




Review

Survey of Neural Text Representation Models

Karlo Babić * , Sanda Martinčić-Ipšić  and Ana Meštrović 

Center for Artificial Intelligence and Cybersecurity and Department of Informatics, University of Rijeka, 51000 Rijeka, Croatia; smarti@inf.uniri.hr (S.M.-I.); amestrovic@inf.uniri.hr (A.M.)

* Correspondence: karlo.babic@uniri.hr; Tel.: +385-51-584-718

Received: 1 October 2020; Accepted: 28 October 2020; Published: 30 October 2020



Abstract: In natural language processing, text needs to be transformed into a machine-readable representation before any processing. The quality of further natural language processing tasks greatly depends on the quality of those representations. In this survey, we systematize and analyze 50 neural models from the last decade. The models described are grouped by the architecture of neural networks as shallow, recurrent, recursive, convolutional, and attention models. Furthermore, we categorize these models by representation level, input level, model type, and model supervision. We focus on task-independent representation models, discuss their advantages and drawbacks, and subsequently identify the promising directions for future neural text representation models. We describe the evaluation datasets and tasks used in the papers that introduced the models and compare the models based on relevant evaluations. The quality of a representation model can be evaluated as its capability to generalize to multiple unrelated tasks. Benchmark standardization is visible amongst recent models and the number of different tasks models are evaluated on is increasing.

Keywords: deep learning; embedding; neural language model; neural networks; NLP; text representation

1. Introduction

Natural language is a valuable and rich source of information for many applications. Still, natural language is discrete and sparse, and as such a challenging source of data. For text to be usable as input data, it first has to be transformed into a suitable representation, which is usually a vector of the text's features, hence a vector of numbers. The vector representations of text can be constructed in many different ways, this paper provides a survey of the neural models that generate continuous and dense text representations.

The basic non-neural text representation methods, which preserve a very limited amount of information, are one-hot encoding and TFIDF (term frequency inverse document frequency) [1]. One-hot encoding creates a Boolean vector of values for each word. The length of one-hot vectors is equal to the vocabulary size, and all the elements are coded with "0" except the word in focus, which is coded with "1". Each dimension in that vector space corresponds to one word. Vectors from that space are used as inputs for machine learning. To represent larger units of text (multi-word units like phrases, sentences, or documents), one-hot vectors have a "1" for each of the words in the unit in focus (bag-of-words representation). TFIDF vectors extend Boolean values from one-hot vectors with frequencies, normalized by the inverse document frequencies.

This survey is focused on neural models that learn text representations as vectors in a continuous n-dimensional space [2]. The learned n-dimensional vector space can be structured to capture semantics, syntax, knowledge, or other language properties. The process of representing text elements as continuous and dense vectors is called embedding (e.g., word embedding). The representations can be learned for any

unit of text like subwords, words, phrases, sentences, and documents. The current challenge in neural text representation learning is to construct task-independent representations, hence representations that generalize well to multiple unrelated tasks [3]. Therefore, in this work, the tasks the models are applied to (downstream tasks) are not put in the front, and the main focus is on representation properties and qualities. Still, researchers evaluate models on a variety of tasks, which we cross-reference in Table A2.

Due to the nature of neural networks, when a model has learned to perform well at a task, a byproduct is a dense low dimensional representation that is formed in the hidden layers of the neural network. Compared to the traditional representation methods, neural models learn which text features to incorporate in representations. Moreover, the learned representations have a smaller dimensionality which makes them more efficient on downstream tasks. As is mentioned in the book [4]: “generally, a good representation is one that makes a downstream learning task easier.” The neural representation learning can be treated as a pre-training step or language modeling step for one or multiple downstream tasks (e.g., machine translation [5], sentiment analysis [6], text generation [3], question answering [7], text summarization [8], topic detection [9], or any other natural language processing task. An important property of the pre-trained representations is faster portability between tasks, or a form of transfer learning, where representations learned for one task can be utilized as the pre-training step for a new task. The dense representation vectors in a continuous space can be readily used for simpler tasks like similarity assessment or as a substitute for the features used for machine learning [10].

This research aims to systematically compare neural models that learn text representations, identify their advantages and drawbacks, and subsequently identify the promising directions for the construction of neural text representation models. More specifically, we aim to identify neural text representation research directions that are feasible for research teams with limited computational resources. To do so, we analyze 50 models published in research papers and at NLP (natural language processing) conferences in the last decade. Papers for the survey are selected following three major criteria. First, we consider papers with a reasonable number of citations. Second, we prefer models with novel ideas or with improvements in any aspect over the previous models. Third, we opt for task-independent (general) representation models. Task-independent representation models perform well on a variety of downstream tasks. Still, to ensure the integrity of this survey, we include some task-dependent models that have largely influenced the field. The representation models included are mainly limited to English texts, except for models trained on machine translation tasks.

Compared to other works reviewing deep learning models for NLP (e.g., [11]), this survey focuses on text representation learning and, as such, reviews models from that perspective. Among similar surveys that analyzed neural network models for text representation are a survey of neural language models [12], a survey of vector representation of meaning [13], a survey of cross-lingual word embedding models [14], and a comparability study of pre-trained language models [15]. The authors in [12] elaborate upon different architectures of the classic neural network language models and their improvements, but it lacks a deep survey and is focused on a narrower set of language models. The survey of vector representation of meaning in [13] presents a comprehensive survey of the wide range of techniques for sense representation; however, it is focused on sense representation models exclusively. The survey of cross-lingual word embedding models in [14] provides a comprehensive typology of models; however, it is framed with cross-lingual word embedding models. The work about language models in [15] proposes starting points for defining reporting standards which are applied when it comes to comparing pre-trained language models (the model architecture, number of parameters, hyperparameter tuning method and tuning time, experimental time, computational resources, training time, lexical resource information and availability, the benchmark performance on un-tuned and tuned single model); however, its survey is focused on several large pre-trained language models.

Our survey goes a step further, categorizing and summarizing the most prominent models created in the last decade. Additionally, we did not limit the scope to a specific architecture, task, or the representation level, instead, we opted for multi-categorization indicating the subtle differences among models, emphasizing their advantages and drawbacks.

To systematically compare the representation models, we used five criteria: (1) representation level (the level of linguistic units for which representations are being learned); (2) input level (the granularity of data on input); (3) model type (the general strategy for representation learning); (4) model architecture (neural network architecture), and (5) model supervision (an indication of the need for labeled training data). Additionally, we review standard datasets and tasks used for the evaluation of neural representation models. The generality of the neural representation model is measured by its capability to perform well on a variety of tasks (e.g., word/sentence+ similarity, word analogy, sentiment classification, paraphrase detection, machine translation, natural language inference, subjectivity, question answering and classification, text retrieval, coreference resolution, reading comprehension, and summarization).

The rest of the paper is organized as follows. Section 2 covers datasets used for the evaluation in the covered papers. Section 3 presents the evaluation tasks used in the described models. Section 4 describes the categorizations that we use to analyze representation models (representation level, input level, model type, model architecture, model supervision). Sections 5 through to 9 cover representation models. Section 10 is a discussion, and Section 11 concludes the paper.

2. Evaluation Datasets

In this section, we list the datasets used for the evaluation of the representation models we cited the results for. From the extensive list of datasets, we have omitted a few that lack the comparative and systematic evaluation and are of limited interest for this study. Later, in Section 3, we connect the datasets with respective evaluation tasks and Table A1 cross-references models with datasets for which we referenced results.

- WordSim-353 [16] contains pairs of words with their similarity and relatedness scores.
- SICK (Sentences Involving Compositional Knowledge) [17] consists of English sentence pairs that were annotated for relatedness and entailment.
- SNLI (Stanford Natural Language Inference) [18] contains sentence pairs manually labeled for entailment classification.
- MultiNLI (Multi-Genre Natural Language Inference) [19] dataset consists of sentence pairs annotated with textual entailment information.
- MRPC (Microsoft Research Paraphrase Corpus) [20] has pairs of sentences along with human labels indicating if the pairs share paraphrase/semantic equivalence relationship.
- STS (Semantic Text Similarity) [21] consists of pairs of sentences with labels annotating their semantic similarity.
- SSWR (Semantic-Syntactic Word Relationship) [22] contains semantic and syntactic word relations (e.g., “Einstein: scientist”, “Mozart: violinist”, “think: thinks”, “say: says”).
- MSR (Measuring Semantic Relatedness) [23] is similar to the SSWR dataset.
- SST (Stanford Sentiment Treebank) [24] has parsing trees with a sentiment classification label for each node.
- IMDB (Internet Movie Database) [25] contains reviews of movies and their sentiment polarity.
- MR (Movie Reviews) [26] is similar to the IMDB dataset, hence contains opinions on movies.
- CR (Customer Reviews) [27] consists of customer reviews about products and their rating.
- SemEval’17 [28] is a dataset from the International Workshop on Semantic Evaluation. It consists of posts from Twitter with sentiment labels.

- TREC (Text REtrieval Conference) [29] is a dataset for question classification with questions divided into semantic categories.
- SQuAD (Stanford Question Answering Dataset) [30] consists of questions posed by humans on a set of Wikipedia articles, where the answer to every question is a segment of text.
- MPQA (Multi-Perspective Question Answering) [31] contains news articles annotated for opinions, beliefs, emotions, sentiments, speculations, and other private states.
- NQ (NaturalQuestions-Open) [32] contains Google queries and their answers.
- WQ (WebQuestions) [33] is a collection of questions found through Google Suggest API, which are manually answered.
- WSJ (Wall Street Journal) consists of Wall Street Journal stories from Penn Treebank [34].
- AG News [35] contains news categorized into four classes (business, sci/tech, world, and entertainment). This version of the dataset contains only titles and disregards the rest of the articles.
- GLUE (General Language Understanding Evaluation) [36] is a collection of datasets for natural language understanding systems.
- RACE (ReAding Comprehension dataset from Examinations) [37] is a dataset for the evaluation of methods in the reading comprehension task, consisting of English exams for Chinese students.
- SUBJ (SUBjectivity classification) [38] contains labeled subjective and objective phrases.
- WMT (Workshop on statistical Machine Translation) is a dataset used for machine translation tasks in WMT workshops.

3. Evaluation Tasks

The quality of the learned representation space has to be evaluated. The evaluations are performed indirectly on different downstream tasks, and each task can provide one perspective about the representation space quality.

