(\boldsymbol{w}^\top \boldsymbol{x}) t_n$$

in a supervised way it suffices to use gradient descent of the form

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_P(\boldsymbol{w}) = \boldsymbol{w}^{(\tau)} + \eta \frac{\partial g}{\partial \boldsymbol{w}} (\boldsymbol{w}^\top \boldsymbol{x}_n) \boldsymbol{x}_n t_n.$$

In neural networks, individual units are combined into a *computational graph* that shows how the outputs of the neural network are computed via elementary functions. For example, in classical *feedforward* neural networks the graph consists of layers, with a unit on a layer connected with every unit on the previous and the following layer; a sample feedforward

architecture with one hidden and one output layer is shown on Fig. 1b. The training is usually done with various modifications of gradient descent that usually require only to be able to compute the derivatives (gradients) of the objective function with respect to the weights. And derivatives can be found automatically: the chain rule for differentiating a composition of functions means that it suffices to know gradients on the previous layer to compute the gradients on the current layer as long as we can differentiate each activation function separately: if $\frac{\partial E}{\partial s_j}$ are derivatives on the next layer, then

$$\frac{\partial E}{\partial s_i} = \sum_{j \in \mathrm{par}(i)} \frac{\partial s_i}{\partial s_j} \frac{\partial E}{\partial s_j}.$$

Thus, the chain rule leads to automatic differentiation on any graph as long as we can perform the following basic operations:

- *fprop*, forward propagation, that computes an intermediate variable using the inputs of a gate;
- *bprop*, backward propagation, that computes the gradient of the error function using the gradients of its children.

Naturally, this approach is easily extended to many different topologies: *deep* networks with several layers, *skip-layer* networks, where some weights directly connect non-adjacent layers, *sparse* networks where connections between adjacent layers are not complete bipartite graphs, and so on. This automatic differentiation (see, e.g., [325]) is precisely what is provided by basic deep learning libraries such as *theano* [15, 24] and *TensorFlow* [1]; other notable deep learning libraries, providing higher-level primitives such as ready to use neural layers and constructions, include Torch [66], Caffe [142], Keras [59], and Blocks [311].

Backpropagation is a very natural idea for training multilayer neural networks, and there are probably multiple sources where it appeared independently; Schmidhuber [261] cites early 1980s works of Werbos [323] and Speelpenning [286], earlier sources on reverse automatic differentiation [187], and mid-1980s works of Parker [238] and LeCun [173, 175] where this idea was already firmly established. However, for a long time researchers had not succeeded in training deep models, partly due to computational constraints and partly due to two closely related conceptual problems of backpropagation [128, 129]: the *vanishing gradients* problem, characteristic for feedforward networks, where already established weights in the final layers of a deep network produce near-zero gradients, and all

subsequent gradients in the backpropagation algorithm are multipled by these near-zero values, making weights in the earlier layers very hard to train, and, conversely, the *exploding gradients* problem, characteristic for recurrent networks, where the chain rule leads to exponential increases in weight updates as the recurrent network unfolds.

The first idea that began the deep learning revolution in mid-2000s was *unsupervised pretraining*: since it is hard to train all layers from the bottom up in a supervised way, and unlabeled data is often abundant and one can get much more unlabeled data than in a supervised corpus, maybe we can get better results by somehow pretraining lower levels to capture the interactions already present in unlabeled data, and then the top layers can start from there and work with features already nicely engineered by the lower levels, tweaking them slightly rather than training the whole architecture from scratch. This idea was implemented at Hinton's group with *restricted Boltzmann machines* (RBM), undirected graphical models that define a joint distribution $p(\mathbf{v}, \mathbf{h})$ on the visible variables $\mathbf{v}$ and hidden variables $\mathbf{h}$ [126, 168, 259, 294].

The other primary approach to unsupervised pretraining is based on *autoencoders*, again a very old idea continuously used in the works of LeCun and others since mid-1980s [37, 125, 127, 174, 177]. The basic idea is simple: suppose we create a neural network that simply tries to copy its input $\boldsymbol{x}$ to its output, passing $\boldsymbol{x}$ through some internal hidden representation $\mathbf{h} = f(\boldsymbol{x})$ (in the simplest case it is the representation on the hidden layer) and then reconstructing it back $\mathbf{r} = g(\mathbf{h}) = g(f(\boldsymbol{x}))$. The objective is to minimize the reconstruction error $L(\mathbf{r}, \boldsymbol{x})$. Autoencoders also enabled deep learning in the mid-2000s with the idea of *denoising autoencoders* [313–315] that regularize autoencoders by adding random noise to the input but requiring to reconstruct the output as faithfully as possible. This approach has been successfully used in image processing. Another productive way to look at denoising autoencoders is to see them as *manifold learning*: we assume that the input data occupies some complicated manifold in its corresponding space, and the problem that we ask the model to solve is to return (project, in some way) the points from a neighborhood of this manifold to the manifold itself. See also a more theoretical view of denoising autoencoders as generative models in [22].

By now, unsupervised pretraining has all but disappeared from many problem domains in deep learning, replaced with improved training techniques that we discuss in Section 2.4. Still, in natural language processing

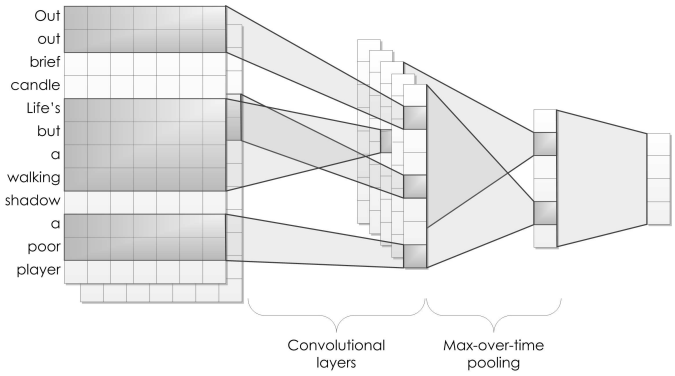

Figure 2. A convolutional neural network with 1D convolutions over a text.

unsupervised pretraining as a general idea is crucial: we cannot hope to learn language from a dataset of, say, product reviews with labeled sentiment, and have to rely on training a "language model" (in a general sense, not in the narrow sense of predicting the next word) in an unsupervised way from the whole huge corpus of texts we have available for a given language. In modern NLP, this often comes in the form of pretraining distributed representations of words and/or characters, which we will discuss in Section 3.

2.2. Convolutional neural networks. Convolutional neural networks (CNNs) consider inputs where there is some notion of "spatial distance" between input dimensions. In particular, CNNs were designed primarily for computer vision problems, where the input image has a natural notion of distance between the pixels, and it is natural that nearby pixels in an image have a much stronger relation to each other and are much more likely to belong to the same object than distant pixels. In a typical CNN, each neuron on the next layer is connected not with all, but only with a small localized subset of neurons on the previous layer; such *convolution* layers usually alternate with *pooling* layers, where activations from different neurons are pooled together. This approach mimics how the human visual cortex actually works: bottom layers distinguish local image features based on small overlapping *receptive fields*, and higher layers of the visual cortex recognize more and more advanced, abstract shapes.

In artificial neural networks, the CNN is, again, an old idea, based on studies of the visual cortex from the 1960s [136], appearing in computer science as early as late 1970s in the *Neocognitron* architecture [88, 89], and then arising again in the works of LeCun et al. in the 1990s [176]. The advances of deep learning have allowed computer vision researchers to move from handcrafted filter banks to automatic feature engineering on lower levels of deep models, and since the late 2000s CNNs have appeared in their modern form [140]. The basic ideas are illustrated on Fig. 2 for the case of a one-dimensional CNN characteristic for natural language processing:

(1) layers are connected in a sparse way: units on level k receive as input only a subset of units on level $k-1$;
(2) at the same time, each filter on a hidden layer is replicated across the entire input vector, learning the same localized features in every part of the input; this means that the weights are shared, and the total number of parameters is not so overwhelming;
(3) a *feature map* thus represents repeated applications of the same unit across all local neighborhoods, i.e., a convolution of the input with a linear filter followed by a nonlinearity; a single hidden layer can contain several feature maps;
(4) convolutional layers are usually interleaved with *pooling*, or *subsampling* layers that combine subsets of the input and output the maximum values of all features; here the idea is that a higher-level feature's exact location is less important than its interaction with other neighboring features; in one-dimensional CNNs, these are usually *max-over-time* pooling layers, which output the maximal value of a feature map along a window.

Convolutional neural networks are a natural fit for image processing and have long been applied to such problems as image classification and recognition, character recognition, image segmentation and object recognition, scene labeling, video processing, and so on; see, e.g., [179, 248]. However, in this survey we are more interested in the recent applications of CNNs to natural language processing. For example, let us consider a rather vanilla application [154], where CNNs are used for semantic sentence classification:

- the model is not as deep as computer vision models and involves only one convolutional layer with max-over-time pooling and a softmax output;

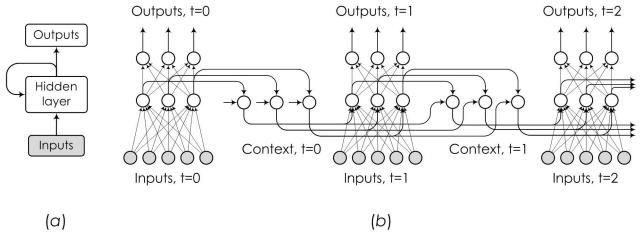

Figure 3. A sample (Elman) RNN: (a) abbreviated representation; (b) unrolled representation.

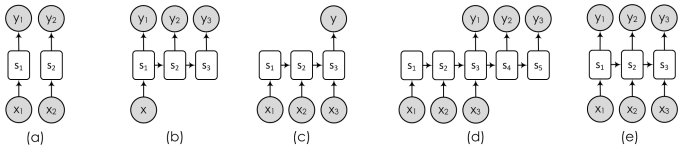

Figure 4. Various RNN-related problems [151].

- regularization is achieved through dropout; the authors report a consistent and significant improvement in accuracy with dropout across all experiments;
- the model is trained on prepared *word2vec* word embeddings (see Section 3.1) and does not attempt to tune word representations for better results;
- still, the authors report better results on such tasks as sentiment analysis and sentence classification than baseline techniques that include recursive autoencoders and recursive neural networks with parse trees (see Section 3.2).

We will refer to other applications of CNN architectures in NLP below but note here three important works: a seminal work by Collobert et al. [68], where convolutional networks were applied to achieve state of the art performance on a number of NLP tasks starting just with words as features, and works by Zhang et al. [353,354] that develop a CNN-based approach to character-level "text understanding from scratch", starting from character-level representations and applying a ConvNet architecture coming from computer vision to text understanding.

2.3. Recurrent neural networks. Previously, we have been talking about neural networks with acyclic computational graphs. If we allow

pseudo-cyclic connections in the graph, where a hidden state is carried over from a previous step of the input sequence to the next, we get *recurrent neural networks* (RNNs). One sample architecture of a classical RNN, the so-called *Elman network* [80], is shown on Fig. 3. The basic idea of an RNN is that while a feedforward ANN can only map a function from the current input to some output, RNN can, at least in principle, use the entire history of its inputs to get the current output. Besides, RNNs are naturally used for sequence learning, including *sequence-to-sequence learning*, where both input and output are sequences, such as, e.g., in machine translation; see Figure 4 for an illustration of different kinds problems that can be solved with recurrent neural networks. Due to this emphasis on sequence-based problems, RNNs are a natural fit for natural language processing, and most applications and architectures discussed below are based on recurrent networks.

The derivatives of an RNN objective function with respect to the weights are usually computed with the so-called *backpropagation through time* algorithm [324, 329]. In backpropagation through time, we compute the errors as in regular backpropagation, but this time activations on the hidden layer have an influence on the output not only through immediate activations on the output layer but also through its influence on the hidden layer activations on the next time step. The exact formulas depend on a specific architecture, but generally they involve a sum over all time steps, so training a recurrent neural network is usually significantly more computationally intensive that training a feedforward network. Note also that any RNN is "deep" by definition: previous states influence subsequence states for a long time, and any recurrent network becomes deep in the usual sense when unfolded for backpropagation through time. Still, it is often beneficial to consider deep RNNs, where various parts of the RNN architecture have multiple layers; see [239] for an overview of various such approaches.

First, an important modification of the basic RNN architecture is *bidirectional RNNs* (see Fig. 5), where not only the past but also the future context is available on every time step [262, 263]. In such problems as time series prediction this architecture would violate causality and would be impossible to apply, but it is quite natural when the entire context is available at once. In particular, this is the case for many NLP problems such as text understanding, machine translation, and the like: it makes sense to read,

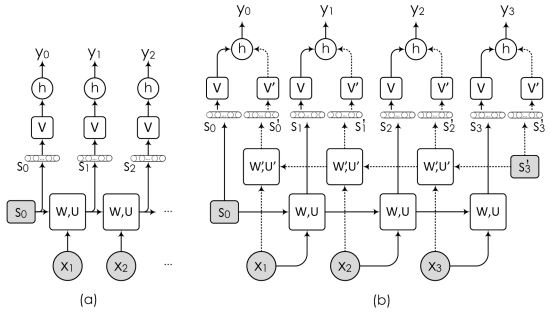

Figure 5. Unidirectional and bidirectional RNNs: (a) a regular RNN; (b) a bidirectional RNN.

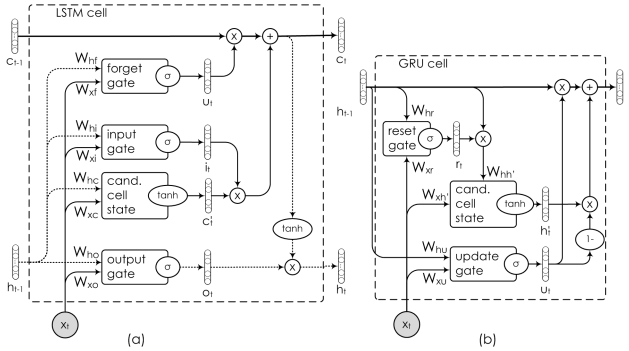

Figure 6. Modern RNN units: (a) LSTM; (b) GRU.

say, the entire sentence before translating it. Hence, bidirectional RNNs, as we will see below, find many applications in NLP.

The basic recurrent architecture, however, has its shortcomings and cannot fully express all possible temporal dependencies. In particular, the influence (gradient) of an input in a classical RNN either decreases exponentially or blows up since it has to be multiplied by the matrix of weights. Therefore, modern recurrent architectures usually make use of more complex units that implement the so-called "constant error carousel", allowing the gradients to flow through a unit unchanged if necessary.

One of the most widely used such modifications of RNNs is called the *Long Short-Term Memory* RNN (LSTM) [111]. The architecture itself was developed by Hochreiter and Schmidhuber in mid-1990s [130, 131] and

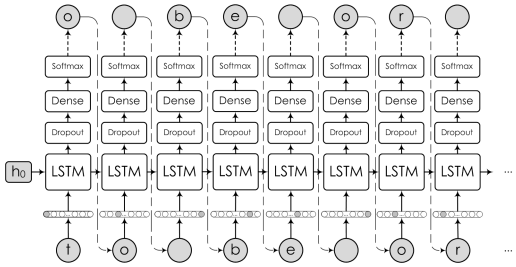

Figure 7. A simple architecture for a character-based language model.

appeared in its modern state in the works of Gers and Schmidhuber [94,95]. It has been successfully applied to numerous NLP problems. The most common (*vanilla*) LSTM architecture is shown on Fig. 6a; it contains three gates: input gate, forget gate, and output gate, together with a recurrent cell. Many different variations on this architecture have been proposed; see [111] for a detailed discussion and experimental comparison across a variety of problems. In particular, one extension of LSTMs important for NLP applications are *bidirectional LSTMs* developed by Graves and Schmidhuber [105, 106], where there are two chains of LSTM cells flowing in both forward and backward direction, similar to a bidirectional RNN.

A recent simplification of the LSTM architecture is given by *Gated Recurrent Units* (GRU) introduced by Cho et al. [57] (Fig. 6b). A GRU is very similar to an LSTM cell but simpler; GRU has a single "update gate" instead of separate forget and input gates, does not distinguish cell state and hidden state, and always exposes the entire hidden state, without a special gate for it. GRUs are simpler and thus faster to train than regular LSTMs, and recent extensive practical comparisons indicate that GRUs achieve similar performance on sequence modeling problems [62]. While the jury is still out, it is plausible that GRUs, which are growing in popularity right now, will outperform LSTMs and become the recurrent unit of choice for NLP tasks.

This overview of recurrent neural architectures naturally brings us to the first NLP problem: *language modeling*, i.e., predicting the next word or symbol in a text by previous symbols. Language modeling and, generally speaking, text generation is the natural direct application of many natural language processing systems that we touch upon in this review. We will see in Section 3.1 that word embeddings were originally applied to

language modeling, and many models and architectures from this survey can be applied to generating new text. On the other hand, even without anything NLP-specific text generation has become a standard example for sequence learning with recurrent neural networks [104], including the famous "meaning of life" example from [295] that was generated character by character with a simple agnostic RNN architecture.[2] Figure 7 shows one possible simple architecture for such sequence learning: input symbols are processed by an LSTM layer, and then LSTM features are used, through a dropout regularizer, to predict the next symbol. This idea constitutes the basis for many NLP-related architectures.

2.4. Regularization and training. Since neural networks, especially deep neural networks, have a very large number of free parameters, problems with overfitting are inevitable, and some form of regularization is required [97]. In classical feedforward neural networks, regularization often comes in the form of *weight decay* [119,164], e.g., L_2 regularization for the weights, and *early stopping*, which basically amounts to controlling the error on a validation set, using the validation error as a proxy for generalization error and stopping training when it begins to increase [243,305,342]. In the commonly used *dropout* technique [287], units in a neural network are "switched off" at random during training: in the simplest case, each unit is left in the network with a fixed probability p, independently of other units, during train time. This procedure is very simple computationally; intuitively, it can be understood as "making" each unit learn a useful feature by itself since it cannot "rely upon" other units to be present and form compositions with it. For a long time, researchers did not know how to apply dropout to recurrent networks and advised not to apply it to recurrent connections. However, recent works on *variational dropout* [91,158] led to a new Bayesian understanding of dropout and, in particular, to an understanding of how to apply dropout correctly to recurrent connections, which is crucial for NLP.

[2]Starting from the seed phrase "The meaning of life is", in one out of 10 attepmts the model generated the following: "The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger..." [295]. Similar models are easy to train and try for oneself, as shown in, e.g., the documentation for the Keras library [59].

We have already mentioned that the deep learning revolution started with unsupervised pretraining, a way to find good starting point for training the weights. In modern deep models, instead of unsupervised pretraining one often simply uses a suitable method of random initialization. One breakthrough here came in [98], which introduced *Xavier initialization*, a simple and well-motivated method for randomly initializing the weights of a neural network based on controlling the output variance. This approach was later extended to ReLU units in [120].

Another standard technique in modern deep learning, *batch normalization* [137], was designed to cope with a problem known as *covariate shift*: as one layer of a neural network changes, all layers above it get different distributions of features as input and have to adapt to continuously shifting distributions. Batch normalization transforms the activations of a layer before feeding it to the next layer so that mean and variance always remain unchanged; it has been shown to significantly speed up convergence in feedforward networks [137]. Again, it was unclear at first how to apply batch normalization to modern recurrent architectures, but a recent work [69] shows how to modify the same technique for LSTMs, which is especially relevant for NLP applications.

There are many important extensions to the basic stochastic gradient descent algorithm. One class of such extensions deals with *momentum* β, which smoothes out the samples produced by stochastic gradient descent. In its simplest form momentum works as $\bar{g} := (1-\beta)\bar{g} + \beta g$, where g is the current value of the gradient; in *adaptive* algorithms training speed and momentum can vary between components. We do not go into details but note the two currently most popular adaptive gradient descent variations, *AdaDelta* [351] and *Adam* [157].

Finally, another huge factor in the modern success of deep learning has dealt with advances in hardware. Some neural network architectures known since the 1990s are being used very successfully in modern applications but did not achieve comparable success when they were invented since it was too hard to train them. Apart from obvious advances due to Moore's law, there also has been a very important qualitative jump from CPUs to GPU hardware: a GPU provides many more, albeit weaker cores than a CPU, which is beneficial for the highly parallelizable training methods of large neural networks, achieving speedups in the range of 10x-50x over CPU implementations. At present, GPU computing libraries such as NVIDIA CUDA and the GPU chips themselves are being developed with

deep learning in mind, providing fast primitives for deep learning libraries; see, e.g., the NVIDIA CUDA Deep Neural Network library (cuDNN) [54]. Fortunately, practice has shown that for deep learning research off-the-shelf GPUs such as the NVIDIA GeForce series work as well or even better as top-of-the-line specialized GPUs such as the NVIDIA Tesla line.

§3. Distributed representations of words and text chunks

In this section, we review some of the fundamentals for using deep learning and neural networks for natural language processing, concentrating on the basic building blocks that serve as first layers and/or inputs to deep models: word and sentence/paragraph embeddings.

3.1. Word embeddings. In classical NLP, words are treated as independent entities, and the models usually start with *one-hot representations*: each word is represented as a vector of dimension $|V|$, where V is the vocabulary, which consists of zeros with a single one at the index corresponding to this word. One-hot representations suffer from an obvious flaw: in reality, words in a language are definitely not independent entities, they are very closely related to each other. It would be very useful to leverage this additional information by moving to a better representation.

Recent advances have made *distributed word representations* into a method of choice for modern natural language processing [101]. In these models, each word from the dictionary is mapped to a Euclidean space $\mathbb{R}^d$ (i.e., to a vector of d real numbers), attempting to capture semantic relationships between the words as geometric relationships in $\mathbb{R}^d$. In a classical word embedding model, one first constructs a vocabulary with one-hot representations of individual words, where each word corresponds to its own dimension, and then trains representations for individual words starting from there, similar to a dimensionality reduction problem. For this purpose, researchers have usually employed a model with one hidden layer that attempts to predict the next word based on a window of several preceding words. Then representations learned at the hidden layer are taken to be the word's features.

The modern field of word embeddings started with the work [16], subsequently extended in [21]. Extending previous work on statistical language models, usually based on word n-grams [42, 51, 103, 161], Bengio et al. proposed distributed word representations that operate as follows:

(1) for each vocabulary word $w \in V$, associate it with a feature vector (word embedding) $\mathbf{v}_w \in \mathbb{R}^d$; typical values of d lie in the hundreds;
(2) express the probability of a word appearing in its context window of size n via the vectors of previous words as

$$p(w_t|w_{t-1}, \ldots, w_{t-n}) = g(t, \mathbf{v}_{w_{t-1}}, \ldots, \mathbf{v}_{w_{t-n}}, \boldsymbol{\omega}),$$

where $\mathbf{v}_{w_{t-1}}, \ldots, \mathbf{v}_{w_{t-n}}$ are vectors of context words and g is the function computed by the neural networks with parameters $\boldsymbol{\omega}$;
(3) train from a large unlabeled text corpus both the vectors $\mathbf{v}_w$ and parameters $\boldsymbol{\omega}$; the objective maximized during training is the corpus log-likelihood

$$L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, ..., w_{t-n+1}; W, \boldsymbol{\omega}) + R(W, \boldsymbol{\omega}),$$

where W is the $|V| \times d$ matrix of weights $\mathbf{v}_w$ and $R(V, \boldsymbol{\omega})$ is a regularization term.

The two most commonly used modern models for word embeddings, *Continuous Bag-of-Words* (CBOW) and *skip-gram*, were both introduced in [214]. During its learning, a CBOW model is trying to reconstruct each word from its context with a network whose architecture is shown on Fig. 8a. The training process proceeds as follows:

(1) each input of this network is a one-hot encoded vector of size $|V|$;
(2) the hidden layer represents the matrix of vector embeddings of words W, so the jth row represents an embedding of the jth word in the vocabulary;
(3) the hidden layer's output is the average of all input vectors, $\mathbf{h} = \frac{1}{C}(\mathbf{v}_{w_1}, \ldots, \mathbf{v}_{w_C})$, where $w_1, \ldots, w_C$ are words in the context of the target word w; usually word embeddings are trained with a context spanning both past and future words;
(4) the neural network's output layer produces a score $u_w = \mathbf{h}^\top \mathbf{v}_w$ for each word $w \in V$; to obtain the posterior multinomial distribution, u_w go through the softmax to get

$$p(w|w_{t-1}, \ldots, w_{t-n}) = \frac{\exp(u_w)}{\sum_{w' \in V} \exp(u_{w'})},$$

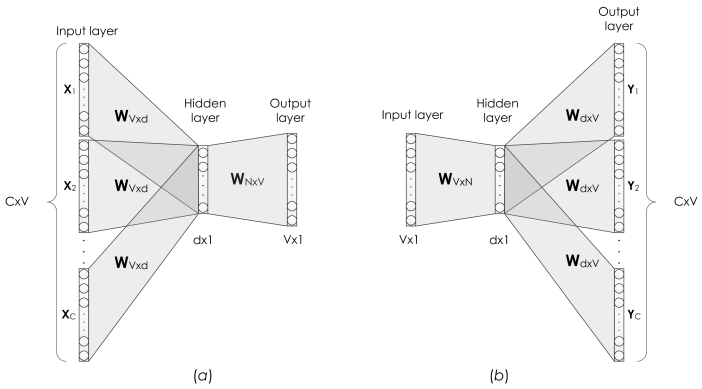

Figure 8. Network architectures for training *word2vec* models [250]: (a) CBOW; (b) skip-gram.

and the loss function E is defined as the negative log-likelihood of the true word w^*:

$$E = -\log p(w^*|w_1, \ldots, w_C) = -u_{w^*} + \log \sum_{w \in V} \exp(u_w)$$
$$= -\mathbf{h}^\top \mathbf{v}_{w^*} + \log \sum_{w \in V} \exp(\mathbf{h}^\top \mathbf{v}_w).$$

The skip-gram model operates in an inverse manner (see the architecture on Fig. 8b): the target word w^* is now at the input layer, the context words are at the output layer, the hidden layer output $\mathbf{h}$ is simply the target word's vector, $\mathbf{h} = \mathbf{v}_{w^*}$, and the loss function corresponds to C separate multinomial distributions:

$$E = -\log p(w_1, \ldots, w_C|w^*)$$
$$= -\sum_{c=1}^{C} u_{w_c} + C \log \sum_{w \in V} \exp(u_w) = -\sum_{c=1}^{C} \mathbf{v}_{w^*}^\top \mathbf{v}_{w_c} + C \log \sum_{w \in V} \exp(\mathbf{v}_{w^*}^\top \mathbf{v}_w).$$

The idea of word embeddings has been applied back to language modeling, e.g., in [215,216,220], and then, starting from [214,217], word representations have been applied for virtually all natural language processing problems, including text classification, extraction of sentiment lexicons, part-of-speech tagging, syntactic parsing and so on. Most of the models

that we review below make use of either one of the word embedding models or a character-level embedding model (see Section 3.3).

The second important model for word embeddings is *Glove* (GLObal VEctors for word representations) [241]. In the Glove model, the objective function for training word embeddings $\mathbf{v}_{w_i}$ and $\tilde{\mathbf{v}}_{w_i}$ is

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(\mathbf{v}_{w_i}^{\top} \tilde{\mathbf{v}}_{w_j} + b_i + \tilde{b}_j - \log X_{ij} \right)^2 ,$$

where $X \in \mathbb{R}^{V \times V}$ is the cooccurrence matrix between words, so X_{ij} is the frequency of word i cooccurring with word j and $X_i = \sum_j X_{ij}$ is the total number of occurrences for word i, $\tilde{\boldsymbol{w}} \in \mathbb{R}^d$ is a separate word embedding used when it is the context word, also in dimension d, and f is a function that does not overweigh frequent cooccurrences too much; usually f is bounded by a constant for large X_{ij}, e.g., $f(x) = \left(\frac{x}{x_{\max}}\right)^{\alpha}$ if $x < x_{\max}$ and 1 otherwise. The idea is to directly express $p(w_j \mid w_i) = \frac{X_{ij}}{X_i}$, the probabilities that word j occurs in the context of word i; natural requirements on the objective function (e.g., the fact that after transposing the X matrix we should replace $\mathbf{v}_{w_i}$ with $\tilde{\mathbf{v}}_{w_i}$ and vice versa) lead to this optimization problem. [241] reported improved results for named entity recognition, and since then Glove vectors have been used for many different NLP tasks.

Variations of word embeddings have been developed in the Polyglot system [3], and a completely different direction with a graph-based model is proposed in [2]. More efficient and/or stable algorithms for training word embeddings have been developed in [197, 198, 214, 221].

In a way, word embeddings rely upon what is known in linguistics as the *distributional hypothesis*: words with similar meaning will occur in similar contexts. This hypothesis appeared in computational linguistics at least as far back as the 1960s [253]; for earlier discussions and applications of the distributional hypothesis, see [167, 234, 257]. However, experiments in [241] and subsequent works also show very interesting effects in word embeddings: semantic relations between concepts represented by words turn into simple geometric relations between word vectors. This general phenomenon, namely combining lexical vectors to model the composed meaning of phrases or larger chunks of text, has become known as *distributional compositional semantics*. The phenomenon of compositional

semantics has been supported in cognitive science [218], including experimental evidence [77]; see also a standard dataset presented in [209] and a survey of vector space models of semantics [309].

Despite the fact that word embeddings have been successfully applied throughout the field, they have at least one obvious shortcoming: in the basic formulation, every word is mapped to a single vector, while in reality natural languages are highly polysemic, with numerous homonyms that often have nothing to do with each other. Rather than try to capture all possible meanings in a single vector, it might make more sense to model different meanings of the same word with different vectors, a problem known as *word sense disambiguation* for word embeddings. Recent efforts in this direction include multi-prototype word embeddings that cluster the contexts for each individual words and introduce a separate vector for every such cluster [133,134,249,331]; other models attempt disambiguation with a topic model such as LSA [344] or LDA [193], extending this approach to a unified tensor skip-gram model that trains word and topic embeddings [192]. A conceptually sound Bayesian approach would be to use nonparametric methods to model the unknown number of senses for a word and use approximate inference to cope with the huge number of resulting latent variables; this is exactly what has been recently done in [14].

3.2. Compositional semantics: sentence and paragraph embeddings. Word embeddings capture the meaning of a single word; however, it is highly dependent on the context, and the meaning of a text is not (always) the sum of meanings of its words. Hence, a natural next step after training word embeddings is to try and capture the meaning of larger chunks of text: one has to find a way to compose word embeddings into some representation of a document, or at least a text chunk such as a sentence or a paragraph. Several different approaches have been proposed for training such larger-scale embeddings:

(1) the simplest idea is to use the sum and/or mean of word embeddings to represent a sentence/paragraph; this has been used as a baseline in [172] but was proposed as a reasonable method for short phrases in [217] and has been shown to be effective for document summarization in [145]; while it is surprising that simple averaging can be an excellent and baseline, hard to beat in most applications, in our opinion this is due to the linear structure which is automatically trained in the semantic space of word embeddings; we recommend to start with

this approach and only branch out further if it proves to be unsatisfactory;
(2) in the *Distributed Memory Model of Paragraph Vectors* (PV-DM) [172], a sentence/paragraph vector is introduced as an additional vector for each paragraph; it acts as a "memory" to provide longer context than the current window;
(3) in the *Distributed Bag of Words Model of Paragraph Vectors* (PV-DBOW) [172], context words in the input are ignored, and the model is forced to predict words randomly sampled from a specific paragraph in the output; the paragraph vector is trained to help predict words from the same paragraph in a small window;
(4) in [203], convolutional neural networks are used to model sequences on tree-based n-grams that account for dependency structure, achieving state of the art results in sentiment and question classification tasks;
(5) a different convolutional architecture to model sentences has been proposed in [149];
(6) the work [160] brings the idea of skip-gram word vectors one level up, introducing the so-called *skip-thought vectors* that capture the meanings of a sentence by training from skip-grams constructed on sentences and pretrained word embeddings;
(7) in [74], distributed representations continue to the level of entire documents, concentrating on large text streams; this work constructs a hierarchical neural language model with a document level and a token level.

To compare all these different approaches, [328] proposes a method for evaluating different sentence embeddings and comparing their quality. The work [124] performs a comprehensive comparison of different approaches to learning distributed representations of sentences (rather than words) from unlabeled data. They find that the results depend on the application in mind, with deeper, more complex models better for supervised systems and shallow log-linear models preferable for representation spaces with simple spatial distance metrics. They also introduce two new interesting approaches, one based on sequential denoising autoencoders and another simplifying the SkipThought model of [160].

Another approach to implementing compositional semantics in terms of word embeddings is to treat some words as *modifiers* of others, i.e., represent some words as vectors and others as operations on the vectors. For example, the work [13] attempts to represent adjective-noun phrases by

modeling adjectives as matrices performing linear operations on the nouns that they modify. Implementing the mathematical foundations laid out in the works of Clark, Coecke, and Sadrzadeh [63–65] for an abstract categorical model of compositional meaning, the work [108] models relational words with matrices and devises a procedure to compose these representations into sentence vectors; they report improved evaluation results as evidence that the basic assumptions of the model do indeed make sense; similar results are reported in [41, 109, 152, 256]. This direction of study currently continues with representing verbs as low-rank tensors [87], multilingual models [122], modeling semantic composition through function application [235], and other works attempting a formal distributional semantics [107, 110, 219]; we do not go into more details of these approaches.

A more general and straightforward approach to such composition is taken in the works of Socher et al. on *recursive neural networks*, which we discuss in detail in Section 4.2.

3.3. Character-level representations. Word embeddings as introduced in Section 3.1 suffer from several conceptual flaws:

(1) the vectors trained for every word are completely independent; this means that we cannot really reuse our knowledge about one word to get an understanding for another, like people do; in particular, in morphology-rich languages each word comes with a plethora of different morphological forms, various derivative words in other parts of speech, derivative words formed by prefixes and suffixes, and so on; a human being understands all these derivative words immediately after he or she understands the basic word but a word embedding model would have to either cluster all of them together in the same vector or obtain a sufficient quantity of usage examples for every form, which is often impractical;
(2) the same applies to out-of-vocabulary words: a word embedding cannot be extended to new words without a reasonably sized set of usage examples while a human being can extrapolate the meaning from the form of a word; e.g., you have probably never seen the word "*polydistributional*"[3] but you already have a good idea of what it means;
(3) in practice word embedding models may grow large for large vocabularies; although applying a trained model is very fast (it is just lookup

[3]At the time of writing, it got a mere 48 hits on Google even with inexact search.