The more tasks a model is evaluated on, the more accurate the evaluation of its generality is. We notice a trend in the number of tasks that the neural text representation models are evaluated on (Table A2), throughout recent years, we can see a growth in the number of tasks each of the models uses for its evaluation. In addition, this is an indication of shifting toward learning more general representation models. In the rest of this section, we cross-reference tasks with datasets used for evaluation and provide a brief description of the tasks. The word/sentence+ similarity task is widely used but is too simple to be able to test better representations, as even shallow text representations perform reasonably well on such tasks. In this task, a model has to quantify the similarity of two words or sentences. The datasets used to evaluate models on the similarity task are WordSim-353, SICK, MRPC, and STS. The most used evaluation measures are Spearman and sometimes Pearson correlation.

The word analogy task can be used as well and can tell us the representation space structure quality. This task evaluates the ability of a model to recognize analogy relations. The datasets used for evaluation on this task are SSWR and MSR. The prevalent evaluation measure for the word analogy task is accuracy.

The sentiment classification task can provide more information about the sentence representation quality. It is used to evaluate the ability of a model to recognize sentiments in text. The datasets used for this task are SST, IMDB, MR, CR, MPQA, and SemEval'17. The most used evaluation measure for the sentiment classification task is accuracy.

The paraphrase detection task is similar to the word/sentence+ similarity task. Models trained on this task learn to recognize if two texts are semantically equivalent. The dataset used for evaluation on this task is MRPC. The most used evaluation measure for the paraphrase detection task is accuracy.

The machine translation task is able to inform us about the sentence encoding quality. In this task, models learn to translate text from the source language into text in the target language. WMT is frequently used as a dataset for machine translation evaluation. The usual evaluation measure for the translation task is the BLEU score, utilized in the majority of machine translation downstream tasks.

The natural language inference task is able of high-quality evaluation as well. It evaluates the ability of a model to recognize textual entailment between two sentences. The datasets used to evaluate models on this task are SICK, SNLI, and MultiNLI. The most used evaluation measure for the inference task is accuracy.

The subjectivity task detects if a text is subjective or objective. The dataset used for evaluation is SUBJ. The most used evaluation measure for the subjectivity task is accuracy.

Representations that perform well at question answering tasks preserve the knowledge and understanding of text semantics. In this task, models learn to answer questions. Question classification is a similar and yet simpler task. The dataset used for evaluation in the question answering task is SQuAD, and the dataset used for question classification is TREC. The most used evaluation measure for the question answering task is the F1 score.

The text retrieval task in its core functions similarly to the word/sentence+ similarity task. Text retrieval models have to retrieve the most relevant text to the query text from the collection of texts. The dataset used for evaluation is TREC. The most used evaluation measure for the text retrieval task is accuracy.

In addition to the frequently engaged downstream tasks above, coreference resolution, reading comprehension, and summarization are occasionally utilized. Coreference resolution is a task of linking related linguistic units referring to the same entity in a text (i.e., linking pronouns to nouns). The reading comprehension task is almost equivalent to the question answering task. The dataset used for evaluation is RACE. Summarization is the task of creating short versions of a text while preserving the most important meaning of the longer text. Additionally, several studies engage perplexity as the standard measure for the evaluation of language models [39].

Some of the models are evaluated on tasks that test their ability to recognize polysemy or anaphora. Models that are able to recognize polysemy are the model by Huang et al. [40], ELMo [41], the model by Akbik et al. [42], and CDWE [43]. Models that are able to work with anaphora (evaluated on the Winograd schema challenge) are GPT [44] and XLNet [45]. Some of the papers that evaluated the models used other evaluation tasks as well. In Table A2, those tasks are categorized as “Classification tasks” and “Other”.

4. Model Categorization

To compare and contrast the models, we used five categorization criteria: (1) the representation level shows the level of linguistic units for which representations are learned; (2) the input level shows the granularity of data a model receives upon input; (3) the model type elaborates the strategy for representation learning; (4) the model architecture covers the main neural network architecture of the model; (5) the model supervision shows how much labeling is needed in training data. Each of these categorizations is described in the rest of this section. For recursive models, we used one more categorization, the parsing tree source (which we describe in Section 7).

4.1. Representation Level

Text can be represented on multiple levels. Models can learn representations for subwords, words, phrases, sentences, paragraphs, or documents, in general for any unit of text.

Subword elements include bytes, characters, and character n-grams. Downstream tasks in languages with rich morphology perform better with subword representations, and out-of-vocabulary words are easily represented [46]. Please note, subword representations only serve as the basis for the derivation of word-level representations.

Word representations are easier to implement, as generated representations are ready for use. Hence, they can be used as word models or combined into representations of larger units of text.

Sentence+ category includes representations for any larger units of text, like phrases, sentences, paragraphs, and documents. Whereas word representations can be combined to form a sentence+ representation, neural network models that are specialized for learning sentence+ representations usually produce better representations.

4.2. Input Level

Whereas the representation level describes granularity of the representations that are learned in a model, the input level describes granularity of text upon input. The granularity of text units upon input can be classified as subword-level (bytes, characters, character n-grams), or word-level. Models with subword-level input are flexible and can easily work with rare or never seen words, whereas models with word-level input are easier to implement.

Earlier models mainly used words as input (e.g., Word2Vec [22,47]), but some (especially convolutional, and later recurrent) models used characters (e.g., CharCNN [48] and ELMo [41], and later FastText [49] used character n-grams). Recent models still use these forms of input, but when working with subword input, subword tokenization techniques improved the effectiveness of the input data.

The subword tokenization algorithm that inspired newer methods (most noticeably SentencePiece and WordPiece) is an algorithm for data compression by Gage [50] called BPE (byte pair encoding), which is used by many attention models (e.g., Transformer [51]). BPE replaces the most common pairs of consecutive bytes of data with bytes that do not occur within that data. SP (SentencePiece) [52] and WP (WordPiece) [53] are used for attention models as well (e.g., XLNet [45] and BERT [54]).

4.3. Model Type

The next criterion for differentiation of representation models considers three high-level strategies for neural network learning: autoencoding, autoregressive, and classification.

Autoencoding models learn efficient data representations in an unsupervised manner. In the strict definition of autoencoders, they learn how to compress input data into a representation, which is then decompressed back into data that should be similar to the input data. In this survey, we generalize the autoencoding definition to encompass models that learn by predicting missing input data or neighboring tokens of the input tokens.

Autoregressive models are time series models. They predict the following tokens by using previously predicted tokens as additional input. While learning to predict, autoregressive models learn text representations as well.

Classification models predict which of a set of categories a new observation belongs to. In order to correctly classify textual data, the model has to learn a representation space in which it is easy to separate textual inputs that belong to different categories.

4.4. Model Architecture

Neural network models that learn text representation most frequently use shallow, recurrent, recursive, convolutional, attentive, or a combination of mentioned architectures [55]. Here we briefly introduce the architectures, whereas details are elaborated in the subsequent sections. Each model is placed in a section that corresponds with the model's general architecture. Within sections, models are ordered by shared base architecture or general principle and by publishing date. Figure A2 in the Appendix A shows the relative frequency of each model architecture through the years.

Shallow neural network architectures in our categorization include feed-forward neural networks that have one hidden layer. Such models are capable of learning only a very shallow representation of text.

RNN (recurrent neural network) architectures are appropriate for sequential data. Text input can be treated sequentially by reading input token by token while updating network recurrent states.

RecNN (recursive neural network) architectures read texts recursively following the structure of parsing trees. Parsing trees can be provided with the text upon input or latently generated by the neural network. Because of that, recursive models are capable of creating high-quality sentence representations.

CNN (convolutional neural network) architectures are well-suited for learning local patterns in text. Pooling layers learn to detect important tokens or features for a specific task it is trained on [55].

Attentive neural network architectures in this categorization include models that use attention mechanisms. An attention mechanism learns dependence between input and output, or within the input elements (self-attention). For instance it can learn which parts of the input are relevant to the task at hand or can capture long-range dependencies in text.

4.5. Model Supervision

The supervision of a model dictates how it is learned and what kind of training data it requires. The models in this survey are unsupervised, supervised, semi-supervised, and un/supervised.

Unsupervised learning is the easiest one to implement from the perspective of data preparation. In the case of text representation, models need only raw text without labels. Occasionally, self-supervised models are referred to as unsupervised and vice versa.

Supervised learning requires labeled data. Data labeling is different for each task: text translation needs pairs of texts in the source and target languages, text summarization requires full texts and their summaries, classification task needs class labels, etc.

Semi-supervised learning uses both unsupervised and supervised learning. Usually, text representation models initially learn representations in an unsupervised way from a dataset containing only the raw text with no labels. Subsequently, unsupervisedly learned representations are fine-tuned through supervised learning on a specific task.

Un/supervised learning is the category we used in this survey for models that have been tested on both unsupervised and supervised tasks.

5. Shallow Models

Shallow models use shallow architectures (usually one hidden layer), which can learn simple representations (Word2Vec architecture is shown in Figure 1). In this category, we include two non-neural models that are frequently used for the same purpose as the rest of the models and are shown to be comparable [56,57]. As a result of their simplicity, shallow models are fast to train and generalize well. Learned shallow representations can be used as input for deeper models that can learn better representations. While performing very well on simple tasks (e.g., word similarity), shallow models do not perform well on complex tasks (tasks that require a deeper understanding, e.g., question answering, summarization). To generate a phrase, sentence, or document representation, shallow representations are combined, resulting in a bag-of-words representation. The most prominent shallow models are systematized in Table 1 according to the input and representation level. The majority of shallow models for text representation learn word-level representations. Usually, they are autoencoding models learned by predicting the missing information upon input.

Language models, and more specifically shallow models described in this section, follow the distributional hypothesis, which states that the words with similar meanings occur in similar contexts [58,59]. To learn representations that reflect the distributional hypothesis, shallow models are trained to predict neighboring words (i.e., context) around the target word and vice versa, to predict the target word from neighboring words. Shallow models mostly contain only one hidden layer in which text

representations are learned. The representation being learned dictates how the input vector transforms into the output vector. The input vector is a one-hot representation of the target word or the neighboring words. In contrast, the output vector is a probability distribution that describes which words are most probably the target or the neighboring words. In the learned vector space, representations are closer together for similar words. Relationships between the words are preserved (i.e., king and man are positioned in the same way as queen and woman, and the positions of countries and their capitals preserve meaningful information).