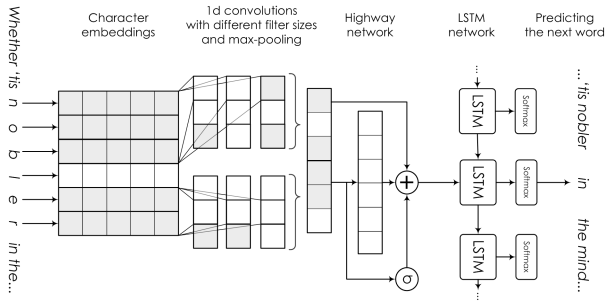

Figure 9. The character-aware neural language model from [155].

to the table of word vectors), either the model has to be stored in memory or access will still be slow.

These problems lead to the idea of *character-level representations*: what if we descend down to the most basic level of written speech and train word embeddings that take into account the actual characters that comprise a word.

First attempts at this problem involved decomposing a word into *morphemes*, the smallest units of meaning in written language [36, 200, 285]. If morphemes were available explicitly they would indeed be a perfect building block for a low-level word representation model since they are precisely what carries the meaning. However, in practice morphemes are not immediately evident from a word, and one has to rely on morphological analyzers that work imperfectly and basically introduce the need to train a separate morphology model, so the problem merely shifts to that model.

In [186], Ling et al. present a *character to word* (C2W) model for learning word embeddings based on bidirectional LSTMs (see Section 2.3). In the model, input characters are first embedded into a distributed representation themselves, and then fed into a bidirectional LSTM which outputs the word embedding. Ling et al. report state of the art results in language modeling (in terms of perplexity) and part-of-speech tagging, especially for morphology-rich languages.

Another natural approach to constructing character-level representations is based on convolutional neural networks. In [353, 354], Zhang et al. develop a method for "text understanding from scratch" based on character-level embeddings. Starting from character quantization with a simple 1-of-m encoding, they then feed this unprocessed data to a convolutional

net with 6 convolutional layers, 3 fully connected layers, and 2 dropout units between the fully connected layers for regularization. They report significant improvements for standard text classification problems.

All of these ideas have been combined in a recent work [155], which constructs a character-aware neural language model. We present this model as a characteristic example of modern character-aware efforts; the model structure is shown on Fig. 9. It begins with a distributional representation of the characters, followed by 1-d convolutions with max-pooling over time. The convolutional features from several different filter sizes are concatenated and fed into a *highway network*, a recently developed architecture that allows for training very deep models by preserving a direct path for the gradient flow through the layers [288]. Only then the results are used in a recurrent network (with LSTM units), and then used to predict the next word in the text. The authors report significantly improved language modeling, including a reduction in perplexity and reasonable nearest neighbors for out-of-vocabulary words.

Note that applying a character model is relatively expensive, and it would slow down applications significantly if one had to run such a complicated model or even simply a bidirectional LSTM for every word. Fortunately, since character-aware models usually depend only on the characters it is easy to store the representations of common words in memory, recalculating them only for rare words; this way, one can strike a proper balance between memory and computational time.

There is, however, one more interesting idea in [354] which is worth noting. In computer vision, it is common practice to augment the input datasets by slight changes in the input images. Computer vision yields itself very easily to such modifications: if we slightly crop, shift, or contract an image, change lighting conditions or downsample to reduce resolution, the objects on the image will remain the same, and the recognition target can be reused. This is not even denoising as used in denoising autoencoders, it is simply new training data for free. Computer vision is lucky to have an almost unlimited source of new training samples but in natural language processing one cannot simply change a word at random and assume that the "big picture" will remain exactly the same. Ideally, we might use human paraphrases but they are impossible to obtain in the necessary quantities. Zhang et al. [354] propose a straightforward idea for such data augmentation: use a human-generated standard thesaurus (from WordNet in their

case) and replace some words at random with their direct synonyms, reporting improved results with this augmentation. We believe that there might be other transformations helpful for NLP data augmentation, and this problem may warrant further study.

3.4. Other approaches and extensions of word embeddings. A number of works extend and refine the basic algorithms for word embeddings; some of them study and/or improve the basic idea of training word embeddings from large raw text corpora, while others attempt to incorporate additional knowledge to produce better embeddings. The RC-NET framework [333] incorporates information from knowledge graphs that contain both relational knowledge (in the form of both semantic and syntactic relations) and categorical knowledge (sets of synonyms, domain knowledge etc.) into a model to train better word embeddings, extending the skip-gram model with additional information. In [81], words are represented as regions of the semantic space rather than individual vectors, leading to improved performance in classification.

Word embeddings and related models have also been applied to other problems, e.g., in [297] paragraph vectors are used as a basic model to represent user behaviour on the Web, treating user activities as words that comprise user descriptions like words comprise a paragraph. This reuse is also a potentially rich area for further study.

We also mention a number of minor word embedding modifications either trained specifically for certain problems and domains or augmenting the model with additional explicit information: semi-supervised approaches to training word vectors have been developed in [308], while [204] introduces a mix of unsupervised and semi-supervised training specifically for sentiment analysis; the work [39] presents distributed word representations designed specifically for natural logic reasoning; in [26] word embeddings are extended with more explicit knowledge (morphological, syntactical, and semantic), with improved results on analogical reasoning and word similarity; in [114], word embeddings are trained with respect to a knowledge graph, while in [228], word embeddings are augmented with a knowledge base; the work [293] improves word representations by jointly modeling syntagmatic and paradigmatic relations, and [180] mines vector representations for linguistic regularities.

A middle ground between character-level and word-level models was struck in the approached presented by Microsoft researchers in *Deep Structured Semantic Models* (DSSM) [93,135,272]. DSSMs use sub-word embeddings obtained with *word hashing*, where a word is represented as a bag of letter trigrams. For example, "model" is encoded as

$$\{\#\text{mo}, \text{mod}, \text{ode}, \text{del}, \text{el}\#\}.$$

With this approach, the vocabulary shrinks to $|V|^3$ (tens of thousands instead of millions), but collisions are very rare; experiments in [135] show that a vocabulary with 500K words had only 22 collisions, while the vector dimension shrank from 500K to only 30K. This representation is also robust to typos and misspellings, which is very important for user-generated texts.

We have already seen that word embeddings capture some basic semantic properties in a natural way: words are semantically similar if their embeddings are close in the semantic space, and simple linear relations between word vectors may represent semantic relations between the corresponding concepts. Another interesting question about distributed representations is whether word embeddings are able to support more sophisticated properties of words than just semantic similarity. For example, the work [352] attempts to extract the logical form of sentences from unstructured text with an eye towards question answering, and there are many works concentrating on event extraction [35, 53, 231, 312].

In [49, 278], Chen, Socher et al. are using distributed representations with newly introduced *neural tensor networks* (NTN) for detection of new semantic relations in knowledge bases. In a neural tensor network model, one of the layers is bilinear and serves for multiplication of input vectors, which allows to consider more flexible operations in embedding space. Bowman et al. showing the ability of the same neural tensor networks to learn logical relations (such as entailment, reverse entailment, equivalence, alternation, negation, and independence) between concepts from distributed representations in [39] and extend this work in [40] to more complex expressions. In the latter, *natural language inference* is performed for pairs of sentences; representation of each sentence is computed in the tree-shaped part of network, and the last layer is using tree outputs as input. The whole network is then trained by maximizing the softmax objective on relation classes. Tensor networks have also been applied to machine translation [268].

In conclusion, we note that although word embeddings have become a staple of neural models in NLP, and most models that follow in the next section treat underlying word embeddings or word embedding architectures as given, this field is still far from exhausted. We believe that the problem of combining word vectors into sentence, paragraph, and/or document vectors and the problem of introducing additional explicit information to word vectors remain widely open and expect exciting developments along these lines. However, we would specifically like to advocate augmenting word embeddings with character-level models; as we have seen, they can serve as a natural extension of word embeddings but at the same time capture relations between similar words, morphological features, and even handle out-of-vocabulary words.

§4. Neural architectures for specific NLP applications

4.1. NLP tasks and evaluation problems. In this main section, we survey the most important neural architectures intended to solve specific NLP tasks. However, NLP is a very diverse field, and there are many different types of NLP problems. We divide these tasks into three broad categories. Interestingly, the harder the problems are the greater has been the contribution of deep learning.

1. Well-defined syntactic problems with semantic complications. This class contains the problems that have a clearly defined unambiguous answer and can usually be framed as classification problems with available datasets, usually syntactic problems. Examples include part-of-speech tagging, morphological segmentation (break a word up into morphemes), stemming and lemmatization, sentence boundary disambiguation and word segmentation (especially relevant for Asian languages), named entity recognition, word sense disambiguation, syntactic parsing (construct a parse tree), and coreference resolution.

Note that although these problems appear mostly syntactic in nature, they require semantic insight to answer correctly. For example, it is mostly a syntactic problem to construct a parse tree, but for many real sentences parsing hinges on common sense. E.g., in Hamlet's "For in that sleep of death what dreams may come" do the dreams come (dependency between "dreams" and "may come") or is some entity, which is currently dreaming, coming later (dependency between "what" and "may come")? The latter, rather Lovecraftian interpretation is rejected by our common sense, but

syntactically is quite possible. In many of these problems, existing techniques make errors precisely where common sense and semantic understanding is required, and it still appears very far out of our reach. Hence, while deep learning has brought new advances in this field, it has not been revolutionary yet.

2. Well-defined, but obviously semantic problems. Here we can mention language modeling, sentiment analysis, relationship/fact extraction, and question answering. Again, we know a clear answer, but the problem is obviously semantic in nature. In these problems, deep learning has brought significant advances. For example, neural language models, one of which we have considered in Section 3.3, significantly outperform their counterparts based on n-grams, and sentiment analysis has become a staple application for recurrent neural networks, including new recursive architectures we considered in Section 3.2. Since many NLP models can serve as language models and can be used for text generation, here we do not attempt to review all neural language models; see the review of deep language models in [5] and a survey of character-level neural language models in [155].

3. Finally, the most vague type of problems are those that involve text generation: text generation per se[4], automatic summarization (in the sense of writing a summary rather than just choosing the most representative sentences), machine translation, dialog and conversational models, and so on. The vagueness here comes from the ill-defined objective function: how do we objectively measure how good the output is in, say, machine translation? Even if we know a correct answer produced by a human translator, there can be plenty of equally correct or even better translations that look nothing like it. The same problem arises for conversational/dialog models: how do we evaluate whether this is a good response?

One possible answer is the *Bilingual Evaluation Understudy* (BLEU) (see [236]) metric, invented for evaluating machine translation. It represents a modification of precision, but weighted and applied to n-grams. The BLEU score for the whole corpus is computed as follows:

$$BLEU = BP \cdot \exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big), \text{ where } BP = \begin{cases} 1, & \text{if } c > r, \\ e^{1-r/c}, & \text{otherwise,} \end{cases}$$

[4]Note the difference with language modeling: a language model predicts the next word in an existing text, so it is basically solving a classification problem; it is much less clear how to evaluate "freely generated" text.

where r is the total reference length, c is the total candidate length, p_n is modified precision, and w_n are positive weights summing to one. Measured against candidate translations, BLEU avoids the pitfalls of simple precision, where a text with only one word from the candidate translation would be considered perfect, and highly correlates with human evaluations. Another highly popular measures include METEOR [170], which is the harmonic mean of unigram precision and unigram recall, Translation Edit Rate (TER) [275], which is the number of edits between the output and reference divided by the average number of reference words, and LEPOR [118] that combines basic factors and language metrics with tunable parameters. Yet another similar class includes metrics that measure the deviation in the semantic space of word embeddings, e.g., between averaged word vectors for the correct translation and averaged word vectors for the candidate translation. These metrics correlate well with BLEU and with each other but show improvements in terms of alignment with human evaluations in several reference settings; they also can be used to evaluate other sequence-to-sequence models with a dataset of known responses, e.g., conversational models. We also note some recent attempts to develop a better quality metric [48, 115, 116, 270].

Other direct evaluation metrics proposed so far in the literature on text generation and conversational models include perplexity and word classification results [266, 317], human evaluation [271], response selection task evaluated, e.g., on the Ubuntu Dialogue Corpus [196], and dialog act prediction task [150].

Apart from these direct evaluation metrics, usually the quality of a language model or text generation model is evaluated with auxiliary NLP tasks with clear evaluation criteria. For instance, the seminal work [68], while in essence presenting a language model, covers the following standard NLP tasks with well-known standard datasets: part of speech tagging evaluated on a dataset from [306], chunking, or shallow parsing evaluated on the CoNLL-2000 shared task [304], and semantic role labeling evaluated on the CoNLL-2005 shared task [47]. Later works also often include sentiment analysis as one of the standard problems (see Section 4.2).

But before we proceed, a note of caution: while BLEU/METEOR scores remain a commonly accepted evaluation standard, they are heavily criticized. The work [191] shows stunning evaluation results in the context of dialogue response generation: human evaluation scores had very small or even nonexistent correlations with all considered automated metrics! The

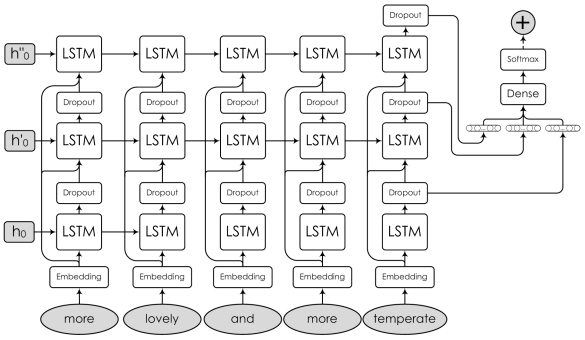

Figure 10. A deep recurrent architecture with skip-layer connections applied to sentiment analysis.

best correlations between BLEU variants and human scores were about 0.35 (on one dataset, on the other they were as low as 0.12); at the same time, human evaluations were correlated with each other at the level of 0.95 or higher, so it was not for lack of a clear gold standard, but only because the standard is so hard to formalize. This critique has already led to new automatically trained evaluation metrics [195], and we expect that this direction will bring new exciting developments.

Over the remainder of this section, we will present an overview of neural architectures used for various NLP tasks. We begin with sentiment analysis, a standard problem that lets us showcase modern recurrent architectures and then proceed to more specialized ideas such as recursive neural networks.

4.2. Sentiment analysis and recursive neural networks. Sentiment analysis [45, 188, 189, 233] is the problem of extracting subjective evaluations encoded in a natural language text, e.g., extracting from a free-text review whether a user liked the product and what specifically did she like and/or dislike. In neural network approaches to NLP, sentiment analysis has become one of the standard tasks for testing the quality of text understanding. Since there are widely available standard training corpora for sentiment analysis in English such as the Rotten Tomatoes movie review dataset [232] later augmented by parse trees by Socher et al. in the Stanford Sentiment Treebank [282], sentiment analysis is often accepted in text modeling and generation as a means of experimental evaluation, and many above-mentioned works also provide results on sentiment analysis datasets.

In particular, long short-term memory networks (LSTM) over word embeddings have been successfully applied to sentiment analysis in [319], while convolutional networks, which are good at discerning local features (e.g., sentiment words in this case), have been used for sentiment analysis in [150].

Over the last few years, recurrent architectures based on LSTM and GRU units have accumulated certain common techniques that improve their training speed and resulting quality. A sample modern recurrent architecture with three layers of LSTM units is shown on Fig. 10; in our experiments, it has produced reasonable results on sentiment analysis for short texts (tweets and blog comments). We note several important features:

- dropout is present between the layers but not between LSTMs inside a layer; this is the standard recommended practice for recurrent architectures [350]; however, recent research suggests that it is both theoretically sound and practically beneficial to use a special form of variational dropout in the LSTMs themselves, where we choose the units to be dropped once for every input sequence (sentence) and do not change them between LSTMs in the sequence [90, 158];
- there are skip-layer connections that go around a layer; on Fig. 10 we concatenated vectors from lower and higher layers; another possibility is to add them, thus making higher layers learn only the remainder of the objective which has not been already learned by previous layers; this is the basic idea of *deep residual networks* that have been instrumental in training very deep convolutional architectures for image processing [121];
- such networks are usually trained with adaptive gradient descent algorithms such as Adam [156] or AdaDelta [351].

These recurrent architectures are the staple of modern NLP based on deep learning: they take a sequence as input and can learn virtually any supervised task.

However, in sentiment analysis specifically the modern state of the art is closely related to *syntactic parsing*, i.e., constructing parse trees from natural language sentences. It is obvious that syntactic parsing must be present in a good sentiment analysis system at least implicitly, and bag-of-words approaches cannot even in principle distinguish phrases with very different sentiment: compare “not bad, it was a good movie” with “bad,

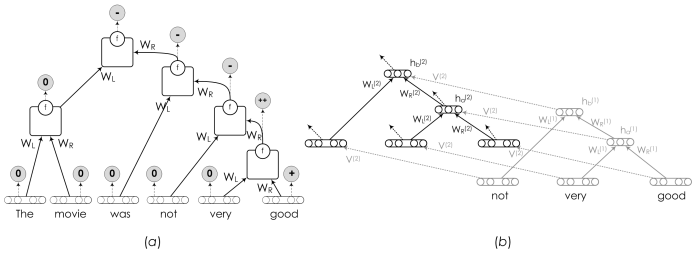

Figure 11. Recursive neural networks: (a) general model; (b) deep recursive NN.

it was not a good movie". Hence, it is natural to extend the syntactic parsing model to capture sentiment from the leaves of the parse tree up, an approach directly related to *compositional semantics* that we discussed in Section 3.2.

In particular, recursive neural networks defined first for syntactic parsing [277] were subsequently applied by Socher et al. to sentiment analysis [281, 282]. A recursive neural network is trained to compose a part of a chunk of text (usually a sentence) with another part, starting from the word vectors and working its way up to the root of the parse tree. In particular, the work [281] introduces *recursive autoencoders* (RAEs). First, note that an autoencoder is a natural way to capture compositional semantics if the parse tree is already constructed:

(1) each node is represented as a vector of length n;
(2) children of a node are concatenated to form a vector twice this length;
(3) the resulting vector serves as input for the autoencoder, the output is again a vector of length n, and it can serve as the next input up the tree.

Figure 11a shows the intuition of this approach with the example of sentiment analysis: note how the "very good" node has a highly positive label but it is flipped when combined with "not". Formally speaking, we represent a tree node v with a vector $\boldsymbol{x}_v$ and construct a recursive network as

$$\boldsymbol{x}_v = f(W_L \boldsymbol{x}_{l(v)} + W_R \boldsymbol{x}_{r(v)} + \mathbf{b}),$$

where $l(v)$ is the left child of vertex v, $r(v)$ is its right child, W_L and W_R are the corresponding weight matrices, $\mathbf{b}$ is the vector of biases, and f is a nonlinear activation function, usually either logistic or ReLU. Note that

matrices W_L and W_R remain the same throughout the tree, we are applying the same transformation at every node, hence the term "recursive".

In the *Unfolding Recursive Auto-Encoder* model (URAE) [279], a recursive autoencoder is trained to collapse all word embeddings into a single vector following the parse tree and then reconstruct back the original sentence; this single collapsed vector can be treated as a representation of the sentence/paragraph. The work [279] successfully applies this technique to paraphrasing and paraphrase detection. This work has been extended in [282] by using recursive autoencoders for sentiment analysis of sentence syntactic trees. In these works, a node is represented by both a vector and a matrix, and the composition becomes more complex, but the main recursive idea remains the same. Moreover, it is relatively straightforward to extend this architecture to *deep recursive networks*, with several layers of the same tree-like recursive architecture one above another; we have illustrated this architecture on Fig. 11b, where we show first level nodes in grey and second level nodes in black, and dashed lines show connections between layers.

This approach allows to assign labels for each node in a tree. In the example above, tree substructure with negation reversed the sentiment of its substructure, a feat impossible for the bag-of-words approach. Socher et al. report significant improvement (from 80% up to 85.4%) for sentence sentiment classification over previous methods, but this architecture can also be applied in other natural language processing tasks [40,273,279,300].

Interestingly, the same research group recently produced a work that attempts to unite tree-based reasoning with standard recurrent architectures. In [298], the basic LSTM architecture is generalized to be able to handle an arbitrary number of input vectors. This lets one organize LSTMs in a tree with an arbitrary number of children for every node; the work [298] reports improved results for sentiment analysis. On the other hand, recursive networks and tree-based LSTMs rely heavily on the quality of the parse trees and available supervision; while for sentiment analysis in English we have the Stanford Sentiment Treebank [282] which has been labeled by hand, in other applications (or even in sentiment analysis for other languages) this approach proves to be much more problematic.

Other neural network approaches to sentiment analysis include a direct application of deep RNNs over word embeddings [138], domain adaptation for sentiment classifiers with deep RNNs [100], and weakly supervised aspect-sentiment models [246]. Automated mining of of sentiment word

lexicons based on word embeddings has been proposed in [185]. We also note new large-scale sentiment evaluation datasets based on automatically mined tweets with sentiment-related hashtags and emoticons [159].

4.3. Dependency parsing and stack LSTMs. In the last section, we have seen that syntactic parsing In general, current state of the art in dependency parsing relies on *continuous-state* parsers, where the current state of the parsing algorithm is encoded as a continuous vector in a Euclidean space [9, 44, 52, 79, 289, 303, 320, 356]. This idea is similar to distributed word representations, and it is no wonder that deep learning does help in this case too, but one has to make special provisions because we are now trying to learn a stack-based algorithm rather than just an abstract function.

In particular, Dyer et al. [79] propose a transition-based parser with continuous embeddings trained from an LSTM; in their approach, the parser manipulates three basic data structures:

(1) a buffer B that contains the sequence of words, represented at time t as a vector $\mathbf{b}_t$;
(2) a stack S that stores partially constructed parses, represented at time t as a vector $\mathbf{s}_t$;
(3) a list A of actions already taken by the parser, represented at time t as a vector $\mathbf{a}_t$.

The vectors $\mathbf{b}_t$, $\mathbf{s}_t$, and $\mathbf{a}_t$ are represented as hidden states of their respective *stack LSTM*s, a modification of LSTM developed in [79]. A stack LSTM augments a regular chain of LSTMs with a "stack pointer" that indicates which output will be read; see Fig. 12 for an illustration, where x_i represent the stack contents. During the "pop" operation, the pointer is simply moved to the left, and during a "push" operation a new LSTM cell is added to the right of the current stack pointer position. To perform a "reduce" operation, the model uses a recursive neural network, composing two subtrees popped from S into a new vector which is then pushed into S.

Note that other successful approaches to syntax parsing and constituency parsing have used conditional random fields (CRFs), with latest CRF-based parsers outperforming even neural network models [117]. A recent work [78] combines CRFs with neural networks, using nonlinear potentials modeled by a neural network instead of linear potential functions based on sparse features.

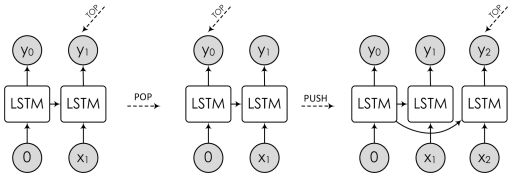

Figure 12. Stack LSTM: a "pop" operation followed by a "push" operation [79].

Another angle for further development is to take morphology into account. Morphology is obviously important for syntactic parsing, but the baseline model of [79] produced state-of-the-art results for the English language by representing each word by its vector representation (word embedding) $\boldsymbol{w}$ and a part-of-speech tag $\mathbf{t}$ provided separately; unknown words are represented by a single "UNK" token. The work [9], which considers dependency parsing in morphologically rich natural languages, produces basic representations by bidirectional character-level LSTMs, similar to [186]. The resulting representations are now available for all words, including out-of-vocabulary ones, and capture the morphological structure of a word well; the authors report experiments with a wide range of morphology-rich languages: Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish, and Turkish [9]. We mark this, and in general the use of a combination of regular and character-level word embeddings as basic models for deep learning, as an important but still incomplete development, and expect new results along these lines.

4.4. Machine translation and attention-based models. Machine translation is a natural application of large-scale language models; it is a very high-level problem that seems both theoretically tempting and obviously useful as one of the end products of NLP. Good translation is virtually impossible without deep understanding of the underlying text, so hopefully we might get the latter while aiming for the former; see also a recent review [199].

The general modern paradigm of *statistical machine translation* (SMT) models the conditional probability $p(y \mid x)$ of a target sequence y (translation) given the source sequence x (text, usually a sentence) [194]. In classical SMT, one usually approximates $\log p(y \mid x)$ with a linear combination of features and then constructs these features, mostly by hand [43, 162].

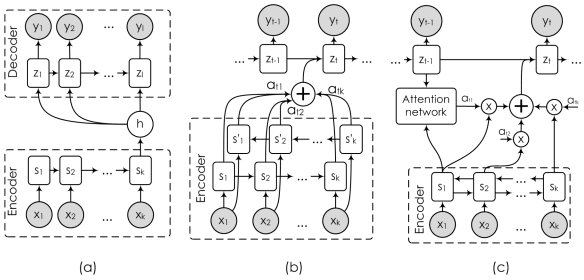

Figure 13. Encoder-decoder architectures for machine translation: (a) basic idea [147]; (b) encoder-decoder with a soft alignment model [7]; (c) encoder-decoder architecture with soft attention [55].

Neural network approaches have been used both for reranking the best lists of possible translations [264] and as part of feature functions [73]. The latter approach is being developed to this day, with state of the art results [210, 212, 230, 268, 290], but here we concentrate on the sequence-to-sequence approach.

The idea is that RNNs can be naturally used to probabilistically model a sequence [104], i.e., we can train an RNN that for a sequence $X = (x_1, x_2, \dots, x_T)$ sequentially models $p(x_1)$, $p(x_2 \mid x_1)$, …, $p(x_T \mid x_{<T}) = p(x_T \mid x_{T-1}, \dots, x_1)$, and then the joint probability $p(X)$ is just their product $p(X) = p(x_1)p(x_2 \mid x_1) \dots p(x_k \mid x_{<k} \dots p(x_T \mid x_{<T})$. This is precisely the intuition behind using RNNs for language modeling [213]: the hidden state is used as the summary for all previous history (note how this avoids the problem of variable input size, which is a big issue for feedforward networks), and we predict the next word based on the hidden state learned from all previous parts of the sequence. See [10] for a survey of neural language modeling with applications to machine translation.

However, when we try to translate a sentence, we do not really go word by word but rather first construct some semantic representation of this sentence and then unroll this representation in a different language. This intuition is formally captured in the *encoder-decoder* architecture as shown on Fig. 13a: use an RNN to construct a hidden state corresponding to the entire sequence (encoding) and then unroll it back to get a sentence in a different language (decoding) [58]. This model can be trained directly with a parallel translation corpus, as has been done in [7, 57, 147, 148, 296]; the

work [92] trains continuous phrase representations in a common semantic space and shows improvements in translation quality.

However, this is not the whole story. One problem with the encoder-decoder approach of [56] is that the entire sentence must be compressed to a single fixed-dimensional vector; hence, the quality drops dramatically for longer sentences, and a much larger model is required which may be hard to train due to lack of training data and impractical computational requirements. Some works have attempted to combat this problem by automatic segmentation [242], but the current solution to this is to introduce a *soft attention* mechanism.

One of the first attempts in this direction was made in [7], where a soft alignment model produces weights $\alpha_{t,i}$ that control how much each input word influences the word currently being translated; these weights were trained with a separate neural network. This idea is shown on Fig. 13b; note how encoder RNNs are replaced with bidirectional RNNs, so that at every word we get a "local" representation that combines the context from both left and right of the sentence but at the same time is more "focused" on the current word.

Starting from [55], the method of choice was soft attention depicted on Fig. 13c: we introduce an additional small neural network that takes as input the current decoder's hidden state and the local representation of the current word and outputs the relevance score for this specific word. This score, again denoted by α_{ij} on Fig. 13c, is then used as indicator of whether the translation model should be focusing on this specific word right now. Note that the model is trained end to end: since attention is soft (in the form of real weights), the gradients are able to flow through the entire network. It turned out that soft attention drastically improves translation for longer sentences, and is now the standard technique for machine translation.

Attention-based techniques have been extended in various ways in recent works in machine translation that represent the current state of the art:

- the work [202] addresses the problem of rare words, which has always been an issue for statistical machine translation, by explicitly training a word alignment model that outputs correspondences between specific words in the source and target sentences, which lets them use a vocabulary to substitute translations for rare words;

- the work [141] attacks the same problem from a different angle, augmenting the basic attention model with importance sampling that lets them extend the model to very large vocabularies;
- the work [201] studies different architectures of attention-based models for machine translation, reporting significant improvements just from selecting the best architecture with the same basic idea.

An influential recent work [330] shows how Google's Neural Machine Translation system (GNMT), which is the backend for Google Translate, actually operates. It is promising for NLP researchers that the basic architecture is exactly the same and consists of encoder, decoder, and attention networks. However, GNMT brings to the table a few important tricks and modifications:

- RNNs have to be deep enough to capture language irregularities, so they use 8 layers for encoder and decoder each;
- at the same time, simply stacking LSTMs does not really work, the gains completely disappear after 4–5 layers; to be able to train deeper architectures, they add residual connections between the layers, similar to [121] (see Section 4.2);
- again, the bottom layer is bidirectional in order to capture as much context as possible;
- GNMT also uses two new ideas for word segmentation:
  - the *wordpiece model* breaks words into smaller chunks (with a separate model); interestingly, this model steps originated as a word segmentation model for Asian languages but proved to be useful for European ones as well;
  - the *mixed word/character model* converts out-of-vocabulary words into characters (specifically marked so that they cannot be confused); this usually applies to proper names, e.g., `Google` might become

```
<B>G <M>o <M>o <M>o <M>g <M>l <E>e.
```

Attention-based models with similar mechanisms are not restricted to machine translation. Approaches similar to attention appeared in neural networks back in 2010 for Boltzmann machines [169]. Modifications such as the ones above can also be applied to any model based on a recurrent neural network; it is also quite natural that they have been applied to image processing [6, 222] and speech recognition [8, 60]. One could argue that in

these works, "attention" is even closer conceptually to what we usually understand by the word, since in the case of machine translation "attention" trains additional weights to find correspondences, while in image processing "attention" actually restricts the area the model is concentrating on. We note a few more applications of attention-based models relevant to NLP:

- a paper called "Show, Attend, and Tell" [334] combines a convolutional network that constructs a representation of an image with a recurrent attention-based network that generates a description;
- in [316], models with attention mechanisms are applied to syntactic parsing, achieving state of the art results with a domain-agnostic sequence-to-sequence model enhanced with attention;
- in [346], *Attention-Based Convolutional Neural Networks* (ACBNN) are introduced for modeling pairs of sentences, with three different architectures;
- in [123], an attention-based RNN reads a text, reads a question about this text, and generates an answer based on the attention matrix.

The work [86] extends attention-based statistical machine translation techniques to a multilingual model, with a single model for all languages whose number of parameters grows only linearly with the number of languages. And, finally, the work [61] explores the possibilities of constructing a machine translation model which is not based exclusively on word embeddings but augments it with a character-level model, producing a unified character-level model with machine translation, achieving state-of-the-art results. Although these results do not significantly outperform word-based approaches, the work [61] clearly shows that it is possible to construct character-level models for machine translation, and they do not break down despite the fact that text measured in characters is much longer than measured in words (recall that long sentences had been a stumbling block for machine translation); we expect new exciting work to be done along these lines.

4.5. Dialog and conversational models. Another exciting application for natural language processing based on neural networks and word embeddings are conversational and dialog models that attempt to model and predict dialogue in natural language [307]; in the case of conversational models, the model is actually supposed to actively participate in a dialog

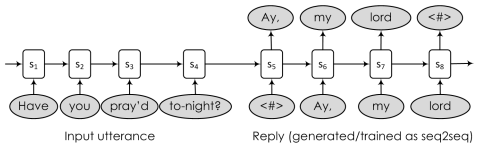

Figure 14. The *seq2seq* conversational architecture from [317].

with a human user. Note that while the objective is similar to that of chat bots that have been developed for many decades already, starting from the seminal ELIZA [321] and continuing to this day, the models we consider here concentrate on actually modeling the meaning and flow of conversation rather than selecting the most plausible answers in order to appear human to third party judges.

Dialog modeling are a characteristic example of sequence-to-sequence problems: given a sequence of words and/or symbols, the model has to produce a reasonable reply, i.e., another sequence of words/symbols. The neural conversational model introduced in [317] uses the *seq2seq* framework from [296]; see Fig. 14 for an illustration. This direct *seq2seq* approach can be easily extended to many applications, including machine translation and question answering, but unlike machine translation in this case it cannot really be expected to model the dialogue since human dialogue usually carries over the context for a very long time, pursuing long-term goals that probably cannot be modeled within *seq2seq*. Still, experiments in [317] show very reasonable dialogues both in the IT helpdesk context and in the general context of movie subtitles.[5]

The work [266] extends the hierarchical recurrent encoder decoder architecture (HRED) proposed in [284], where it was used for context-aware query suggestion for information retrieval. The basic idea of [266] is to view dialogue as a two-level system: a sequence of utterances, each of which is in turn a sequence of words. To model this two-level system, HRED trains:

(1) *encoder* RNN that maps each utterance in a dialogue into a single utterance vector;
(2) *context* RNN that processes all previous utterance vectors and combines them into the current context vector;

[5]Note a very interesting OpenSubtitles dataset based on movie subtitles that provide high-quality parallel text corpora with conversational language, priceless for both dialog modeling and machine translation [302].

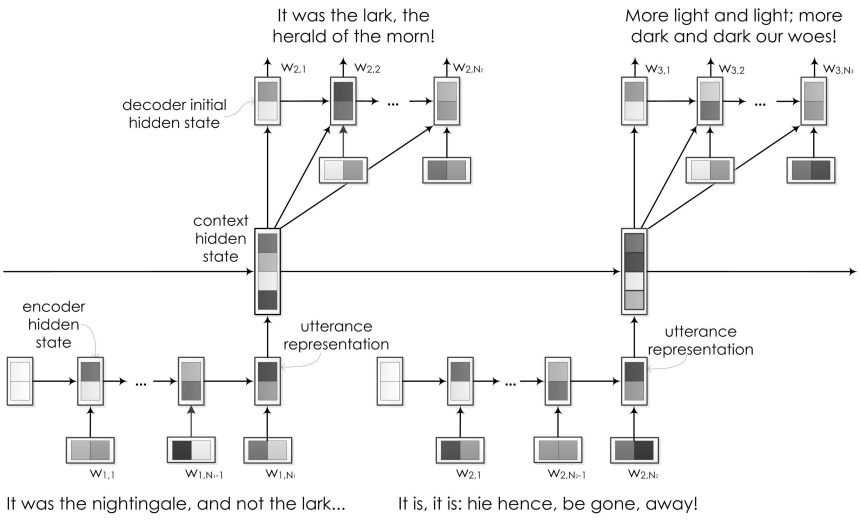

Figure 15. The HRED architecture for a conversational model from [266, 284].

(3) *decoder* RNN that predicts the tokens in the next utterance, one at a time, conditional on the context RNN.