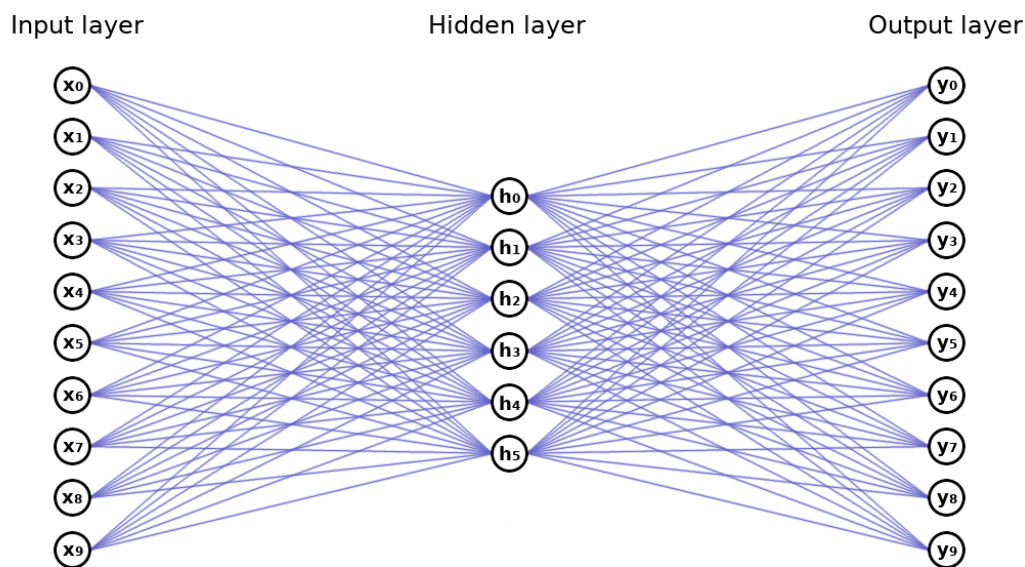


Figure 1. Word2Vec architecture. In addition to an input layer and an output layer, shallow architectures have a small number of hidden layers (one in this case).

Table 1. The categorization of shallow models by input and representation level. All listed models are unsupervised.

Model	Input	Representation
Huang et al. [40]	word	word
Word2Vec [22,47]	word	word
Deps [60]	word	word
GloVe [61]	word	word
Doc2Vec [62]	word	sentence+
FastText [49]	n-grams	word

The unsupervised model presented by Huang et al. [40] uses both local and global context via a joint training objective. It learns multiple representations per word, which is effective for homonyms and polysemous words. The meaning of a word can be ambiguous given only local context. By using a global context, the meaning of a word can be disambiguated more precisely. On the task of measuring similarity and relatedness, it achieves a Spearman correlation of 0.713 on the WordSim-353 dataset, which outperforms C&W (convolutional model described in Section 8) by 0.16.

Word2Vec [22,47] popularized shallow word representation using neural networks. Word2Vec has two separate autoencoding models: continuous bag-of-words (CBOW) and skip-gram. Both learn word representations through unsupervised learning. The CBOW model scans over the text with a context window and learns to predict the target word. The context window contains n preceding and n succeeding

words around the target word. The skip-gram model conversely predicts the words in the context from the target word. Word2Vec neural network has only one hidden layer, and word representations are extracted from that layer. The accuracies for the skip-gram model on the SSWR dataset for semantics and syntax are 50% and 55.9%, respectively, outperforming the model by Huang et al. by 36.7% and 44.3%. The accuracy on the MSR dataset for the skip-gram model is 56%.

Whereas Word2Vec uses local neighboring words as context, its extension Deps [60] uses neighboring words in dependency parse trees to learn word representations. Deps generalizes skip-gram to include arbitrary contexts and uses dependency-based contexts derived from parse trees.

GloVe [61] directly captures global corpus statistics through unsupervised learning. It is inspired by neural models but is not a neural model itself. GloVe combines global matrix factorization and local context window methods through a bilinear regression model. By doing so, it trains by using the co-occurrence matrix and learns word representations in a way that it can predict co-occurrence probability ratios. An example was given in [61]. Let $i = ice$ and $j = steam$, if $k = solid$, we expect the ratio P_{ik}/P_{jk} to be large (P_{xy} is probability of words x and y occurring together). If $k = gas$, the ratio should be small. For words that are related to both ice and $steam$, or to neither, the ratio should be closer to 1. This is used instead of raw probabilities because ratios distinguish relevant words from irrelevant words better. It is shown, that mathematically GloVe is similar to Word2Vec [56,57]. GloVe slightly outperforms Word2Vec on multiple tasks. On the SSWR dataset, GloVe has 75% total accuracy (81.9% semantics and 69.3% syntax), which is an improvement of 9.4% over Word2Vec. GloVe is evaluated on five word similarity tasks (including WordSim-353 where it achieves a 0.759 Spearman correlation), and in each evaluation, GloVe outperforms Word2Vec.

The shallow models above can learn representations for words. Doc2Vec [62] is an extension of Word2Vec that can learn representations for documents (or parts of texts). While predicting context words of a target word, Doc2Vec receives the target word and the document ID upon input. Through learning, as it learns useful relations between documents and words, it learns not only representations for words but documents as well. The accuracy on the SST dataset is 87.8% for binary classification, and 48.7% for fine-grained classification, outperforming the recursive model by Socher et al. [24] by 2.4% and 3%, respectively. Accuracy on the IMDB dataset is 92.58%.

All models described above use word-level input, but subword-level input can be beneficial. FastText [49] is an unsupervised non-neural model inspired by neural models. Similarly to Word2Vec, it has two separate models, CBOW and skip-gram. It learns representations for character n-grams, and each word is represented as a bag-of-character n-grams. Previous models were limited to assigning a distinct vector to each word. Representations on the subword level are shown to perform better for morphologically rich languages and rare words. On the WordSim-353 dataset, FastText has a Spearman correlation of 0.71 (similar to Word2Vec's 0.72). On the SSWR dataset, it achieves 77.8% semantic and 74.9% syntactic accuracy, performing on the semantic task similarly as Word2Vec and outperforming it on the syntactic task by 4.8%.

6. Recurrent Models

Models like Word2Vec are insensitive to word order, they only capture the relations between words—if a word is in the context of another word. The ordering of words in a text is meaningful, since different orderings of words can denote different meanings. RNNs (recurrent neural networks) process input as a sequence and learn ordered representations from a text, hence they are well suited for learning representations of longer linguistic units like phrases, sentences, paragraphs, or documents. While tokens from the input sequence are processed, the history of all the previous tokens is preserved as a state in the neurons (Figure 2). The most influential RNN models are listed in Table 2 according to their input

level, representation level, and model supervision. The majority of RNNs for text representation are autoregressive models. Output tokens are generated by taking previous output tokens as an additional input (together with the representation of the encoded sequence, currently being learned).

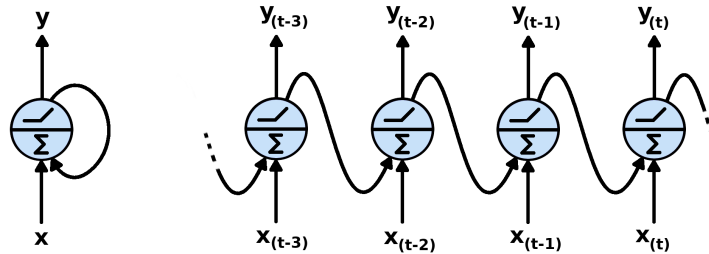


Figure 2. Recurrent architecture and the unfolding in time. Recurrent nodes have connections which lead to the next node, and connections which loop back to the same node. The unfolding in time shows the same node in three consecutive iterations.

In shallow feedforward models, the hidden layer and ultimately the output is affected only by the current input. In recurrent models, the hidden layer (or layers) and the output are affected not only by the current input but also by the previous inputs. That is achieved by using recurrence, where the recurrent layers propagate information not only to the next layer but back to their input as well. To train neural networks, gradients are calculated and used to update parameters in a direction that depends on the loss function. With longer texts, RNNs have a vanishing (and exploding) gradient problem as the gradient is calculated through a long chain of recurrence. That problem is reduced by LSTM (long short term memory unit) and its variants like GRU (gated recurrent unit), which have gates that learn what information is important and, as such, has to be propagated more strongly. LSTM’s architecture is shown in Figure 3. It has a cell state (horizontal line on the top), which is used for memory. Gates below control information flow by selectively forgetting old and memorizing new information relevant for the current task.

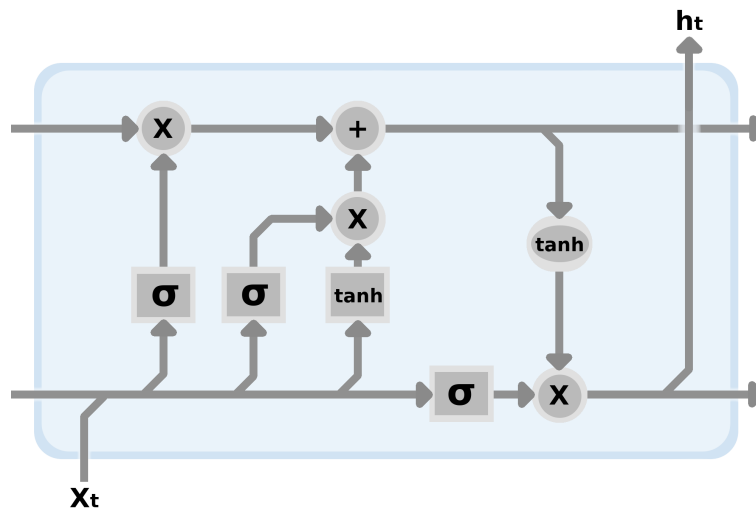


Figure 3. The LSTM (long short term memory) cell. Variables x_t and h_t are input and output, respectively, at time t . Squares with “ σ ” or “ \tanh ” represent layers, whereas ovals with “ X ”, “ $+$ ”, or “ \tanh ” represent pointwise operations.

Table 2. The categorization of recurrent models by input level, representation level, and supervision. BPE stands for byte pair encoding.

Model	Input	Repres.	Supervision
Cho et al. [63]	word	sentence+	supervised
Seq2Seq [5]	word	sentence+	supervised
Skip-Thoughts [64]	word	sentence+	unsupervised
RCNN [65]	word	sentence+	supervised
FastSent [66]	word	sentence+	unsupervised
CoVe [67]	word	word	supervised
Akbik et al. [42]	character	word	unsupervised
ELMo [41]	character	word	unsupervised
Subramanian et al. [68]	word	sentence+	semi-supervised
LASER [69]	BPE	sentence+	supervised

A supervised encoder–decoder model introduced by Cho et al. [63] uses two RNNs. One RNN encodes a sequence of symbols into a representation vector, and the other decodes the representation into another sequence of symbols. The sequence of symbols can be a phrase, part of a sentence, a sentence, or a longer linguistic unit. The model is trained on a machine translation task. The model achieves a 34.54 BLEU score on the English–French translation task from the WMT’14 dataset, which is an improvement of 1.24 points over the baseline method (Moses).