Figure 15 shows the HRED architecture with a sample dialogue excerpt. Serban et al. use bidirectional RNNs, initialize the weights with *word2vec* representations trained on a large dataset (Google News), bootstrap from a question-answer subtitle corpus with short questions and answers, and perform the main training with the MovieTriples dataset based on the MovieDiC dataset [12], with promising results both quantitatively (in terms of perplexity) and qualitatively, judging by dialogue excerpts.

This line of work was continued in [267], which develops a variational lower bound for the hierarchical model and optimizes it; the resulting *Variational Hierarchical Recurrent Encoder-Decoder* (VHRED) model estimates latent variables in the dialogue that model the complex dependencies between individual utterances. A very recent extension [265] extends this to a different form of priors (piecewise constant), which leads to multimodal document modeling, generating responses to the time and events in the original query.

Another rising trend in dialog and conversation models is adversarial (reinforcement) learning. We begin with the work [113], which presents a model for text generation based on *Deep Q-Networks* (DQN) [224, 254], a recently developed reinforcement learningmodel that has led to many exciting advances in applications of reinforcement learning: world-class results in the game of Go [274], learning to play Atari video games from visual inputs [223], continuous control in simulated physics tasks [182], and even training neural Turing machines [349]. The model uses an encoder-decoder LSTM network similar to [296] to extract features from the input sentence, but decoding proceeds iteratively: DQN changes the output sentence and makes it closer to the input sentence (encoded by LSTMs) in terms of the BLEU score [184, 237], which is used as the reward. DQN appears to significantly outperform regular LSTM networks when decoding previously unseen sentences [113]. In other applications, the work [181] presents a more direct application of reinforcement learning to improve specifically dialogue response generation, while [291] improve dialogue systems with online active reward learning, also a reinforcement learning technique. The active reward learning uses a Gaussian process to model dialogue success, with the dialogue represented in the semantic space, where it is mapped by an encoder-decoder architecture.

Dialog and conversation models are developing very rapidly, with many new recent exciting developments. In the "attention with intention" model (see. [340]), a separate network is used to to model the intention process for the dialogue. In [322], snapshot learning is used together with some weak supervision in the form of particular events occurring in the output sequence (i.e., whether we still want to say something or have already said it). In [112], the authors propose to add an explicit copying mechanism to *seq2seq*: we often need to copy some part of the query in the response, and in [112] this copying is implemented with state changes and attention.

4.6. Question answering and memory networks. General free-text question answering (QA) is one of the hardest NLP challenges; this is one of the problems that come closest to true language understanding. Apart from direct applications for answering questions, a sufficiently general solution for question answering can also be applied to a wider range of tasks: for instance, some NLP problems can also be formulated as questions about text, e.g., "Who is an author of this story?" or "What is it about?".

While question answering in terms of trivia-like questions has long reached very high levels, with IBM Watson outperforming top *Jeopardy!*

Task 1: *Single Supporting Fact* Mary went to the bathroom. John moved to the hallway. Mary travelled to the office. Where is Mary? **A**: *office*	**Task 4**: *Two Argument Relations* The office is north of the bedroom. The bedroom is north of the bathroom. The kitchen is west of the garden. What is north of the bedroom? **A**: *office* What is the bedroom north of? **A**: *bathroom*
Task 7: *Counting* Daniel picked up the football. Daniel dropped the football. Daniel got the milk. Daniel took the apple. How many objects is Daniel holding? **A**: *two*	**Task 10**: *Indefinite Knowledge* John is either in the classroom or the playground. Sandra is in the garden. Is John in the classroom? **A**: *maybe* Is John in the office? **A**: *no*
Task 15: *Basic Deduction* Sheep are afraid of wolves. Cats are afraid of dogs. Mice are afraid of cats. Gertrude is a sheep. What is Gertrude afraid of? **A**: *wolves*	**Task 20**: *Agent's Motivations* John is hungry. John goes to the kitchen. John grabbed the apple there. Daniel is hungry. Where does Daniel go? **A**: *kitchen* Why did John go to the kitchen? **A**: *hungry*

Figure 16. Sample question answering tasks proposed in [326].

contestants in 2011 [85], actual comprehension of the questions that require human-like reasoning remains a very challenging problem. Again, we begin with evaluation. To ensure that some QA system is performing well not only due to a large and elaborate knowledge base but with actual text comprehension, the work [326] proposed a set of fairly simple tasks consisting of questions and supporting facts. These questions do not require any special knowledge, and all of them can be easily answered by a human, but they do require reasoning and understanding of semantic structure; we show some sample problems from [326] on Fig. 16. Current QA models are often tested on these tasks with various levels of access to supporting facts and external resources. Note also the work [318] that constructs deep models for adding new information to knowledge bases.

While these tests show the reasoning abilities of a model, question answering is still tightly connected with information retrieval. An important part of question answering is to find semantically close entities; see a survey [163] and references therein. Recent works in this direction map questions to logical queries over a graph of facts, a knowledge base, constructed in the system [23, 341]. Deep learning approaches have been used in this direction; e.g., [139] uses DT-RNN, a model introduced in [280], for mapping answers and questions from the Quiz Bowl game to the same semantic space using the max-margin objective. The correct answer is assumed to

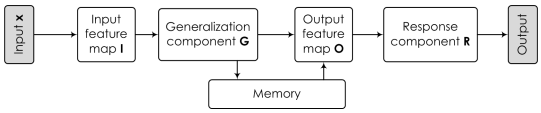

Figure 17. The general IGOR architecture of memory networks

be the one closest to the question. The modularity and flexibility of this model make it appealing for other tasks as well.

The current state of the art in question answering as shown in [326] are *memory networks*. The basic memory network model is presented in [327]. A memory network consists of an array of objects (memory) and the following components (that are learned during training; see Fig. 17 for the architecture flowchart):

- I (input feature map) that converts the input to the internal feature representation;
- G (generalization) that updates old memories after receiving new input;
- O (output feature map) that produces new output given a new input and a memory state;
- R (response) that converts the output of O into the output response format (e.g., text).

In [327], memory networks improved the state of the art in question answering, and extensions have already been developed since then. In [165], Kumar et al. present *dynamic memory networks*, a model that changes memory to an episodic memory module that chooses which parts of the input to focus on with an attention mechanism, with improved results in question answering and other applications including sentiment analysis and part-of-speech tagging. The work [292] presents an extension of memory networks called *end-to-end memory networks* that can be viewed as a continuous version of memory networks; the difference is that the latter require supervision on each layer of the network, while end-to-end memory networks can be trained with input-output pairs. The work [292] reports state of the art results on question answering and language modeling. Memory networks have been applied to large-scale simple question answering with transfer learning [33]; other applications include learning algorithms [144, 146] and an interesting application to answering visual questions, i.e., questions about images [339].

Yih et al. [301], on the other hand, combine a more classical approach of knowledge database queries with deep learning. The main idea is that apart from simple similarity of question and answer, we should also capture the relation to the answer entity. The scope of this work is limited to single-relation questions only. Extraction is done by a CNN-based semantic model, and the inputs are processed by a technique proposed in [135]: the words are first segmented to letter-n-grams and then represented as count-vectors of n-grams. Convolutional layer of the network is then used to embed local window vectors to contextual features, and max-pooling layers are supposed to extract key words, critical to answering the question. In [76], question answering is done with multi-column convolutional neural networks, an architecture with parallel vertical columns instead of fully connected layers.

However, deep structures are not the only possible approach. The key foundation for most of the current work is word embeddings and, as Zhou et al. demonstrate in [355], a proper embedding model can lead to significant improvements over the baseline even with a simple bag-of-words approach; compare also with approaches of [82, 343].

What current state of the art question answering systems generally lack is the ability for the so-called *common sense reasoning*, i.e., understanding of various aspects of the world that humans find so easy and natural; this is clear from the questions in Table 16. Note that while the best results on those was given by extended MemNN introduced in [327], it still failed on some questions even when supporting facts were provided, so QA models still have a very long way to go. Current attempts to model common sense reasoning are usually of logical nature [4, 11, 205], and it would be an interesting challenge to bring them together with neural network approaches.

A recent work from Google DeepMind [123] discusses how one can define that a machine has actually learned to read and comprehend written text. There are very few datasets on text comprehension, and supervised learning has been virtually absent from this field since it is unclear even simply what kind of a supervised dataset one could need. In [123], the authors propose a relatively simple and straightforward way to convert unlabeled datasets in the form of, say, a newspaper article and its matching summary, into (context, query, answer) triples that could then be used for supervised training of text comprehension models; their approach to creating question answering datasets may lead to new breakthroughs in machine reading comprehension.

4.7. Topic models with word embeddings. There have been several attempts to use distributed word representations to construct topic models. For example, the Neural Topic Model developed by Cao et al. [46] models both topic-word and document-topic distributions with neural networks, training n-gram embeddings together with document-topic embeddings; this model has also been extended to the supervised setting.

Yang et al. [337] model a topic as a Gaussian cluster in the semantic space, thus making the topic model into a Gaussian mixture. Word embeddings are trained with the usual models, trying to predict the current word from its near context, but embeddings are sampled from a Gaussian mixture; an Expectation-Maximization scheme is used to train both mixture parameters and word embeddings.

In a recent work [310], Tutubalina and Nikolenko developed a new technique that extends existing approaches to sentiment-based topic modeling; it is also called aspect-based opinion mining in this field since these models are usually applied to user reviews and opinions regarding certain products with the goal to mining specific aspects [190]. Using already existing pretrained word embeddings, the work improved sentiment classification in such sentiment topic models as JST and Reverse-JST [183], ASUM [347], and USTM [348], training new aspect-specific lexicons of sentiment words based on a small set of "seed" sentiment words.

The work [247] presents a new topic modeling algorithm, *Vec2Topic*, that combines distributed representations and topic modeling in a new way. Specifically, they introduce two new interesting metrics for a word's topical relevance:

(1) *depth*: given a hierarchical (agglomerative) clustering of all word vectors from the corpus' vocabulary, the depth of a word w is the number of links from the root of the clustering graph to w; this means that if a word is part of a tighly linked cluster of similar words, it is probably going to be deeper than a word which is all alone;
(2) *degree*: the degree of a word w is the number of unique words cooccurring in the same sentence with w; the idea here is that most important words, words that define their respective topics, should probably be used in many different contexts, cooccurring with many other words.

Combining depth and degree in a multiplicative score of a word, the authors extend it to the score of a topic, and then mine the top scoring topics, by which they mean clusters of word vectors in the the semantic

space. This approach is reported to produce good topics [247], and it is computationally very efficient.

Still, this field seems to be open for new research, and it appears that existing neural topic models can be significantly improved.

4.8. Other applications. One important natural problem in natural language processing is *semantic matching* of text chunks, i.e., semantic similarity between sentences, paragraphs, or other parts of text; this is relevant, for instance, for paraphrasing and clause coherence [32,336]. In [132], a convolutional architecture is used to match sentences. Unlike previous works, in [345], a convolutional architecture is used to match semantic similarity across different levels of granularity (providing what is called *multigranular comparability* in the architecture). Experiments in [345] show promising results in paraphrase identification. This is also very similar to evaluating *text similarity*, another common problem for testing various language models. The work [153] considers the text similarity problem for short texts, suggesting several metrics for combining word embeddings into semantic representations of short texts; the results are evaluated on the standard Microsoft Research Paraphrase Corpus data set [75, 332].

Relational learning is another important factor: it many application domains, multirelational data in the form of graph-like structures, sometimes called *knowledge bases* where nodes represent entities and edges represent relations between these entities have become widely available and have to be used in intelligent data mining/processing. In the case of natural language processing well-known examples of such knowledge bases are provided by, e.g., WordNet [30, 83, 84] and Semantic Web resources such as FreeBase [28]. In general, this field is known as *statistical relational learning* [96]. While these knowledge bases have been previously used to help solve NLP problems [229, 276], deep learning has led to new approaches proposed for relational learning.

In [31], Bordes et al. develop a deep learning approach to constructing *meaning representations* based on WordNet synsets, getting entity embeddings rather than word embeddings, reporting state of the art results on semantic parsing and word sense disambiguation. In particular, Bordes et al. introduce a semantic matching energy function which was later extended in [32] to other kinds of multirelational data, where it provided state of the art results on link prediction in a wide variety of multirelational datasets.

Neural networks have been used for learning to rank with text pairs, a problem most relevant for information retrieval (a search needs to rank query-document pairs) but also relevant for question answering. In information retrieval, word embeddings are used and retrained to learn good semantic representations for search queries and Web documents that can be easily matched with each other, with search results ranking as the final goal in mind. The work [272] presents a *convolutional latent semantic model* (CLSM) trained on clickthrough data. In [283], the classical RankNet architecture is modified to achieve personalized search with the help of deep models. A recent work [255] concentrates on extracting answer-entailing structures under the assumption that there is a part of text that explicitly explains the answer to a given question. In [269], convolutional neural networks are used to develop a joint representation for short text pairs. In a related application to Web search, recurrent networks (both Elman RNNs and LSTMs) serve as a basis for a click model, where user clicks in web search results are used to rerank query-document pairs and improve search results [34].

Domain adaptation is the problem of adapting the models from one domain to another; in NLP, this means using one kind of texts (e.g., literary works from the XIX and XX century) to process another kind (e.g., user-generated content on the Web) [143]. Unsupervised domain adaptation is usually based on feature cooccurrence statistics; recent methods for domain adaptation include learning such features via denoising autoencoders [50, 240], domain disambiguation techniques to be applied during the training of word embeddings [29], and learning feature embeddings for domain features with a model similar to word embeddings [338]. The domain adaptation problem is also naturally useful for word sense disambiguation [244].

§5. Conclusion

In this survey, we have seen plenty of applications for deep neural architectures in natural language processing. For many problems, approaches based on modern neural networks have outperformed previous state of the art techniques, almost the entire field of NLP has already been transformed by deep learning, and this process will no doubt continue in the future.

However, deep learning for NLP has not yet been quite as revolutionary as in some other fields. While in image processing tasks such as face

recognition recent advances bring neural networks to human-level performance [299], nothing even close to that has so far happened for NLP problems. In our opinion, this is in a big part due to the fact that even NLP problems that appear purely syntactic on the surface often have deep semantic implications. For example, let us rephrase a problem from the Winograd Schema challenge, a competition for commonsense reasoning models [226], as anaphora resolution. Which noun does the pronoun "it" refer to in the following sentences:

- the suitcase did not fit in the trunk because it was too big;
- the suitcase did not fit in the trunk because it was too small.

Any human familiar with car trunks and suitcases will have no problem with parsing these sentences, but syntactically they are absolutely identical, and a machine learning model would have to know quite a lot about the actual physical environment to be able to give a better than random answer.

We believe that commonsense reasoning and capturing this "intuitive" information that any human learns in the first years of his or her life is a major obstacle to further NLP progress. It might be one of the keys to not only superficially passing the Turing test but also achieving true text understanding, which appears to be a requirement for general artificial intelligence. But even without these prospects, which for now remain rather far-fetched, neural networks for natural language processing represent a burgeoning field where much has been done, more is being done right now, and yet more remains to be done in the future.

References

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. ManΓc, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. ViΓcgas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015, Software available from tensorflow.org.
2. C. Aggarwal, P. Zhao, *Graphical models for text: A new paradigm for text representation and processing*, SIGIR '10, ACM, 2010, pp. 899–900.
3. R. Al-Rfou, B. Perozzi, S. Skiena, *Polyglot: Distributed word representations for multilingual nlp*, Proc. 17th Conference on Computational Natural Language Learning (Sofia, Bulgaria), ACL, 2013, pp. 183–192.

4. G. Angeli, C. D. Manning, *Naturalli: Natural logic inference for common sense reasoning*, Proc. 2014 EMNLP (Doha, Qatar), ACL, October 2014, pp. 534–545.
5. E. Arisoy, T. N. Sainath, B. Kingsbury, B. Ramabhadran, *Deep neural network language models*, Proc. NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, ACL, 2012, pp. 20–28.
6. J. Ba, V. Mnih, K. Kavukcuoglu, *Multiple object recognition with visual attention*, ICLR'15, 2015.
7. D. Bahdanau, K. Cho, Y. Bengio, *Neural machine translation by jointly learning to align and translate*, arXiv (2014).
8. D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, Y. Bengio, *End-to-end attention-based large vocabulary speech recognition*, arXiv (2015).
9. M. Ballesteros, C. Dyer, N. A. Smith, *Improved transition-based parsing by modeling characters instead of words with lstms*, Proc. EMNLP 2015 (Lisbon, Portugal), ACL, 2015, pp. 349–359.
10. P. Baltescu, P. Blunsom, *Pragmatic neural language modelling in machine translation*, NAACL HLT 2015, 2015, pp. 820–829.
11. L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, N. Schneider, *Abstract meaning representation for sembanking*, Proc. 7th Linguistic Annotation Workshop and Interoperability with Discourse (Sofia, Bulgaria), ACL, August 2013, pp. 178–186.
12. R. E. Banchs, *Movie-dic: A movie dialogue corpus for research and development*, ACL '12, ACL, 2012, pp. 203–207.
13. M. Baroni, R. Zamparelli, *Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space*, EMNLP '10, ACL, 2010, pp. 1183–1193.
14. S. Bartunov, D. Kondrashkin, A. Osokin, D. P. Vetrov, *Breaking sticks and ambiguities with adaptive skip-gram*, Proc. 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016, 2016, pp. 130–138.
15. F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, Y. Bengio, *Theano: New features and speed improvements*, Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.
16. Y. Bengio, R. Ducharme, P. Vincent, *A neural probabilistic language model*, Journal of Machine Learning Research **3** (2003), 1137–1155.
17. Y. Bengio, *Learning deep architectures for ai*, Foundations and Trends in Machine Learning **2** (2009), no. 1, 1–127.
18. Y. Bengio, *Practical recommendations for gradient-based training of deep architectures*, Neural Networks: Tricks of the Trade - Second Edition, 2012, pp. 437–478.
19. Y. Bengio, A. Courville, P. Vincent, *Representation learning: A review and new perspectives*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), no. 8, 1798–1828.
20. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, *Greedy layer-wise training of deep networks*, NIPS'06, MIT Press, 2006, pp. 153–160.
21. Y. Bengio, H. Schwenk, J.-S. SenΓccal, F. Morin, J.-L. Gauvain, *Neural probabilistic language models*, Innovations in Machine Learning, Springer, 2006, pp. 137–186.

22. Y. Bengio, L. Yao, G. Alain, P. Vincent, *Generalized denoising auto-encoders as generative models*, arXiv (2013).
23. J. Berant, A. Chou, R. Frostig, P. Liang, *Semantic parsing on Freebase from question-answer pairs*, Proc. 2013 EMNLP (Seattle, Washington, USA), ACL, October 2013, pp. 1533–1544.
24. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, *Theano: a CPU and GPU math expression compiler*, Proc. Python for Scientific Computing Conference (SciPy), 2010, Oral Presentation.
25. D. P. Bertsekas, *Convex analysis and optimization*, Athena Scientific, 2003.
26. J. Bian, B. Gao, T.-Y. Liu, *Knowledge-powered deep learning for word embedding*, Machine Learning and Knowledge Discovery in Databases, Springer, 2014, pp. 132–148.
27. C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006.
28. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, *Freebase: A collaboratively created graph database for structuring human knowledge*, , SIGMOD '08, ACM, 2008, pp. 1247–1250.
29. D. Bollegala, T. Maehara, and K.-i. Kawarabayashi, *Unsupervised cross-domain word representation learning*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 730–740.
30. F. Bond, K. Paik, *A Survey of WordNets and their Licenses*, GWC 2012, 2012, p. 64–71.
31. A. Bordes, X. Glorot, J. Weston, Y. Bengio, *Joint learning of words and meaning representations for open-text semantic parsing*, JMLR, 2012.
32. A. Bordes, X. Glorot, J. Weston, Y. Bengio, *A semantic matching energy function for learning with multi-relational data*, Machine Learning **94** (2013), no. 2, 233–259.
33. A. Bordes, N. Usunier, S. Chopra, J. Weston, *Large-scale simple question answering with memory networks*, arXiv (2015).
34. A. Borisov, I. Markov, M. de Rijke, P. Serdyukov, *A neural click model for web search*, WWW '16, ACM, 2016, p. to appear.
35. E. Boros, R. Besançon, O. Ferret, B. Grau, *Event role extraction using domain-relevant word representations*, Proc. 2014 EMNLP (Doha, Qatar), ACL, October 2014, pp. 1852–1857.
36. J. A. Botha, P. Blunsom, *Compositional morphology for word representations and language modelling*, Proc. 31th ICML, 2014, pp. 1899–1907.
37. H. Bourlard, Y. Kamp, *Auto-association by multilayer perceptrons and singular value decomposition*, Manuscript M217, Philips Research Laboratory, Brussels, Belgium, 1987.
38. O. Bousquet, U. Luxburg, G. Ratsch (eds.), *Advanced lectures on machine learning*, Springer, 2004.
39. S. R. Bowman, C. Potts, C. D. Manning, *Learning distributed word representations for natural logic reasoning*, arXiv (2014).
40. _____, *Recursive neural networks for learning logical semantics*, arXiv (2014).

41. A. Bride, T. Van de Cruys, N. Asher, *A generalisation of lexical functions for composition in distributional semantics*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 281–291.
42. P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, J. C. Lai, *Class-based n-gram models of natural language*, Comput. Linguist. **18** (1992), no. 4, 467–479.
43. P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, R. L. Mercer, *The mathematics of statistical machine translation: Parameter estimation*, Comput. Linguist. **19** (1993), no. 2, 263–311.
44. J. Buys, P. Blunsom, *Generative incremental dependency parsing with neural networks*, Proc. 53rd ACL and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Vol. 2: Short Papers, 2015, pp. 863–869.
45. E. Cambria, *Affective computing and sentiment analysis*, IEEE Intelligent Systems **31** (2016), no. 2.
46. Z. Cao, S. Li, Y. Liu, W. Li, d H. Ji, *A novel neural topic model and its supervised extension*, Proc. 29th AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA., 2015, pp. 2210–2216.
47. X. Carreras, L. Marquez, *Introduction to the conll-2005 shared task: Semantic role labeling*, CONLL '05, ACL, 2005, pp. 152–164.
48. B. Chen, H. Guo, *Representation based translation evaluation metrics*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 2: Short Papers (Beijing, China), ACL, July 2015, pp. 150–155.
49. D. Chen, R. Socher, C. D. Manning, A. Y. Ng, *Learning new facts from knowledge bases with neural tensor networks and semantic word vectors*, International Conference on Learning Representations (ICLR), 2013.
50. M. Chen, Z. E. Xu, K. Q. Weinberger, F. Sha, *Marginalized denoising autoencoders for domain adaptation*, Proc. 29th ICML, icml.cc / Omnipress, 2012.
51. S. F. Chen, J. Goodman, *An empirical study of smoothing techniques for language modeling*, ACL '96, ACL, 1996, pp. 310–318.
52. X. Chen, Y. Zhou, C. Zhu, X. Qiu, X. Huang, *Transition-based dependency parsing using two heterogeneous gated recursive neural networks*, Proc. EMNLP 2015 (Lisbon, Portugal), ACL, 2015, pp. 1879–1889.
53. Y. Chen, L. Xu, K. Liu, D. Zeng, J. Zhao, *Event extraction via dynamic multi-pooling convolutional neural networks*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 167–176.
54. S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, *cudnn: Efficient primitives for deep learning*, arXiv (2014).
55. K. Cho, *Introduction to neural machine translation with gpus*, 2015.
56. K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, arXiv (2014).
57. K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, *Learning phrase representations using rnn encoder–decoder for statistical machine translation*, Proc. 2014 EMNLP (Doha, Qatar), ACL, 2014, pp. 1724–1734.

58. K. Cho, B. van Merrienboer, Ç. Gulçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, Proc. EMNLP 2014, 2014, pp. 1724–1734.
59. F. Chollet, *Keras*, `https://github.com/fchollet/keras`, 2015.
60. J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, *Attention-based models for speech recognition*, arXiv (2015).
61. J. Chung, K. Cho, Y. Bengio, *A character-level decoder without explicit segmentation for neural machine translation*, arXiv (2016).
62. J. Chung, Ç. Gulçehre, K. Cho, Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, arXiv (2014).
63. S. Clark, B. Coecke, M. Sadrzadeh, *A compositional distributional model of meaning*, Proc. Second Symposium on Quantum Interaction (QI-2008) (2008), 133–140.
64. ———, *Mathematical foundations for a compositional distributed model of meaning*, Linguistic Analysis **36** (2011), no. 1-4, 345–384.
65. B. Coecke, M. Sadrzadeh, S. Clark, *Mathematical foundations for a compositional distributional model of meaning*, arXiv (2010).
66. R. Collobert, S. Bengio, J. Marithoz, *Torch: A modular machine learning software library*, 2002.
67. R. Collobert, J. Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, Proc. 25th international conference on Machine learning, ACM, 2008, pp. 160–167.
68. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, *Natural language processing (almost) from scratch*, Journal of Machine Learning Research **12** (2011), 2493–2537.
69. T. Cooijmans, N. Ballas, C. Laurent, A. Courville, *Recurrent batch normalization*, arXiv (2016).
70. L. Deng, Y. Liu (eds.), *Deep learning in natural language processing*, Springer, 2018.
71. L. Deng, D. Yu, *Deep learning: Methods and applications*, Foundations and Trends in Signal Processing **7** (2014), no. 3–4, 197–387.
72. L. Deng, D. Yu, *Deep learning: Methods and applications*, Foundations and Trends in Signal Process. **7** (2014), no. 3–4, 197–387.
73. J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, J. Makhoul, *Fast and robust neural network joint models for statistical machine translation*, Proc. 52nd ACL, Vol. 1: Long Papers (Baltimore, Maryland), ACL, June 2014, pp. 1370–1380.
74. N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, N. Bhamidipati, *Hierarchical neural language models for joint representation of streaming documents and their content*, WWW '15, ACM, 2015, pp. 248–255.
75. B. Dolan, C. Quirk, C. Brockett, *Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources*, COLING '04, ACL, 2004.
76. L. Dong, F. Wei, M. Zhou, K. Xu, *Question answering over freebase with multi-column convolutional neural networks*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 260–269.
77. S. A. Duffy, J. M. Henderson, R. K. Morris, *Semantic facilitation of lexical access during sentence processing*, Journal of Experimental Psychology: Learning, Memory, and Cognition **15** (1989), 791–801.

78. G. Durrett, D. Klein, *Neural CRF parsing*, arXiv (2015).
79. C. Dyer, M. Ballesteros, W. Ling, A. Matthews, N. A. Smith, *Transition-based dependency parsing with stack long short-term memory*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 334–343.
80. J. L. Elman, *Finding structure in time*, Cognitive Science **14** (1990), no. 2, 179–211.
81. K. Erk, *Representing words as regions in vector space*, CoNLL '09, ACL, 2009, pp. 57–65.
82. A. Fader, L. Zettlemoyer, O. Etzioni, *Paraphrase-driven learning for open question answering*, Proc. 51st ACL, Vol. 1: Long Papers (Sofia, Bulgaria), ACL, August 2013, pp. 1608–1618.
83. C. Fellbaum (ed.), *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, MA, 1998.
84. C. Fellbaum, *Wordnet and wordnets*, Encyclopedia of Language and Linguistics (Oxford) (K. Brown, ed.), Elsevier, 2005, pp. 665–670.
85. D. A. Ferrucci, E. W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. M. Prager, N. Schlaefer, C. A. Welty, *Building Watson: An overview of the DeepQA project*, AI Magazine **31** (2010), no. 3, 59–79.
86. O. Firat, K. Cho, Y. Bengio, *Multi-way, multilingual neural machine translation with a shared attention mechanism*, arXiv (2016).
87. D. Fried, T. Polajnar, S. Clark, *Low-rank tensors for verbs in compositional distributional semantics*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 2: Short Papers (Beijing, China), ACL, 2015, pp. 731–736.
88. K. Fukushima, *Neural network model for a mechanism of pattern recognition unaffected by shift in position — Neocognitron*, Transactions of the IECE **J62-A(10)** (1979), 658–665.
89. K. Fukushima, *Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position*, Biological Cybernetics **36** (1980), no. 4, 193–202.
90. Y. Gal, *A theoretically grounded application of dropout in recurrent neural networks*, arXiv:1512.05287 (2015).
91. Y. Gal, Z. Ghahramani, *Dropout as a Bayesian approximation: Insights and applications*, Deep Learning Workshop, ICML, 2015.
92. J. Gao, X. He, W. tau Yih, L. Deng, *Learning continuous phrase representations for translation modeling*, Proc. ACL 2014, ACL, 2014.
93. J. Gao, P. Pantel, M. Gamon, X. He, L. Deng, Y. Shen, *Modeling interestingness with deep neural networks*, EMNLP, 2014.
94. F. A. Gers, J. Schmidhuber, F. Cummins, *Learning to forget: Continual prediction with LSTM*, Neural Computation **12** (2000), no. 10, 2451–2471.
95. F. A. Gers, J. Schmidhuber, *Recurrent nets that time and count*, Neural Networks, 2000. IJCNN 2000, Proc. IEEE-INNS-ENNS International Joint Conference on, vol. 3, IEEE, 2000, pp. 189–194.
96. L. Getoor, B. Taskar, *Introduction to statistical relational learning (adaptive computation and machine learning)*, MIT Press, 2007.

97. F. Girosi, M. Jones, T. Poggio, *Regularization theory and neural networks architectures*, Neural Computation **7** (1995), no. 2, 219–269.
98. X. Glorot, Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, International conference on artificial intelligence and statistics, 2010, pp. 249–256.
99. X. Glorot, A. Bordes, Y. Bengio, *Deep sparse rectifier networks*, AISTATS, vol. 15, 2011, pp. 315–323.
100. ———, *Domain adaptation for large-scale sentiment classification: A deep learning approach*, Proc. 28th ICML, 2011, pp. 513–520.
101. Y. Goldberg, *A primer on neural network models for natural language processing*, arXiv (2015).
102. I. Goodfellow, Y. Bengio, A. Courville, *Deep learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.
103. J. T. Goodman, *A bit of progress in language modeling*, Comput. Speech Lang. **15** (2001), no. 4, 403–434.
104. A. Graves, *Generating sequences with recurrent neural networks*, arXiv (2013).
105. A. Graves, S. Fernandez, J. Schmidhuber, *Bidirectional LSTM networks for improved phoneme classification and recognition*, Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II, 2005, pp. 799–804.
106. A. Graves, J. Schmidhuber, *Framewise phoneme classification with bidirectional LSTM and other neural network architectures*, Neural Networks **18** (2005), no. 5-6, 602–610.
107. E. Grefenstette, *Towards a formal distributional semantics: Simulating logical calculi with tensors*, arXiv (2013).
108. E. Grefenstette, M. Sadrzadeh, *Experimental support for a categorical compositional distributional model of meaning*, EMNLP '11, ACL, 2011, pp. 1394–1404.
109. E. Grefenstette, M. Sadrzadeh, S. Clark, B. Coecke, S. Pulman, *Concrete sentence spaces for compositional distributional models of meaning*, Proc. 9th International Conference on Computational Semantics (IWCS11) (2011), 125–134.
110. E. Grefenstette, M. Sadrzadeh, S. Clark, B. Coecke, S. Pulman, *Concrete sentence spaces for compositional distributional models of meaning*, Computing Meaning, Springer, 2014, pp. 71–86.
111. K. Greff, R. K. Srivastava, J. KoutnΓ-k, B. R. Steunebrink, J. Schmidhuber, *LSTM: A search space odyssey*, arXiv (2015).
112. J. Gu, Z. Lu, H. Li, V. O. K. Li, *Incorporating copying mechanism in sequence-to-sequence learning*, arXiv (2016).
113. H. Guo, *Generating text with deep reinforcement learning*, arXiv (2015).
114. S. Guo, Q. Wang, B. Wang, L. Wang, L. Guo, *Semantically smooth knowledge graph embedding*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 84–94.
115. R. Gupta, C. Orasan, J. van Genabith, *Reval: A simple and effective machine translation evaluation metric based on recurrent neural networks*, Proc. 2015 EMNLP (Lisbon, Portugal), ACL, September 2015, pp. 1066–1072.

116. F. Guzmán, S. Joty, L. Marquez, P. Nakov, *Pairwise neural machine translation evaluation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, July 2015, pp. 805–814.
117. D. Hall, G. Durrett, D. Klein, *Less grammar, more features*, Proc. 52nd ACL, Vol. 1: Long Papers (Baltimore, Maryland), ACL, June 2014, pp. 228–237.
118. A. L. F. Han, D. F. Wong, L. S. Chao, *LEPOR: A robust evaluation metric for machine translation with augmented factors*, Proc. COLING 2012: Posters (Mumbai, India), The COLING 2012 Organizing Committee, December 2012, pp. 441–450.
119. S. J. Hanson, L. Y. Pratt, *Comparing biases for minimal network construction with back-propagation*, Advances in Neural Information Processing Systems (NIPS) 1 (D. S. Touretzky, ed.), San Mateo, CA: Morgan Kaufmann, 1989, pp. 177–185.
120. K. He, X. Zhang, S. Ren, J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, Proc. ICCV 2015, 2015, pp. 1026–1034.
121. ———, *Deep residual learning for image recognition*, Proc. 2016 CVPR, 2016, pp. 770–778.
122. K. M. Hermann, P. Blunsom, *Multilingual models for compositional distributed semantics*, Proc. 52nd ACL, Vol. 1: Long Papers (Baltimore, Maryland), ACL, 2014, pp. 58–68.
123. K. M. Hermann, T. Kočisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, *Teaching machines to read and comprehend*, arXiv (2015).
124. F. Hill, K. Cho, A. Korhonen, *Learning distributed representations of sentences from unlabelled data*, arXiv (2016).
125. G. E. Hinton, J. L. McClelland, *Learning representations by recirculation*, Neural Information Processing Systems (D. Z. Anderson, ed.), American Institute of Physics, 1988, pp. 358–366.
126. G. E. Hinton, S. Osindero, Y.-W. Teh, *A fast learning algorithm for deep belief nets*, Neural Computation **18** (2006), no. 7, 1527–1554.
127. G. E. Hinton, R. S. Zemel, *Autoencoders, minimum description length and helmholtz free energy*, Advances in Neural Information Processing Systems 6 (J. D. Cowan, G. Tesauro, and J. Alspector, eds.), Morgan-Kaufmann, 1994, pp. 3–10.
128. S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut fur Informatik, Lehrstuhl Prof. Brauer, Technische Universitat Munchen*, 1991, Advisor: J. Schmidhuber.
129. S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, A Field Guide to Dynamical Recurrent Neural Networks (S. C. Kremer and J. F. Kolen, eds.), IEEE Press, 2001.
130. S. Hochreiter, J. Schmidhuber, *Long Short-Term Memory*, Tech. Report FKI-207-95, Fakultat fur Informatik, Technische Universitat Munchen, 1995.
131. ———, *Long Short-Term Memory*, Neural Computation **9** (1997), no. 8, 1735–1780.
132. B. Hu, Z. Lu, H. Li, Q. Chen, *Convolutional neural network architectures for matching natural language sentences*, Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2014, pp. 2042–2050.