With the addition of LSTM units, the Seq2Seq [5] model improves performance on longer sequences over basic RNN models. It is learned in a supervised way on a machine translation task. Seq2Seq uses LSTM units in recurrent layers. This model can produce representations for phrases and sentences. Seq2Seq scores 34.81 BLEU on the English–French translation task from the WMT’14 dataset, outperforming the recurrent model by Bahdanau et al. [70] by 6.36 points.

Whereas most recurrent models learn text representations by sequentially predicting words (autoregressive models), Skip-Thoughts [64] is an unsupervised encoder–decoder model that learns to reconstruct the surrounding sentences of an encoded sentence, and in that aspect, it is similar to other autoencoding models. It uses encoder with GRU activations and an RNN decoder with a conditional GRU. Skip-Thoughts learns representations for sentences and performs just as well as the LSTM approach. On the SICK semantic relatedness subtask, Skip-Thoughts performs with the Pearson correlation 0.8655 and the Spearman correlation 0.7995, which is similar to Tree-LSTM (recursive model described in Section 7). Accuracy on the MRPC dataset is 75.8% and the F1 score is 83.

RCNN [65] (recurrent convolutional neural network) is a supervised bidirectional RNN (BiRNN) with a pooling layer after the BiRNN layer. Sentence representations are learned through the BiRNN, which scans over texts upon input in both directions (forward and backward), whereas normal RNNs scan texts only in the forward direction. The pooling layer learns to select the most important words for a text classification task. Fine-grained classification accuracy on the SST dataset is 47.21%, similar to the accuracy achieved by Doc2Vec.

A simple variant of the Skip-Thoughts model FastSent [66] is an unsupervised model with GRUs. FastSent receives a bag-of-words representation of a sentence upon input and predicts adjacent sentences, which are also represented as bag-of-words representations. FastSent outperforms Skip-Thoughts on unsupervised tasks: the Spearman correlation for the SICK dataset is 0.61, which is an improvement of 0.04 over Skip-Thoughts, whereas for STS tasks the Spearman correlation is 0.63, which is an improvement of 0.36 over Skip-Thoughts. FastSent under-performs Skip-Thoughts on six supervised sentence classification tasks (on MRPC, MR, CR, SUBJ, MPQA, and TREC dataset). Overall, FastSent has an average accuracy of 77.92% on the supervised tasks, under-performing Skip-Thoughts by 5.83%.

CoVe [67] (Context Vectors) is a supervised bidirectional LSTM (BiLSTM) encoder with an attentive LSTM decoder trained for machine translation. Word representations are extracted from the BiLSTM encoder. The attentive LSTM decoder first uses the LSTM to produce a hidden state, after which the attention mechanism computes the relevance between each encoding time-step and the current state. CoVe is evaluated on seven classification tasks (on SST-2, SST-5, IMDB, TREC-6, TREC-50, SNLI, and SQuAD dataset). Overall, its average accuracy is 84.8%, which outperforms GloVe by 1.8% and Skip-Thoughts by 2.2%.

Models mentioned previously produced one vector for each subword, word, or sentence, which cannot account for polysemy (multiple meanings of the same word). ELMo [41] (Embeddings from Language Models) is an unsupervised model that uses BiLSTM and can learn polysemy. ELMo's word representations are learned functions of the internal states of the BiLSTM. In different contexts, the same word has different representation vectors, which can account for different word meanings. ELMo achieves 54.7% fine-grained classification accuracy on the SST dataset, outperforming CoVe by 1.8%.

The model introduced by Akbik et al. [42] is an unsupervised character-level language model. The word representations are extracted from BiLSTM hidden states. As word representations are contextualized by their surrounding text, the same word can have different meanings depending on its context, which can model polysemy. Word representations are concatenations of hidden states of the forward pass through a sentence until the last character in the word, and the backward pass through a sentence until the first character in the word. This model has 93.09% accuracy on the CoNLL'03 dataset, outperforming ELMo by 0.87%.

A semi-supervised model introduced by Subramanian et al. [68] uses bidirectional GRU (BiGRU) for sentence representation multi-task learning. The tasks that the model is trained on are skip-thought, machine translation, constituency parsing, and natural language inference. The authors demonstrate that multi-task learning leads to consistent improvements over previous single-task methods (some of which are FastSent, Skip-Thoughts, and CNN-LSTM, which is described in Section 8).

LASER [69] (Language-Agnostic SEntence Representations) learns joint multilingual sentence representations for 93 languages through a supervised machine translation task. The learned representations are general with respect to the input language and the task. It uses the BiLSTM encoder to generate a sentence representation, and the LSTM decoder to generate a sentence in a target language. The encoder receives no information about what the input language is, due to which it learns language-independent representations. The model is trained by translating all 91 languages to both English and Spanish languages. On the XNLI (cross-lingual natural language inference) dataset [71], with 70.19% average accuracy on 15 languages, this model archives similar results as BERT (attention model described in Section 9).

7. Recursive Models

Recurrent models learn representations with ordered information; however, recursive neural networks go one step further and learn deeply structured information like trees. Recursive neural networks' (RecNN) process inputs in a recursive fashion through a tree structure (example given in Figure 4). Each node in a tree for a word, phrase, sentence, or a larger unit of text is associated with a respective representation. The most influential RecNN models are shown in Table 3 with their model supervision, and tree source (given, learned, or latent). Most of the RecNNs for text representation are autoencoding or classification models. The representation for each node can be learned either by an autoencoding method similar to the shallow models or through a classification task.

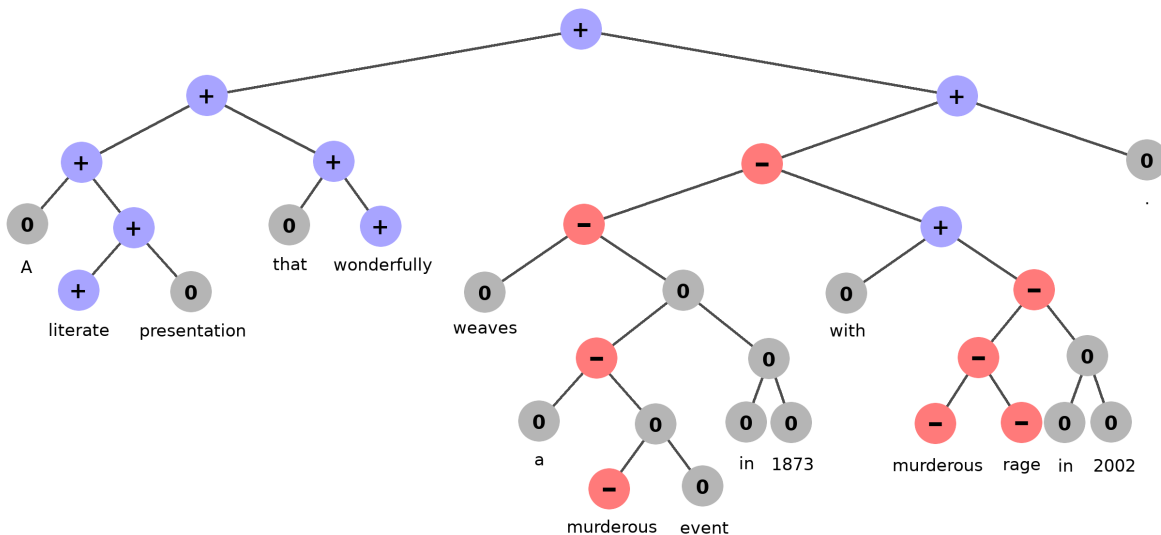


Figure 4. A parsing tree of a recursive neural network predicting word sentiment classes. The leaf nodes are input tokens, all the other nodes are representations of the combination of the child nodes. The root node is representation of the entire input text.

Recursive neural networks are generalized recurrent neural networks. Whereas recurrent neural networks read input in a linear chain, recursive neural networks read input in any hierarchical structure. Representations are merged recursively throughout the tree structure, where merging combines two or more representations from the lower level into one higher-level representation. Hence, learning of the recursive neural representations enables combining representations of more granular linguistic units into larger linguistic units (e.g., from characters to sentences, or from words to documents). The merging process is repeated recursively until the root node is reached. The root node represents the whole input sequence (Figure 4) and leaf nodes are input tokens. The tree structure can be given upon input, learned from labeled texts (texts paired with their parse trees), or implicitly generated by a neural network with no supervision (latent trees).

Table 3. The categorization of recursive models by supervision and parsing tree source. All the listed models work with word-level input and learn sentence+ representations. Un/supervised supervision represents both unsupervised and supervised learning.

Model	Supervision	Tree
RAE [72]	un/supervised	latent
MV-RNN [73]	un/supervised	given
AdaSent [74]	supervised	latent
Tree-LSTM [75]	supervised	given
RL-SPINN [76]	reinforcement	latent
ST-Gumbel [77]	unsupervised	latent
DIORA [78]	unsupervised	latent

RAE [72] (Recursive AutoEncoders) introduced an architecture based on recursive autoencoders for sentence-level prediction of sentiment label polarity. An autoencoder is a type of neural network that learns representations of input data in an unsupervised manner. The encoder part of the network encodes the input into a dense representation, while the decoder part of the network decodes that representation into

output (that should be as close as possible to the input). While RAE can learn text representation through supervised learning of sentiment distribution, it can learn text representation through unsupervised learning as well. An important RAE feature is the ability to learn latent parsing trees, meaning that they are not learned or given upon input, but generated by concatenating neighboring pairs of words or phrases and combining the ones with the lowest reconstruction error (in the autoencoder) into parent nodes. On sentiment classification, RAE achieves an accuracy of 77.7% on the MR dataset and 86.4% on the MPQA opinion dataset, which is a slight improvement over Tree-CRF [79].

MV-RNN [73] (Matrix-Vector recursive neural network) learns vector and matrix representations for every node in the tree (phrases and sentences). The vector captures the meaning of the node, whereas the matrix captures how it changes the meaning of neighboring words or phrases. Supervised training for each of the tasks is done by adding a softmax classifier on top of each parent node. The model receives parse trees from a parser. It achieves 79% accuracy on the MR dataset, which is an 1.3% improvement over RAE.