133. E. H. Huang, R. Socher, C. D. Manning, A. Y. Ng, *Improving word representations via global context and multiple word prototypes*, ACL '12, ACL, 2012, pp. 873–882.
134. E. H. Huang, R. Socher, C. D. Manning, A. Y. Ng, *Improving word representations via global context and multiple word prototypes*, Proc. 50th ACL: Long Papers-Volume 1, ACL, 2012, pp. 873–882.
135. P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, L. Heck, *Learning deep structured semantic models for web search using clickthrough data*, Proc. CIKM, 2013.
136. D. H. Hubel, T. N. Wiesel, *Receptive fields and functional architecture of monkey striate cortex*, Journal of Physiology (London) **195** (1968), 215–243.
137. S. Ioffe, C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, arXiv (2015).
138. O. Irsoy, C. Cardie, *Opinion mining with deep recurrent neural networks*, Proc. EMNLP, 2014, pp. 720–728.
139. M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, H. DaumΓc III, *A neural network for factoid question answering over paragraphs*, Empirical Methods in Natural Language Processing, 2014.
140. K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, *What is the best multi-stage architecture for object recognition?*, Proc. 12th ICCV, 2009, pp. 2146–2153.
141. S. Jean, K. Cho, R. Memisevic, Y. Bengio, *On using very large target vocabulary for neural machine translation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 1–10.
142. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, T. Darrell, *Caffe: Convolutional architecture for fast feature embedding*, arXiv (2014).
143. M. Joshi, M. Dredze, W. W. Cohen, C. P. RosΓc, *What's in a domain? multi-domain learning for multi-attribute data*, Proc. 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Atlanta, Georgia), ACL, 2013, pp. 685–690.
144. A. Joulin, T. Mikolov, *Inferring algorithmic patterns with stack-augmented recurrent nets*, arXiv (2015).
145. M. Kageback, O. Mogren, N. Tahmasebi, D. Dubhashi, *Extractive summarization using continuous vector space models*, Proc. 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL, 2014, pp. 31–39.
146. L. Kaiser, I. Sutskever, *Neural gpus learn algorithms*, arXiv (2015).
147. N. Kalchbrenner and P. Blunsom, *Recurrent continuous translation models*, EMNLP, vol. 3, 2013, p. 413.
148. ———, *Recurrent convolutional neural networks for discourse compositionality*, arXiv (2013).
149. N. Kalchbrenner, E. Grefenstette, P. Blunsom, *A convolutional neural network for modelling sentences*, arXiv (2014).
150. ———, *A convolutional neural network for modelling sentences*, Proc. 52nd ACL, Vol. 1: Long Papers (Baltimore, Maryland), ACL, 2014, pp. 655–665.
151. A. Karpathy, *The unreasonable effectiveness of recurrent neural networks*, 2015.

152. D. Kartsaklis, M. Sadrzadeh, S. Pulman, *A unified sentence space for categorical distributional-compositional semantics: Theory and experiments*, Proc. 24th International Conference on Computational Linguistics (COLING): Posters (Mumbai, India), 2012, pp. 549–558.
153. T. Kenter, M. de Rijke, *Short text similarity with word embeddings*, , CIKM '15, ACM, 2015, pp. 1411–1420.
154. Y. Kim, *Convolutional neural networks for sentence classification*, Proc. 2014 EMNLP (Doha, Qatar), ACL, October 2014, pp. 1746–1751.
155. Y. Kim, Y. Jernite, D. Sontag, A. M. Rush, *Character-aware neural language models*, arXiv (2015).
156. D. Kingma, J. Ba, *Adam: A method for stochastic optimization*, arXiv (2014).
157. D. P. Kingma, J. Ba, *Adam: A method for stochastic optimization*, arXiv (2014).
158. D. P. Kingma, T. Salimans, M. Welling, *Variational dropout and the local reparameterization trick*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 2575–2583.
159. S. Kiritchenko, X. Zhu, S. M. Mohammad, *Sentiment analysis of short informal texts*, Journal of Artificial Intelligence Research (2014), 723–762.
160. R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, S. Fidler, *Skip-thought vectors*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 3294–3302.
161. R. Kneser, H. Ney, *Improved backing-off for m-gram language modeling*, Proc. ICASSP-95, vol. 1, 1995, pp. 181–184.
162. P. Koehn, *Statistical machine translation*, 1st ed., Cambridge University Press, New York, NY, USA, 2010.
163. O. Kolomiyets, M.-F. Moens, *A survey on question answering technology from an information retrieval perspective*, Inf. Sci. **181** (2011), no. 24, 5412–5434.
164. A. Krogh, J. A. Hertz, *A simple weight decay can improve generalization*, Advances in Neural Information Processing Systems 4 (D. S. Lippman, J. E. Moody, and D. S. Touretzky, eds.), Morgan Kaufmann, 1992, pp. 950–957.
165. A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, R. Socher, *Ask me anything: Dynamic memory networks for natural language processing*, arXiv (2015).
166. J. Lafferty, A. McCallum, F. C. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*.
167. T. . Landauer, S. T. Dumais, *A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge*, Psychological review **104** (1997), no. 2, 211–240.
168. H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio, *An empirical evaluation of deep architectures on problems with many factors of variation*, ICML '07, ACM, 2007, pp. 473–480.
169. H. Larochelle, G. E. Hinton, *Learning to combine foveal glimpses with a third-order boltzmann machine*, Advances in Neural Information Processing Systems 23 (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), Curran Associates, Inc., 2010, pp. 1243–1251.

170. A. Lavie, K. Sagae, S. Jayaraman, The Significance of Recall in Automatic Metrics for MT Evaluation, 134–143, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 134–143.
171. Q. V. Le, N. Jaitly, G. E. Hinton, *A simple way to initialize recurrent networks of rectified linear units*, arXiv (2015).
172. Q. V. Le, T. Mikolov, *Distributed representations of sentences and documents*, arXiv (2014).
173. Y. LeCun, *Une procΓcdure d'apprentissage pour rΓcseau a seuil asymΓctrique*, Proc. Cognitiva 85, Paris (1985), 599–604.
174. Y. LeCun, *Modeles connexionnistes de l'apprentissage (connectionist learning models)*, Ph.D. thesis, UniversitΓc P. et M. Curie (Paris 6), 1987.
175. ———, *A theoretical framework for back-propagation*, Proc. 1988 Connectionist Models Summer School (CMU, Pittsburgh, Pa) (D. Touretzky, G. Hinton, and T. Sejnowski, eds.), Morgan Kaufmann, 1988, pp. 21–28.
176. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *Gradient-based learning applied to document recognition*, Intelligent Signal Processing, IEEE Press, 2001, pp. 306–351.
177. Y. LeCun, F. Fogelman-Soulie, *Modeles connexionnistes de l'apprentissage*, Intellectica, special issue apprentissage et machine (1987).
178. Y. LeCun, Y. Bengio, G. Hinton, *Human-level control through deep reinforcement learning*, Nature **521** (2015), 436–444.
179. Y. LeCun, K. Kavukcuoglu, C. Farabet, *Convolutional networks and applications in vision*, Proc. ISCAS 2010, 2010, pp. 253–256.
180. O. Levy, Y. Goldberg, I. Ramat-Gan, *Linguistic regularities in sparse and explicit word representations*, CoNLL, 2014, pp. 171–180.
181. J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, J. Gao, *Deep reinforcement learning for dialogue generation*, Proc. 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, 2016, pp. 1192–1202.
182. T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, *Continuous control with deep reinforcement learning*, arXiv (2015).
183. C. Lin, Y. He, R. Everson, S. Ruger, *Weakly supervised joint sentiment-topic detection from text*, IEEE Transactions on Knowledge and Data Engineering **24** (2012), no. 6, 1134 – 1145 (und).
184. C.-Y. Lin, F. J. Och, *Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics*, ACL '04, ACL, 2004.
185. Z. Lin, W. Wang, X. Jin, J. Liang, D. Meng, *A word vector and matrix factorization based method for opinion lexicon extraction*, WWW '15 Companion, ACM, 2015, pp. 67–68.
186. W. Ling, C. Dyer, A. W. Black, I. Trancoso, R. Fermandez, S. Amir, L. Marujo, T. Luis, *Finding function in form: Compositional character models for open vocabulary word representation*, Proc. EMNLP 2015 (Lisbon, Portugal), ACL, 2015, pp. 1520–1530.
187. S. Linnainmaa, *The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors*, Master's thesis, Univ. Helsinki, 1970.

188. B. Liu, *Sentiment analysis and opinion mining*, Synthesis Lectures on Human Language Technologies, vol. 5, Morgan & Claypool Publishers, 2012.
189. ———, *Sentiment analysis: mining opinions, sentiments, and emotions*, Cambridge University Press, 2015.
190. ———, *Sentiment analysis: Mining opinions, sentiments, and emotions*, Cambridge University Press, 2015.
191. C. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, J. Pineau, *How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation*, Proc. EMNLP 2016, 2016, pp. 2122–2132.
192. P. Liu, X. Qiu, X. Huang, *Learning context-sensitive word embeddings with neural tensor skip-gram model*, IJCAI'15, AAAI Press, 2015, pp. 1284–1290.
193. Y. Liu, Z. Liu, T.-S. Chua, M. Sun, *Topical word embeddings*, , AAAI'15, AAAI Press, 2015, pp. 2418–2424.
194. A. Lopez, *Statistical machine translation*, ACM Comput. Surv. **40** (2008), no. 3, 8:1–8:49.
195. R. Lowe, M. Noseworthy, I. V. Serban, N. Angelard-Gontier, Y. Bengio, J. Pineau, *Towards an automatic turing test: Learning to evaluate dialogue responses*, Submitted to ICLR 2017, 2017.
196. R. Lowe, N. Pow, I. Serban, J. Pineau, *The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems*, arXiv (2015).
197. Q. Luo, W. Xu, *Learning word vectors efficiently using shared representations and document representations*, AAAI'15, AAAI Press, 2015, pp. 4180–4181.
198. Q. Luo, W. Xu, J. Guo, *A study on the cbow model's overfitting and stability*, Web-KR '14, ACM, 2014, pp. 9–12.
199. M.-T. Luong, M. Kayser, C. D. Manning, *Deep neural language models for machine translation*, Proc. Conference on Natural Language Learning (CoNLL) (Beijing, China), ACL, 2015, pp. 305–309.
200. M.-T. Luong, R. Socher, C. D. Manning, *Better word representations with recursive neural networks for morphology*, CoNLL (Sofia, Bulgaria), 2013.
201. T. Luong, H. Pham, C. D. Manning, *Effective approaches to attention-based neural machine translation*, Proc. 2015 EMNLP (Lisbon, Portugal), ACL, September 2015, pp. 1412–1421.
202. T. Luong, I. Sutskever, Q. Le, O. Vinyals, W. Zaremba, *Addressing the rare word problem in neural machine translation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 11–19.
203. M. Ma, L. Huang, B. Xiang, B. Zhou, *Dependency-based convolutional neural networks for sentence embedding*, Proc. ACL 2015, Vol. 2: Short Papers, 2015, p. 174.
204. A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, *Learning word vectors for sentiment analysis*, HLT '11, ACL, 2011, pp. 142–150.
205. B. MacCartney, C. D. Manning, *An extended model of natural logic*, Proc. Eight International Conference on Computational Semantics (Tilburg, The Netherlands), ACL, January 2009, pp. 140–156.
206. D. J. MacKay, *Information theory, inference and learning algorithms*, Cambridge University Press, 2003.

207. C. D. Manning, *Computational linguistics and deep learning*, Computational Linguistics (2016).
208. C. D. Manning, P. Raghavan, H. Schutze, *Introduction to information retrieval*, Cambridge University Press, 2008.
209. M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, R. Zamparelli, *Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment*, SemEval-2014 (2014).
210. B. Marie, A. Max, *Multi-pass decoding with complex feature guidance for statistical machine translation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 2: Short Papers (Beijing, China), ACL, July 2015, pp. 554–559.
211. W. McCulloch, W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics **7** (1943), 115–133.
212. F. Meng, Z. Lu, M. Wang, H. Li, W. Jiang, Q. Liu, *Encoding source language with convolutional neural network for machine translation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 20–30.
213. T. Mikolov, *Statistical language models based on neural networks*, Ph.D. thesis, Ph. D. thesis, Brno University of Technology, 2012.
214. T. Mikolov, K. Chen, G. Corrado, J. Dean, *Efficient estimation of word representations in vector space*, arXiv (2013).
215. T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, S. Khudanpur, *Recurrent neural network based language model*, INTERSPEECH **2** (2010), 3.
216. T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, S. Khudanpur, *Extensions of recurrent neural network language model*, Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, IEEE, 2011, pp. 5528–5531.
217. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, *Distributed representations of words and phrases and their compositionality*, arXiv (2013).
218. J. Mitchell, M. Lapata, *Composition in distributional models of semantics*, Cognitive Science **34** (2010), no. 8, 1388–1429.
219. ———, *Composition in distributional models of semantics*, Cognitive Science **34** (2010), no. 8, 1388–1429.
220. A. Mnih, G. E. Hinton, *A scalable hierarchical distributed language model*, Advances in neural information processing systems, 2009, pp. 1081–1088.
221. A. Mnih, K. Kavukcuoglu, *Learning word embeddings efficiently with noise-contrastive estimation*, Advances in Neural Information Processing Systems 26 (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2013, pp. 2265–2273.
222. V. Mnih, N. Heess, A. Graves, k. kavukcuoglu, *Recurrent models of visual attention*, Advances in Neural Information Processing Systems 27 (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2014, pp. 2204–2212.
223. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, *Playing atari with deep reinforcement learning*, NIPS Deep Learning Workshop, 2013.
224. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie,

A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, *Human-level control through deep reinforcement learning*, Nature **518** (2015), no. 7540, 529–533.
225. G. Montavon, G. B. Orr, K. Muller (eds.), *Neural networks: Tricks of the trade - second edition*, Lecture Notes in Computer Science, vol. 7700, Springer, 2012.
226. L. Morgenstern, C. L. Ortiz, *The winograd schema challenge: Evaluating progress in commonsense reasoning*, AAAI'15, AAAI Press, 2015, pp. 4024–4025.
227. K. P. Murphy, *Machine learning: a probabilistic perspective*, Cambridge University Press, 2013.
228. A. Neelakantan, B. Roth, A. McCallum, *Compositional vector space models for knowledge base completion*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 156–166.
229. V. Ng, C. Cardie, *Improving machine learning approaches to coreference resolution*, ACL '02, ACL, 2002, pp. 104–111.
230. Y. Oda, G. Neubig, S. Sakti, T. Toda, S. Nakamura, *Syntax-based simultaneous translation through prediction of unseen syntactic constituents*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 198–207.
231. M. Osborne, S. Moran, R. McCreadie, A. Von Lunen, M. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, T. Jackson, F. Ciravegna, A. O'Brien, *Real-time detection, tracking, and monitoring of automatically discovered events in social media*, Proc. 52nd ACL: System Demonstrations (Baltimore, Maryland), ACL, June 2014, pp. 37–42.
232. B. Pang, L. Lee, *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*, ACL '05, ACL, 2005, pp. 115–124.
233. ———, *Opinion mining and sentiment analysis*, Foundations and trends in information retrieval **2** (2008), no. 1-2, 1–135.
234. P. Pantel, *Inducing ontological co-occurrence vectors*, ACL '05, ACL, 2005, pp. 125–132.
235. D. Paperno, N. T. Pham, M. Baroni, *A practical and linguistically-motivated approach to compositional distributional semantics*, Proc. 52nd ACL, Vol. 1: Long Papers (Baltimore, Maryland), ACL, 2014, pp. 90–99.
236. K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, *Bleu: a method for automatic evaluation of machine translation*, Proc. 40th ACL, ACL, 2002, pp. 311–318.
237. ———, *Bleu: A method for automatic evaluation of machine translation*, ACL '02, ACL, 2002, pp. 311–318.
238. D. B. Parker, *Learning-logic*, Tech. Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT, 1985.
239. R. Pascanu, Ç. Gulçehre, K. Cho, Y. Bengio, *How to construct deep recurrent neural networks*, arXiv (2013).
240. Y. Peng, S. Wang, and .-L. Lu, Marginalized Denoising Autoencoder via Graph Regularization for Domain Adaptation, 156–163, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 156–163.
241. J. Pennington, R. Socher, C. Manning, *Glove: Global vectors for word representation*, Proc. 2014 EMNLP (Doha, Qatar), ACL, 2014, pp. 1532–1543.

242. J. Pouget-Abadie, D. Bahdanau, B. van Merrienboer, K. Cho, Y. Bengio, *Overcoming the curse of sentence length for neural machine translation using automatic segmentation*, arXiv (2014).
243. L. Prechelt, Early Stopping — But When?, 53–67, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 53–67.
244. J. Preiss, M. Stevenson, *Unsupervised domain tuning to improve word sense disambiguation*, Proc. 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Atlanta, Georgia), ACL, 2013, pp. 680–684.
245. S. Prince, *Computer vision: Models, learning, and inference*, Cambridge University Press, 2012.
246. A. Ramesh, S. H. Kumar, J. Foulds, L. Getoor, *Weakly supervised models of aspect-sentiment for online course discussion forums*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 74–83.
247. R. S. Randhawa, P. Jain, G. Madan, *Topic modeling using distributed word embeddings*, arXiv (2016).
248. M. Ranzato, G. E. Hinton, Y. LeCun, *Guest editorial: Deep learning*, International Journal of Computer Vision **113** (2015), no. 1, 1–2.
249. J. Reisinger and R. J. Mooney, *Multi-prototype vector-space models of word meaning*, HLT '10, ACL, 2010, pp. 109–117.
250. X. Rong, *word2vec parameter learning explained*, arXiv (2014).
251. F. Rosenblatt, *Principles of neurodynamics*, Spartan, New York, 1962.
252. F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review **65** (1958), no. 6, 386–408.
253. H. Rubenstein, J. B. Goodenough, *Contextual correlates of synonymy*, Communications of the ACM **8** (1965), no. 10, 627–633.
254. A. A. Rusu, S. G. Colmenarejo, Ç. Gulçehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, R. Hadsell, *Policy distillation*, arXiv (2015).
255. M. Sachan, K. Dubey, E. Xing, M. Richardson, *Learning answer-entailing structures for machine comprehension*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 239–249.
256. M. Sadrzadeh, E. Grefenstette, *A compositional distributional semantics, two concrete constructions, and some experimental evaluations*, , QI'11, Springer-Verlag, 2011, pp. 35–47.
257. M. Sahlgren, *The Distributional Hypothesis*, Italian Journal of Linguistics **20** (2008), no. 1, 33–54.
258. R. Salakhutdinov, *Learning Deep Generative Models*, Annual Review of Statistics and Its Application **2** (2015), no. 1, 361–385.
259. R. Salakhutdinov, G. Hinton, *An efficient learning procedure for deep boltzmann machines*, Neural Computation **24** (2012), no. 8, 1967–2006.
260. R. Salakhutdinov, G. E. Hinton, *Deep boltzmann machines*, Proc. Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009, 2009, pp. 448–455.
261. J. Schmidhuber, *Deep learning in neural networks: An overview*, Neural Networks **61** (2015), 85–117.

262. M. Schuster, *On supervised learning from sequential data with applications for speech recognition*, Ph.D. thesis, Nara Institute of Science and Technolog, Kyoto, Japan, 1999.
263. M. Schuster, K. K. Paliwal, *Bidirectional recurrent neural networks*, IEEE Transactions on Signal Processing **45** (1997), no. 11, 2673–2681.
264. H. Schwenk, *Continuous space language models*, Comput. Speech Lang. **21** (2007), no. 3, 492–518.
265. I. V. Serban, A. G. O. II, J. Pineau, A. C. Courville, *Multi-modal variational encoder-decoders*, arXiv (2016).
266. I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, J. Pineau, *Hierarchical neural network generative models for movie dialogues*, arXiv (2015).
267. I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, Y. Bengio, *A hierarchical latent variable encoder-decoder model for generating dialogues*, Proc. 31st AAAI, 2017, pp. 3295–3301.
268. H. Setiawan, Z. Huang, J. Devlin, T. Lamar, R. Zbib, R. Schwartz, J. Makhoul, *Statistical machine translation features with multitask tensor networks*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 31–41.
269. A. Severyn, A. Moschitti, *Learning to rank short text pairs with convolutional deep neural networks*, SIGIR '15, ACM, 2015, pp. 373–382.
270. K. Shah, R. W. M. Ng, F. Bougares, L. Specia, *Investigating continuous space language models for machine translation quality estimation*, Proc. 2015 EMNLP (Lisbon, Portugal), ACL, September 2015, pp. 1073–1078.
271. L. Shang, Z. Lu, H. Li, *Neural responding machine for short-text conversation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, July 2015, pp. 1577–1586.
272. Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, *A latent semantic model with convolutional-pooling structure for information retrieval*, CIKM '14, ACM, 2014, pp. 101–110.
273. C. Silberer, M. Lapata, *Learning grounded meaning representations with autoencoders.*, ACL (1), 2014, pp. 721–732.
274. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, *Mastering the Game of Go with Deep Neural Networks and Tree Search*, Nature **529** (2016), no. 7587, 484–489.
275. M. Snover, B. Dorr, R. Schwartz, L. Micciulla, J. Makhoul, *A study of translation edit rate with targeted human annotation*, Proc. Association for Machine Translation in the Americas, 2006, pp. 223–231.
276. R. Snow, S. Prakash, D. Jurafsky, A. Y. Ng, *Learning to Merge Word Senses*, Proc. Joint Meeting of the Conference on Empirical Methods on Natural Language Processing and the Conference on Natural Language Learning, 2007, pp. 1005–1014.
277. R. Socher, J. Bauer, C. D. Manning, A. Y. Ng, *Parsing with compositional vector grammars*, Proc. ACL, 2013, pp. 455–465.

278. R. Socher, D. Chen, C. D. Manning, A. Ng, *Reasoning with neural tensor networks for knowledge base completion*, Advances in Neural Information Processing Systems (NIPS), 2013.
279. R. Socher, E. H. Huang, J. Pennin, C. D. Manning, A. Y. Ng, *Dynamic pooling and unfolding recursive autoencoders for paraphrase detection*, Advances in Neural Information Processing Systems, 2011, pp. 801–809.
280. R. Socher, A. Karpathy, Q. Le, C. Manning, A. Ng, *Grounded compositional semantics for finding and describing images with sentences*, Transactions of the Association for Computational Linguistics **2014** (2014).
281. R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, C. D. Manning, *Semi-supervised recursive autoencoders for predicting sentiment distributions*, Proc. EMNLP 2011, ACL, 2011, pp. 151–161.
282. R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, Proc. EMNLP 2013, vol. 1631, Citeseer, 2013, p. 1642.
283. Y. Song, H. Wang, X. He, *Adapting deep ranknet for personalized search*, WSDM 2014, ACM, 2014.
284. A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, J.-Y. Nie, *A hierarchical recurrent encoder-decoder for generative context-aware query suggestion*, CIKM '15, ACM, 2015, pp. 553–562.
285. R. Soricut, F. Och, *Unsupervised morphology induction using word embeddings*, Proc. 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Denver, Colorado), ACL, 2015, pp. 1627–1637.
286. B. Speelpenning, *Compiling fast partial derivatives of functions given by algorithms*, Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana-Champaign, 1980.
287. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research **15** (2014), no. 1, 1929–1958.
288. R. K. Srivastava, K. Greff, J. Schmidhuber, *Training very deep networks*, NIPS'15, MIT Press, 2015, pp. 2377–2385.
289. P. Stenetorp, *Transition-based dependency parsing using recursive neural networks*, Deep Learning Workshop at NIPS 2013, 2013.
290. J. Su, D. Xiong, Y. Liu, X. Han, H. Lin, J. Yao, M. Zhang, *A context-aware topic model for statistical machine translation*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 229–238.
291. P.-H. Su, M. Gasic, N. Mrkši, L. M. Rojas Barahona, S. Ultes, D. Vandyke, T.-H. Wen, S. Young, *On-line active reward learning for policy optimisation in spoken dialogue systems*, Proc. 54th ACL, Vol. 1: Long Papers (Berlin, Germany), ACL, August 2016, pp. 2431–2441.
292. S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, *Weakly supervised memory networks*, arXiv (2015).
293. F. Sun, J. Guo, Y. Lan, J. Xu, X. Cheng, *Learning word representations by jointly modeling syntagmatic and paradigmatic relations*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 136–145.

294. I. Sutskever, G. E. Hinton, *Deep, narrow sigmoid belief networks are universal approximators*, Neural Computation **20** (2008), no. 11, 2629–2636.
295. I. Sutskever, J. Martens, G. Hinton, *Generating text with recurrent neural networks*, ICML '11, ACM, 2011, pp. 1017–1024.
296. I. Sutskever, O. Vinyals, Q. V. Le, *Sequence to sequence learning with neural networks*, arXiv (2014).
297. Y. Tagami, H. Kobayashi, S. Ono, A. Tajima, *Modeling user activities on the web using paragraph vector*, WWW '15 Companion, ACM, 2015, pp. 125–126.
298. K. S. Tai, R. Socher, C. D. Manning, *Improved semantic representations from tree-structured long short-term memory networks*, Proc. 53rd ACL and 7th IJCNLP, Vol. 1, 2015, pp. 1556–1566.
299. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, *Deepface: Closing the gap to human-level performance in face verification*, CVPR '14, IEEE Computer Society, 2014, pp. 1701–1708.
300. D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, B. Qin, *Learning sentiment-specific word embedding for twitter sentiment classification.*, ACL (1), 2014, pp. 1555–1565.
301. W. T. Yih, X. He, C. Meek, *Semantic parsing for single-relation question answering*, Proc. ACL, ACL, 2014.
302. J. Tiedemann, *News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces*, Recent Advances in Natural Language Processing (vol V) (Amsterdam/Philadelphia) (N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, eds.), John Benjamins, Amsterdam/Philadelphia, 2009, pp. 237–248.
303. I. Titov, J. Henderson, *A latent variable model for generative dependency parsing*, IWPT '07, ACL, 2007, pp. 144–155.
304. E. F. Tjong Kim Sang, S. Buchholz, *Introduction to the conll-2000 shared task: Chunking*, ConLL '00, ACL, 2000, pp. 127–132.
305. B. Y. Tong Zhang, *Boosting with early stopping: Convergence and consistency*, The Annals of Statistics **33** (2005), no. 4, 1538–1579.
306. K. Toutanova, D. Klein, C. D. Manning, Y. Singer, *Feature-rich part-of-speech tagging with a cyclic dependency network*, NAACL '03, ACL, 2003, pp. 173–180.
307. Y. Tsuboi, H. Ouchi, *Neural dialog models: A survey*, Available from `http://2boy.org/~yuta/publications/neural-dialog-models-survey-20150906.pdf`., 2015.
308. J. Turian, L. Ratinov, Y. Bengio, *Word representations: A simple and general method for semi-supervised learning*, ACL '10, ACL, 2010, pp. 384–394.
309. P. D. Turney, P. Pantel, et al., *From frequency to meaning: Vector space models of semantics*, Journal of artificial intelligence research **37** (2010), no. 1, 141–188.
310. E. Tutubalina, S. I. Nikolenko, *Constructing aspect-based sentiment lexicons with topic modeling*, Proc. 5th International Conference on Analysis of Images, Social Networks, and Texts (AIST 2016), 2016, p. to appear.
311. B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, Y. Bengio, *Blocks and fuel: Frameworks for deep learning*, arXiv (2015).
312. D. Venugopal, C. Chen, V. Gogate, V. Ng, *Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features*, Proc. 2014 EMNLP (Doha, Qatar), ACL, October 2014, pp. 831–843.

313. P. Vincent, *A connection between score matching and denoising autoencoders*, Neural Computation **23** (2011), no. 7, 1661–1674.
314. P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, *Extracting and composing robust features with denoising autoencoders*, ICML '08, ACM, 2008, pp. 1096–1103.
315. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion*, Journal of Machine Learning Research **11** (2010), 3371–3408.
316. O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, G. E. Hinton, *Grammar as a foreign language*, arXiv (2014).
317. O. Vinyals, Q. V. Le, *A neural conversational model*, ICML Deep Learning Workshop, arXiv:1506.05869, 2015.
318. V. Viswanathan, N. F. Rajani, Y. Bentor, R. Mooney, *Stacked ensembles of information extractors for knowledge-base population*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 177–187.
319. X. Wang, Y. Liu, C. Sun, B. Wang, X. Wang, *Predicting polarities of tweets by composing word embeddings with long short-term memory*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 1343–1353.
320. D. Weiss, C. Alberti, M. Collins, S. Petrov, *Structured training for neural network transition-based parsing*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 323–333.
321. J. Weizenbaum, *Eliza — a computer program for the study of natural language communication between man and machine*, Communications of the ACM **9** (1966), no. 1, 36–45.
322. T. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, S. J. Young, *Conditional generation and snapshot learning in neural dialogue systems*, Proc. 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, 2016, pp. 2153–2162.
323. P. J. Werbos, *Applications of advances in nonlinear sensitivity analysis*, Proc. 10th IFIP Conference, 31.8 - 4.9, NYC, 1981, pp. 762–770.
324. ———, *Backpropagation through time: what it does and how to do it*, Proc. IEEE **78** (1990), no. 10, 1550–1560.
325. P. J. Werbos, *Backwards differentiation in AD and neural nets: Past links and new opportunities*, Automatic Differentiation: Applications, Theory, and Implementations, Springer, 2006, pp. 15–34.
326. J. Weston, A. Bordes, S. Chopra, T. Mikolov, *Towards ai-complete question answering: A set of prerequisite toy tasks*, arXiv (2015).
327. J. Weston, S. Chopra, A. Bordes, *Memory networks*, arXiv (2014).
328. L. White, R. Togneri, W. Liu, and M. Bennamoun, *How well sentence embeddings capture meaning*, ADCS '15, ACM, 2015, pp. 9:1–9:8.
329. R. J. Williams, D. Zipser, *Gradient-based learning algorithms for recurrent networks and their computational complexity*, Backpropagation (Hillsdale, NJ, USA) (Y. Chauvin and D. E. Rumelhart, eds.), L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995, pp. 433–486.
330. Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil,

W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, *Google's neural machine translation system: Bridging the gap between human and machine translation*, arXiv (2016).
331. Z. Wu, C. L. Giles, *Sense-aware semantic analysis: A multi-prototype word representation model using wikipedia*, AAAI'15, AAAI Press, 2015, pp. 2188–2194.
332. S. Wubben, A. van den Bosch, E. Krahmer, *Paraphrase generation as monolingual translation: Data and evaluation*, INLG '10, ACL, 2010, pp. 203–207.
333. C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, T.-Y. Liu, *Rc-net: A general framework for incorporating knowledge into word representations*, , CIKM '14, ACM, 2014, pp. 1219–1228.
334. K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, Y. Bengio, *Show, attend and tell: Neural image caption generation with visual attention*, arXiv (2015).
335. R. Xu, D. Wunsch, *Clustering*, Wiley-IEEE Press, 2008.
336. X. Xue, J. Jeon, W. B. Croft, *Retrieval models for question and answer archives*, SIGIR '08, ACM, 2008, pp. 475–482.
337. M. Yang, T. Cui, W. Tu, *Ordering-sensitive and semantic-aware topic modeling*, arXiv (2015).
338. Y. Yang, J. Eisenstein, *Unsupervised multi-domain adaptation with feature embeddings*, Proc. 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Denver, Colorado), ACL, 2015, pp. 672–682.
339. Z. Yang, X. He, J. Gao, L. Deng, A. J. Smola, *Stacked attention networks for image question answering*, arXiv (2015).
340. K. Yao, G. Zweig, B. Peng, *Attention with intention for a neural network conversation model*, arXiv (2015).
341. X. Yao, J. Berant, B. Van Durme, *Freebase qa: Information extraction or semantic parsing?*, Proc. ACL 2014 Workshop on Semantic Parsing (Baltimore, MD), ACL, June 2014, pp. 82–86.
342. Y. Yao, L. Rosasco, A. Caponnetto, *On early stopping in gradient descent learning*, Constructive Approximation **26** (2007), no. 2, 289–315.
343. W.-t. Yih, M.-W. Chang, C. Meek, A. Pastusiak, *Question answering using enhanced lexical semantic models*, Proc. 51st ACL, Vol. 1: Long Papers (Sofia, Bulgaria), ACL, August 2013, pp. 1744–1753.
344. W.-t. Yih, G. Zweig, J. C. Platt, *Polarity inducing latent semantic analysis*, EMNLP-CoNLL '12, ACL, 2012, pp. 1212–1222.
345. W. Yin, H. Schutze, *Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 63–73.
346. W. Yin, H. Schutze, B. Xiang, B. Zhou, *ABCNN: attention-based convolutional neural network for modeling sentence pairs*, arXiv (2015).
347. J. Yohan, O. A. H., *Aspect and sentiment unification model for online review analysis*, WSDM '11, ACM, 2011, pp. 815–824.
348. A. M. Z. Yang, A. Kotov, S. Lu, *Parametric and non-parametric user-aware sentiment topic models*, Proc. 38th ACM SIGIR, 2015.

349. W. Zaremba, I. Sutskever, *Reinforcement learning neural Turing machines*, arXiv (2015).
350. W. Zaremba, I. Sutskever, and O. Vinyals, *Recurrent neural network regularization*, arXiv (2014).
351. M. D. Zeiler, *ADADELTA: an adaptive learning rate method*, arXiv (2012).
352. L. S. Zettlemoyer, M. Collins, *Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars*, arXiv (2012).
353. X. Zhang, Y. LeCun, *Text understanding from scratch*, arXiv (2015).
354. X. Zhang, J. Zhao, Y. LeCun, *Character-level convolutional networks for text classification*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 649–657.
355. G. Zhou, T. He, J. Zhao, P. Hu, *Learning continuous word embedding with metadata for question retrieval in community question answering*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 250–259.
356. H. Zhou, Y. Zhang, S. Huang, J. Chen, *A neural probabilistic structured-prediction model for transition-based dependency parsing*, Proc. 53rd ACL and the 7th IJCNLP, Vol. 1: Long Papers (Beijing, China), ACL, 2015, pp. 1213–1222.

Поступило 2 октября 2020 г.

Saarland University,
66123 Saarbrücken, Germany
E-mail: arkhangelskaya.ekaterina@gmail.com

St. Petersburg State University
7/9 Universitetskaya nab.,
St. Petersburg, 199034 Russia;
St. Petersburg Department of
Steklov Institute of Mathematics,
St. Petersburg, Russia
E-mail: s.nikolenko@spbu.ru, sergey@logic.pdmi.ras.ru