Whereas other RecNNs process inputs as trees, AdaSent [74] process inputs in a hierarchical fashion (pyramid structure) which does not require parse trees upon input. The pyramid structure forms a hierarchy of representations from words to sentences. The model is trained on supervised classification tasks. Classification uses the hierarchy of representations as input. The hierarchy of representations is first summed on each level of the hierarchy, and then a gating network calculates a weight for each of the levels. AdaSent has 83.1% accuracy on the MR dataset, outperforming RAE by 5.4%. Its accuracy on the MPQA opinion dataset is 93.3%, outperforming RAE by 6.9%. It outperforms MV-RNN as well on the MR dataset by 6.9% with a 93.3% accuracy.

Tree-LSTM [75] introduced a generalization of LSTM to tree-structured network topologies. The model is trained on a supervised sentiment task and it requires parse trees upon input. LSTM architectures can process only sequential information, Tree-LSTM extends LSTM in a way which can process structured information. Tree-LSTM performs better than an RecNN with basic RNNs because of the same reasons LSTM outperforms RNN. It has 51% fine-grained classification accuracy on the SST dataset, outperforming RAE, and MV-RNN by 7.8% and 6.6%, respectively.

RL-SPINN [76] (Reinforcement Learning SPINN) uses reinforcement learning to learn sentence representations. RL-SPINN's architecture is based on the SPINN model (Stack-augmented Parser-Interpreter Neural Network) [80]. Parse trees are latent, meaning that the tree structures are not learned from labeled data or given upon input, but are unsupervisedly optimized for the downstream task. RL-SPINN performs similarly as Tree-LSTM and DCNN (the convolutional model described in a later section) on the SST dataset with an 86.5% accuracy for binary classification. It has a 0.359 mean squared error on the SICK dataset, under-performing Tree-LSTM by 0.067.

ST-Gumbel [77] (Straight-Through Gumbel-softmax estimator) is a Tree-LSTM modification as well. ST-Gumbel uses latent trees that are computed with a composition query vector that measures the validity of a composition. Text representation is learned through unsupervised training. On the SST dataset, it achieves 53.7% accuracy for fine-grained classification, and 90.7% accuracy for binary classification, outperforming Tree-LSTM by 2.7% in both fine-grained and binary classification.

DIORA [78] (Deep Inside-Outside Recursive Autoencoders) is an unsupervised model that learns to predict each word in a sentence while being conditioned on the rest of the sentence. It considers all possible trees over the input sentence. The CYK (Cocke-Younger-Kasami) parsing algorithm extracts the highest-scoring parse tree (latent trees). The architecture consists of recursive autoencoders. The training is done through inside-outside passes, where the inside representations of each node are encoded by using the children of the nodes, and outside representations of each node are encoded using only the context of the node subtree. On WSJ dataset, DIORA has a 55.7 F1 score, which is an increase of 32.9 with respect to ST-Gumbel, and an increase of 42.5 with respect to RL-SPINN.

8. Convolutional Models

Recurrent and recursive neural networks are a good fit for textual modality (as they are appropriate for sequential data), whereas convolutional neural networks (CNN) are originally used for 2-D data, and as such had to be modified to fit textual modality. CNNs proved successful with visual data (images and videos); however, they can also be used for text representation learning (the architecture is shown in Figure 5). CNNs by their nature learn to abstract input data through multiple convolutional levels and detect specific patterns on each level (e.g., textures or borders of objects in images, syllables, or word n-grams in text). The most representative CNN models used for learning text representations are shown in Table 4 with their input level, representation level, and model supervision. CNNs for text representation were in the beginning mostly classification models, but later CNN models were combined with other architectures, typically with autoencoding or autoregressive models.

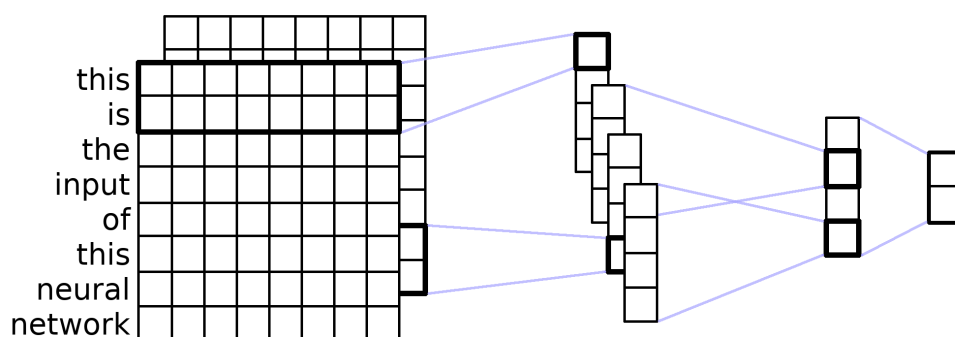


Figure 5. Convolutional architecture. A convolution has multiple filters, and each filter has a kernel (a matrix of weights) that is being trained. The kernel slides over the values from the previous layer, producing values that are sent to the next layer. Each filter learns to recognize a different pattern.

Convolutional models for images learn to recognize patterns from the lowest level (e.g., edges, corners) to the highest level (e.g., cat, dog, house). Similarly, convolutional models for text learn patterns like syllables, syntax, and phrases. The notable difference between convolutions for images and text is the dimensionality of the convolutions. Convolutions for images are two-dimensional, and convolutions for text are mostly one-dimensional. In Figure 5, input is connected to the convolution layer. The convolution layer in this example has four filters. Each filter has a kernel (a matrix of weights) trained to detect patterns that are of importance to the task at hand. Kernel slides over the values from the previous layer (in this example, input values) and outputs an activation that corresponds to how much that region of image or text fits the pattern. Next, the convolution layer is connected to the fully connected layer. Whereas recurrent and recursive models can easily read inputs of undefined length, convolutional models need pooling layers to process inputs of different sizes. The pooling layer lowers the previous layer's dimensionality by forwarding (pooling) the maximum or average of each region.

The model introduced by C&W [81] (Collobert and Weston) uses a single convolutional neural network architecture that, given a sentence upon input, outputs part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words, and the likelihood that the sentence makes sense. All of these tasks are trained jointly, resulting in an implicitly trained language model. While the language model is learned unsupervisedly, downstream tasks are learned supervisedly. During learning, all the tasks share weights, resulting in better language representation. Note that individual word representations can be extracted from the trained network.

Table 4. The categorization of convolutional models by input level, representation level, and supervision. Un/supervised supervision represents both unsupervised and supervised learning.

Model	Input	Repres.	Supervision
C&W [81]	word	word	semi-supervised
DCNN [82]	word	sentence+	supervised
CharCNN [48]	character	word	unsupervised
ByteNet [83]	character	sentence+	un/supervised
CNN-LSTM [84]	word	sentence+	unsupervised
CDWE [43]	word	word	supervised

DCNN [82] (dynamic convolutional neural network) is a convolutional model supervised for each downstream task separately. The architecture consists of multiple one-dimensional convolutional layers and dynamic k-max pooling layers, which induce a latent tree over the input sentence. The last layer, which is fully connected, contains the representation of the input sentence. This model achieves an 48.5% and 86.8% accuracy for fine-grained and binary classification on the SST dataset, outperforming the baseline method (Naive Bayes) by 7.5% and 5%, respectively. On the SST dataset, its performance is similar to the recursive model RL-SPINN described in Section 7.

CNNs are a good fit for character-level input, which is why CharCNN [48] (character-level convolutional neural network) is an unsupervised model that relies only on character-level inputs to learn word representations. In CharCNN architecture, single-layer character-level CNN reads the input, and LSTM generates the output. Each of the filters in the CNN detects a single character n-gram. The perplexity of this model on the English Penn Treebank test set [34] is 78.9, which is similar to the perplexity of the recurrent model by Zaremba et al. [85] that contains a double number of parameters.

Similarly to CharCNN, ByteNet [83] relies only on character-level inputs. It is unsupervised for the language modeling task and supervised for the machine translation task. ByteNet implements two mechanisms: the first mechanism allows the preservation of the temporal resolution of the sequences, and the second mechanism allows the network to process source and target sequences of different lengths. The model runs in a time that is linear to the length of the sequence. On the WMT'14 and WMT'15 dataset, it achieves a 23.75 and 26.26 BLEU score, respectively, outperforming the recurrent model by Chung et al. [86] by 2.42 and 2.81 points, respectively.

CNN-LSTM [84] is an unsupervised encoder–decoder architecture with a CNN encoder and LSTM decoder. It has two modes of training: autoencoder and future predictor. The autoencoder reconstructs the input sentence, whereas the future predictor predicts the rest of the sentence. The average accuracy for five-sentence classification tasks (on MR, CR, SUBJ, MPQA, and TREC dataset) is 87.1%, which is an improvement of 8% over FastSent.

By using deconvolution (transposed convolution) CDWE [43] (Convolution–Deconvolution Word Embedding) can learn multiple vectors for each word. CDWE is supervised for each downstream task. It generates multiple prototypes in the deconvolution layer for each word, which can model polysemy. After that, according to the context, a proper prototype for a word is selected. The average accuracy for three sentence classification tasks (on TREC, AG News, and MR dataset) is 90.9%, which is an improvement of 13% over CharCNN.

9. Attention Models

The architectures from the previous sections (shallow, recurrent, recursive, and convolutional) are either very general or are originally developed for a different modality of input data. The neural attention mechanism was created to capture long-range dependencies and was inspired by how humans read and understand longer texts [87]. Long-range dependencies exceed the dependencies captured within

the limited boundaries of context windows as used in shallow models. RNNs are better at capturing long-range dependencies but are not as good as attention models since they are focused equally on every word in the input. Conversely, neural networks with attention can focus on parts of a text that are more important for a current task, and thus perform better with long texts (an example is shown in Figure 6). The most influential attention models are shown in Table 5 with their input level and model supervision. As attention mechanisms can be implemented in a wide range of architectures, the models using them can be autoencoding, autoregressive, or classification models. Models that greatly gain upon the attention mechanism (e.g., Transformer, which is described later) are mostly autoencoding or autoregressive models.

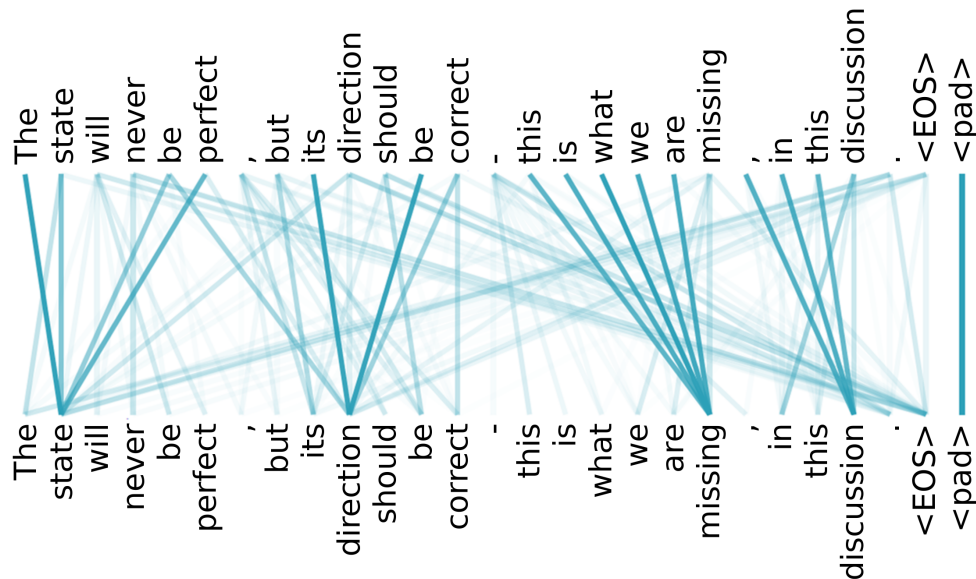


Figure 6. A visualization of a learned self-attention head on a sentence. The visualization shows learned relations between the words this self-attention head has learned. Each head learns a different kind of relations between the words.

Attention mechanism learns which parts of the input vector are important for each part of the output vector (e.g., while translating a sentence from one language to another, each output word depends more on some parts of the input than the other parts, so it makes sense to pay more attention to those parts when deciding what the next word in the output is). Self-attention is similar, and it learns dependency as well, but between words in the input rather than between the input and output (e.g., when predicting the next word of the sentence “I am eating a green”, it makes sense to pay more attention to words “eating” and “green” to predict the next word “apple”). The attention mechanism has multiple heads, which are similar to filters in convolutions. Each head learns to pay attention in different ways. One head can learn to attend to words that help decide the tense of the next word, whereas another head can learn to attend to entity names.

HAN [88] (Hierarchical Attention Network) is a supervised hierarchical attention network for a document classification task. HAN architecture is based on BiGRUs and attention mechanisms on the word and sentence level. First HAN learns sentence representations and then aggregates those representations into a document representation. The attention mechanism enables it to attend to different words when building sentence representations, and to different sentences when building document representations. On the IMDB dataset, HAN achieves 49.4% accuracy, outperforming Doc2Vec by 15.3%.

Table 5. The categorization of attention models by input level and supervision. All listed models learn sentence+ representations. BPE and WP stand for byte pair encoding and WordPiece, respectively. Un/supervised supervision represents both unsupervised and supervised learning.

Model	Input	Supervision
HAN [88]	word	supervised
Kim et al. [89]	word	unsupervised
Lin et al. [90]	word	supervised
Transformer [51]	BPE	semi-supervised
GPT [44]	BPE	semi-supervised
DiSAN [91]	word	supervised
Bi-BloSAN [92]	word	supervised
ReSAN [93]	word	supervised
BERT [54]	WP	unsupervised
Liu et al. [94]	word	supervised
GPT-2 [3]	BPE	unsupervised
XLM [95]	BPE	un/supervised
Star-Transformer [96]	word	supervised
Transformer-XL [97]	char. or word	unsupervised
MASS [98]	BPE	unsupervised
SBERT [99]	word	un/supervised
XLNet [45]	SP	unsupervised
ALBERT [100]	WP	unsupervised
SpanBERT [101]	WP, word, span	unsupervised
REALM [7]	WP	semi-supervised
ELECTRA [102]	WP	unsupervised

Authors Kim et al. [89] introduce structured attention networks (a generalization of the basic attention procedure), which can attend to partial segments or subtrees while learning representations. This approach does not consider non-projective dependency structures and its inside–outside algorithm is difficult to parallelize. To overcome the parallelization problem, Liu et al. [94] implicitly consider non-projective dependency trees and make each step of the learning process differentiable. Evaluated on the SNLI dataset, this model achieves 86.5% accuracy, which outperforms the recursive model SPINN [80] by 3.3%.

The supervised model introduced by Lin et al. [90] instead of a vector to represent text uses a 2-D matrix. Each row in that matrix representation is attending to a different part of the sentence. The model has two parts, the first part is a BiLSTM, and the second part is a self-attention mechanism. Each LSTM hidden state provides only a short-term context information around each word, whereas self-attention captures longer-range dependencies by summing the weighted LSTM hidden states. Evaluated on the SNLI dataset, this model achieves 84.4% accuracy slightly under-performing the model by Kim et al.

Most of the sequence translation models use recurrent or convolutional neural networks with an encoder and a decoder. Some also connect the encoder and decoder through an attention mechanism. Transformer [51] is a sequence translation model that is used by many models created subsequently. Transformer is a neural network architecture based exclusively on attention mechanisms, dispensing recurrent or convolutional layers. Self-attention mechanisms in Transformer show the ability to correctly resolve anaphora. The goal of some of the convolutional neural network models is to reduce sequential computation by relating signals between two positions in a sequence. Hence, in CNN models the number of computations grows linearly or logarithmically with the distance between those positions. In Transformer architecture, the number of computations is constant, which makes it easier to learn dependencies between distant positions. On the WMT’14 dataset (English–German translation task), Transformer achieves a 28.4 BLEU score, outperforming ByteNet by 4.65 points.

Similarly to Transformer, DiSAN [91] (Directional Self-Attention Network) and ReSAN [93] (Reinforced Self-Attention Network) are based only on attention mechanisms without any RNN or CNN structure. DiSAN is composed of a directional self-attention, followed by a multi-dimensional attention that creates a vector representation. It achieves 85.62% accuracy on the SNLI dataset. It has a 90.8% average accuracy on four-sentence classification datasets (on CR, MPQA, SUBJ, and TREC dataset), outperforming Skip-Thoughts by 2.2%. ReSAN integrates both soft and hard attention into one model, outperforming DiSAN on the SNLI dataset by 0.7%. On the SICK dataset it achieves a Spearman correlation of 0.8163, close to the DiSAN's correlation of 0.8139. Star-Transformer [96] is a lightweight alternative to Transformer which reduces model complexity by sparsification. Sparsification replaces the fully connected structure with a star-shaped structure, which reduces the number of connections from n^2 to $2n$. On the SNLI dataset, it performs with 86% accuracy, outperforming Transformer by 3.8%. Whereas Transformer takes 49.31 ms on test time, Star-Transformer takes 10.94 ms.

BERT [54] (Bidirectional Encoder Representations from Transformers) is a deep bidirectional Transformer. Deep bidirectional means that it is conditioned on every word in the left and right contexts at the same time. It does so by masking some percentage of the input tokens at random and then predicts those masked tokens. BERT is an unsupervised model that learns sentence representations. It achieves an 82.1% average accuracy on a subset of GLUE datasets, which is a 7% improvement over GPT (a model mentioned below). ALBERT [100] (A Lite BERT) optimizes BERT by lowering memory consumption and increasing the training speed. Evaluated on five datasets (on SquAD1.1, SQuAD2.0, MultiNLI, SST-2, and RACE dataset), ALBERT performs with an 88.7% average accuracy, outperforming BERT by 3.5%. SpanBERT [101] extends BERT by masking random spans (sections of text) whereas BERT is masking random tokens, and by training the representations to predict the entire content of the masked span. It has an 82.8% average accuracy on a subset of GLUE datasets, which is an improvement of 2.4% over BERT.

GPT [44] (Generative pre-trained Transformer) is a semi-supervised model based on Transformer. Its training is unsupervised for pre-training of the language model and supervised for fine-tuning to a downstream task. To avoid interventions into the architecture for each of the downstream tasks, they convert structured inputs into an ordered sequence that GPT can process. Largely following GPT architecture, GPT-2 [3] learns byte sequence representations through unsupervised training. The main task of this model is language modeling. GPT-2 performs well with anaphora. It is the state-of-the-art at the time of the publishing of the GPT-2 paper, outperforming other models in seven out of eight tested language modeling datasets.

Authors in XLM [95] (cross-lingual Language Model) introduced two methods to learn cross-lingual language models based on Transformer. One method is supervised and uses monolingual data, the other method is unsupervised and uses parallel texts in each of the languages. XLM has a 0.69 Pearson correlation on the SemEval'17 dataset, outperforming the model by Conneau et al. [103] by 0.04.

Standard Transformer learns short dependencies, and context becomes fragmented because of the segmentation of input contexts while training. Transformer-XL [97] is based on Transformer architecture with an added segment-level recurrence mechanism. Whereas the basic Transformer can learn dependencies of length only equal to the segment length, Transformer-XL can learn long-range dependencies by using its recurrence mechanism. It can learn dependencies that are 80% longer than dependencies in RNNs and 450% longer than dependencies in basic Transformers. Transformer-XL also solves the problem of context fragmentation, which appears in Transformer because of its segmentation. Similarly to Transformer-XL, Bi-BloSAN [92] (Bi-directional Block Self-Attention Network) captures long-range dependency. It does so with a segment-level self-attention mechanism. When evaluated on datasets CR, MPQA, SUBJ, TREC, and SST, Bi-BloSAN performs similarly to DiSAN.

MASS [98] (MAsked Sequence to Sequence pre-training) is a semi-supervised model (unsupervised pre-training and supervised fine-tuning to a specific task) based on Transformer. The encoder receives a

sentence with a randomly masked fragment, while the decoder predicts that masked fragment. MASS is a generalization of GPT and BERT. It has a hyperparameter that defines the length of the masked fragment, which when set to 1 makes MASS equivalent to BERT, and when set to the number of tokens in a sentence makes MASS equivalent to GPT. For the machine translation task evaluated on NewsTest'14 and NewsTest'16 from the WMT dataset, MASS scores a 34 average BLEU score, which is an improvement of 1.85 points over XLM.

A modification of BERT, SBERT [99] (Sentence-BERT) is a supervised model trained on pairs of sentences. SBERT uses Siamese and triplet network structures. The Siamese structure consists of two BERT networks with tied weights, which are then fed into one layer that calculates the similarity between the inputs of the two BERTs. The Triplet structure receives upon input an anchor sentence, a positive sentence, and a negative sentence. The triplet network is then trained in a way to make the distance between the anchor sentence and the positive sentence smaller while making the distance between the anchor sentence and the negative sentence bigger. BERT is slow on large-scale tasks like semantic similarity comparison, information retrieval, and clustering. It takes BERT 65 hours to find the most similar sentence pair in a collection of 10,000 sentences, whereas SBERT takes 5 seconds to compute 10,000 sentence representations and 0.01 seconds to compute cosine similarity.

Generally, autoencoding models (e.g., BERT) perform better than autoregressive models (e.g., Transformer-XL), but BERT is not optimal because it neglects the dependency between the masked positions and suffers from a pre-train/finetune discrepancy [45]. Autoencoding models perform better because of their ability to model bidirectional contexts. XLNet [45] is an unsupervised model that combines BERT and Transformer-XL. XLNet integrates ideas from Transformer-XL while learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order. As a result of its autoregressive formulation, XLNet overcomes the limitations of BERT. XLNet is able to model anaphora. It outperforms BERT on 20 evaluated datasets [45].

Models similar to BERT (that use masked language modeling training methods) corrupt the input by replacing some tokens with a mask and then train a model to reconstruct those tokens. ELECTRA [102] (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) has a more efficient training method. It replaces selected tokens with alternatives produced by a generator network and then trains a discriminative model to predict for each token if it was replaced by another plausible token or not. The architecture has two components: the generator, which is typically a small masked language model, and the discriminator—the ELECTRA model. The generator and the discriminator are trained jointly: the generator of the output text with tokens replaced with plausible alternatives, and the discriminator to detect which tokens are replaced. The ELECTRA model is similar to a generative adversarial network (GAN), but the generator is not trained to generate text that would deceive the discriminator, instead it produces tokens that have the highest likelihood. On the GLUE datasets, ELECTRA performs similarly as XLNet with an average accuracy of 89.5%, outperforming BERT by 5.5%.

REALM [7] (Retrieval-Augmented Language Model) is a semi-supervised masked language model. Language models typically capture world knowledge implicitly, REALM does it in a more modular and interpretable way by incorporating knowledge retriever in the learning procedure. Two main components of REALM are: the neural knowledge retriever, which finds a document containing the answer to the question, and the knowledge-augmented encoder, which outputs the answer to the question with the help of the retrieved document. The knowledge retriever uses a BERT-style architecture, and the knowledge-augmented encoder uses a vanilla Transformer architecture. After fine-tuning on the Open-domain Question Answering task, REALM outperforms BERT on the NQ and WQ datasets by 13.9% and 23%, achieving 40.4% and 40.7% accuracy, respectively.

10. Discussion

In this survey, we have systematized the findings according to the defined criteria: the representation level (the level of linguistic units for which representations are being learned); the input level (the granularity of data upon input); the model type (the general strategy for representation learning); the model architecture (neural network architecture); and the model supervision (an indication of the need for labeled training data). Still, there are additional insights into the criteria for comparing, selecting, and evaluating the representation models, as well as insights into which model properties are important to consider when building or choosing an architecture for a specific downstream task and language.

10.1. Comparison

The majority of shallow models are unsupervised, learn word-level representations, and take word-level input. They perform well on simple tasks like measuring word similarity and are the easiest architecture to train.

Word-level input was popular among recurrent models, but recently subword-level input is more frequently used. In this study, we noticed that recurrent models do not prefer one type of supervision or representation level more frequently than other types.

The majority of recursive models work with word-level input and learn sentence+ representations for larger chunks of text like phrases, sentences, paragraphs, or entire documents. Recently, recursive models that generate latent trees are preferable due to the low data preparation requirement, but unsupervised learning is still not the norm. They are a good fit for semantic and sentiment tasks [75] where the tree structure of a text is important. Recursive models are currently underexplored, and some of the popular neural network libraries (e.g., TensorFlow) do not support them well (with some exceptions like DyNet).

Convolutional and attention models mainly use subword or word-level input. Attention models most frequently learn sentence+ representations. Unsupervised models are preferred again because of large amounts of data usable without any manual labeling. For tasks that process long texts with long-range dependencies, attention models outperform all the other models. Lately, attention models are increasing in size to improve the performance but at the cost of increased training time and higher memory requirements.

Shallow models are effective when dealing with word-level units, but if the downstream task depends on higher-level units (e.g., phrases, sentences, etc.), recursive, convolutional, or attention models are a better choice. Currently, a significant drawback with recursive models is a complicated training procedure that is hard to implement and leads to slower training times. Convolutional models can learn local patterns and are a good fit for computer vision tasks. It has been shown that convolutions can be used for text, but currently, attention models outperform them on text-related tasks as attention mechanisms can learn long-range dependencies.

Subword input provides the most flexibility, as unseen and rare words are easily dealt with, and input vectors are not as large. The subword-level facilitates the learning of neural models for morphologically rich languages. Complex downstream tasks benefit from sentence+ representations, but word representations and shallow models still work well for simple downstream tasks.

10.2. Computational Complexity

Model's computational complexity is an important factor when deciding which model or architecture to use, or which one is superior. As such, when reporting results for a model, information about computational complexity should be a priority. Some of the models in this survey use the number of trainable parameters as a proxy for computational complexity.

Word2Vec's skip-gram model has the complexity $C \times (D + D \times \log_2(V))$ (where C is window size, V is size of the vocabulary, and D is dimensionality of the representations) [22]. If vocabulary contains 30k words, representations' dimensionality is 300, and window size is 10, Word2Vec's computational complexity would be 47618. If accounting for the number of training steps, the complexity is equal to $47,618 \times E \times T$, where E is the number of the training epochs and T is the number of the words in the training set. On the other end of computational complexity, GPT-2 has 1.5B trainable parameters [3], which if used as a proxy for complexity is approximately 3,000,000% higher complexity than Word2Vec.

10.3. Evaluation

A representation model's quality can be evaluated as its capability to generalize to multiple unrelated tasks. A model that performs well on different unrelated downstream tasks is general and is not overfitted to a single task. Benchmark standardization is visible amongst recent models, where models are evaluated on the same set of tasks, and the number of different tasks models are evaluated on is increasing (Figure A1 in Appendix A). The accuracy, with the exception of task-specific evaluation measures, seems to be a widely adopted evaluation principle across different downstream tasks.

The most used tasks for the evaluation of the models are sentiment classification, natural language inference, and sentence+ similarity as listed in Table A2 in Appendix A. The tasks of paraphrase detection, machine translation, and question answering are popular as well. A trend can be seen through time, where earlier models were evaluated most frequently on just one task; however, recently that number has grown, and now some models are evaluated on up to approximately twenty tasks (e.g., XLNet). This growth is the reflection of the trend towards general neural representation models.

10.4. Challenges

Large industry-based research groups are developing extremely large models with billions of parameters that require powerful hardware and custom systems for training. Even the inference of test cases becomes problematic with the largest models [104]. For smoother progress of large text representation models, new solutions that can support large models' training on clusters are expected.

Further, to be able to compare representation models, a standardized evaluation strategy will have to emerge including definitions for standard datasets, tasks, and measures. There are several obstacles for research teams with limited computational resources when studying the comparative performance of their models against large models. Namely, contrasting the models of different magnitudes of parameters and training time will lead to inconsistent and possibly misleading results. To obtain comparable results, the large models would have to be evaluated in several versions, ranging from the smallest to the biggest (measured by the number of trainable parameters). Accordingly, low-resourced research teams would be able to compare models fairly by evaluating the model of the same magnitude as an adequate version of the large model.

Evaluating a model on the same downstream task used for training does not assess its generality. One solution for addressing the generality evaluation is to use cross-validation across tasks. A model will be trained on one of the tasks and evaluated on the rest, repeating that process for every task in the benchmark. Hence, there is a need for defining a better evaluation measure that will quantify the quality of the neural text representation model's generality across tasks.

10.5. Future Research

The majority of neural text representation models learn from large texts with the aim of encoding, decoding, classifying, translating, or predicting text sequences. Such approaches can generate text representations that are useful for natural language processing tasks of a similar nature.

For future NLP tasks in the field of AI, which are general and deep, new learning strategies will have to be constructed that will be able to learn beyond the current specialized compression functions for text (text representation models). The new learning strategies would have to be able to learn text representations that capture not only information present in the text, but common sense and human intuition as well. For that kind of learning, it is possible that language models will have to learn as agents in environments in order to grasp general concepts about the interaction with the world and between the elements that exist in the world.

Other than learning from the new information presented by the outside world (be that text from Wikipedia or interaction with objects), future models could additionally learn on their own by creating analogies and imagining new scenarios.

11. Conclusions

This survey has systematized a large number of text representation models by representation level, input level, model type, model architecture, and model supervision. We do not limit this survey by model architecture or learned representation level. Early models most frequently learned word representation, as they require the simplest models. More recent models mostly learn phrase, sentence, or document representations, as they are better suited for complex tasks. Older models used word-level input, which does not require complex model architectures. Subword-level input soon became more prevalent as it is more effective and requires less memory. As for model types, autoencoding and autoregressive models allow unsupervised learning (whereas classification models mostly do not), which is becoming the standard.

Shallow neural networks are simple and easy to train, but their learned representations are simple as well, which makes them ineffective for complex tasks. Recurrent neural networks are made for sequential data, which text can be seen as. Such neural networks can learn phrase, sentence, or document representations easily, but they still have issues (e.g., vanishing and exploding gradient). Recursive neural networks are a generalization of recurrent neural networks that are able to preserve not only sequential information but structure information as well. They are currently not explored as much as the other architectures. Convolutional neural networks detect local patterns and reduce dimensions with pooling, but are more frequently used for computer vision than for natural language processing. Attentive neural networks are comparable to convolutional and recurrent neural networks in terms of input and representation level. But attention models are capable of learning longer long-range dependencies resulting in a boosted performance on various tasks.

Still, it is difficult to rank the models. Our suggestion is to set criteria, similar to the ones used in this study, and systematically compare models against them, to decide which one is the best fit for the language and problem at hand. This is of the utmost importance when we opt for the training of models that go beyond the English language. For research teams with limited computational resources, as it is hard to compete with large industry-based research groups, it is advisable to research and develop new architectures and learning strategies that provide novel results or characteristics. Likewise, the definition of general evaluation principles can contribute to meaningful and feasible comparisons of novel models even for the smaller research teams.

For future work, a number of research directions exist. Multimodal learning has the potential to help learn better text representation, but more training data is needed, or different learning strategies. Recursive neural networks are currently underexplored but seem like a good fit for natural language. Specialized lexical resources have the potential to improve text representation as well. The standardization of evaluation strategy for representation models is very possibly the most important step towards the improvement of neural networks for general natural language processing.

Author Contributions: Conceptualization and methodology, S.M.-I. and A.M.; formal analysis, investigation, resources, and data curation, K.B. and S.M.-I.; writing—original draft preparation, K.B.; writing—review and editing, S.M.-I. and K.B.; visualization, K.B.; supervision and project administration, A.M.; funding acquisition, A.M. and S.M.-I. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the University of Rijeka under the project numbers uniri-drustv-18-20 and uniri-drustv-18-38.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Table A1. Datasets used for evaluation of models we referenced in comparisons.

Dataset	Models
WordSim-353	Huang et al. [40], GloVe [61], FastText [49]
SICK	Skip-Thoughts [64], FastSent [66], RL-SPINN [76], DiSAN [91]
SNLI	CoVe [67], Kim et al. [89], Lin et al. [90], DiSAN [91], ReSAN [93], Star-Transformer [96]
MultiNLI	ALBERT [100]
MRPC	Skip-Thoughts [64], FastSent [66], ELECTRA [102]
STS	FastSent [66], ELECTRA [102]
SSWR	Word2Vec [22,47], GloVe [61], FastText [49]
MSR	Word2Vec [22,47]
SST	Doc2Vec [62], RCNN [65], CoVe [67], ELMo [41], Tree-LSTM [75], RL-SPINN [76], ST-Gumbel [77], DCNN [82], BERT [54], Bi-BloSAN [92], ELECTRA [102]
IMDB	Doc2Vec [62], CoVe [67], HAN [88]
MR	FastSent [66], RAE [72], MV-RNN [73], AdaSent [74], CNN-LSTM [84], CDWE [43]
CR	FastSent [66], CNN-LSTM [84], DiSAN [91], Bi-BloSAN [92]
SemEval'17	XML [95]
TREC	FastSent [66], CoVe [67], CNN-LSTM [84], CDWE [43], DiSAN [91], Transformer-XL [97]
SQuAD	CoVe [67], ALBERT [100]
MPQA	FastSent [66], RAE [72], AdaSent [74], CNN-LSTM [84], DiSAN [91], Bi-BloSAN [92]
NQ	REALM [7]
WQ	REALM [7]
WSJ	DIORA [78]
AG News	CDWE [43]
GLUE	BERT [54], SpanBERT [101]
RACE	ALBERT [100]
SUBJ	FastSent [66], CNN-LSTM [84], DiSAN [91], Bi-BloSAN [92]
WMT	Cho et al. [63], Seq2Seq [5], ByteNet [83], Transformer [51], MASS [98]

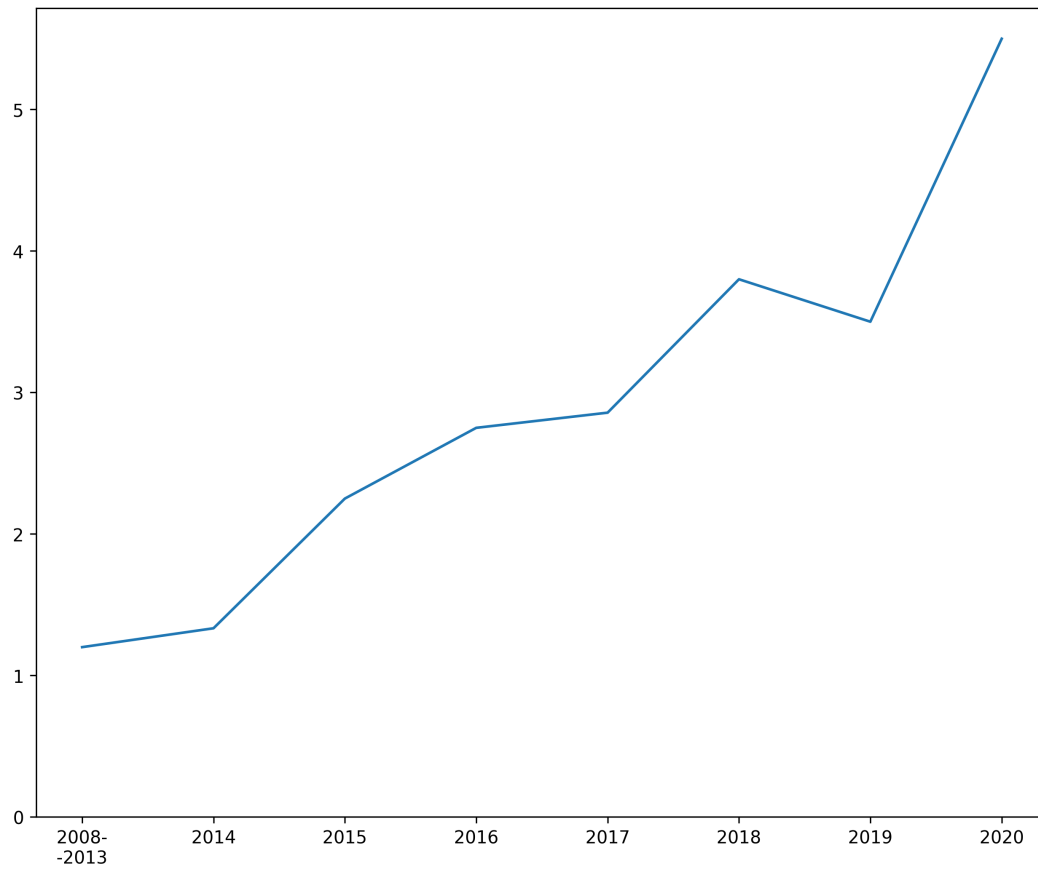


Figure A1. The average number of tasks models are evaluated on, by year.

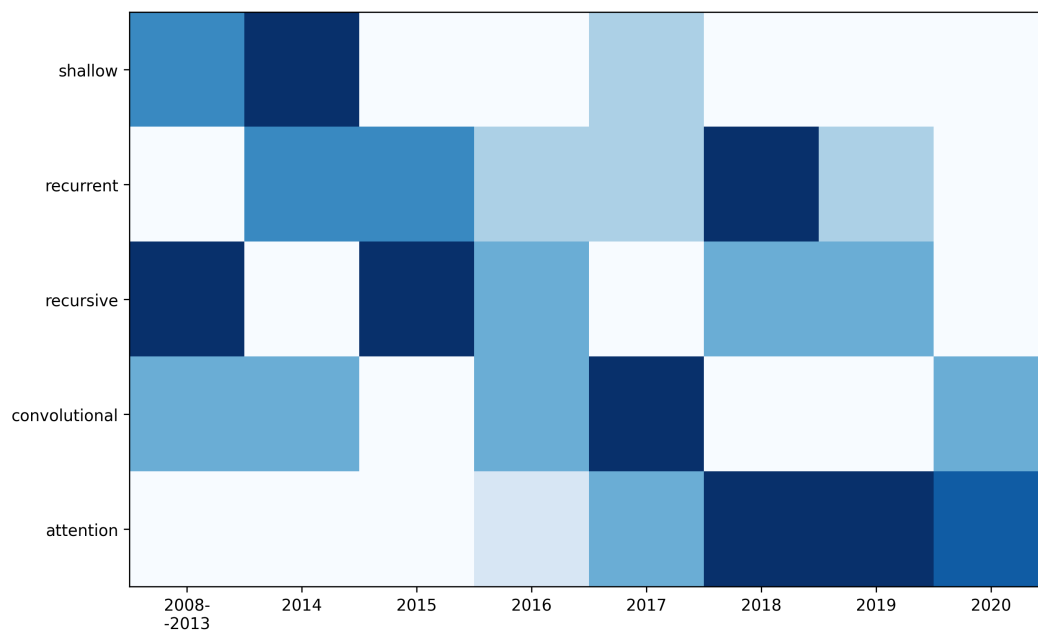


Figure A2. Number of models categorized by the main architecture and year. Values are normalized by rows. Higher values are darker.

References

1. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008.
2. Goldberg, Y. Neural network methods for natural language processing. *Synth. Lect. Hum. Lang. Technol.* **2017**, *10*, 1–309. [CrossRef]
3. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
4. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, UK, 2016.
5. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. 2014. Available online: <https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf> (accessed on 29 October 2020).
6. Jianqiang, Z.; Xiaolin, G.; Xuejun, Z. Deep Convolution Neural Networks for Twitter Sentiment Analysis. *IEEE Access* **2018**, *6*, 23253–23260. [CrossRef]
7. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; Chang, M.W. Realm: Retrieval-augmented language model pre-training. *arXiv* **2020**, arXiv:2002.08909.
8. Hailu, T.T.; Yu, J.; Fantaye, T.G. A Framework for Word Embedding Based Automatic Text Summarization and Evaluation. *Information* **2020**, *11*, 78. [CrossRef]
9. Bodrunova, S.S.; Orekhov, A.V.; Blekanov, I.S.; Lyudkevich, N.S.; Tarasov, N.A. Topic Detection Based on Sentence Embeddings and Agglomerative Clustering with Markov Moment. *Future Internet* **2020**, *12*, 144. [CrossRef]
10. Martinčić-Ipšić, S.; Miličić, T.; Todorovski, L. The Influence of Feature Representation of Text on the Performance of Document Classification. *Appl. Sci.* **2019**, *9*, 743. [CrossRef]
11. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [CrossRef]
12. Jing, K.; Xu, J.; He, B. A Survey on Neural Network Language Models. *arXiv* **2019**, arXiv:1906.03591.
13. Camacho-Collados, J.; Pilehvar, M.T. From word to sense embeddings: A survey on vector representations of meaning. *J. Artif. Intell. Res.* **2018**, *63*, 743–788. [CrossRef]
14. Ruder, S.; Vulić, I.; Søgaard, A. A survey of cross-lingual word embedding models. *J. Artif. Intell. Res.* **2019**, *65*, 569–631. [CrossRef]
15. Aßenmacher, M.; Heumann, C. On the comparability of Pre-trained Language Models. *arXiv* **2020**, arXiv:2001.00781.
16. Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; Ruppín, E. Placing search in context: The concept revisited. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 406–414.
17. Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; Zamparelli, R. A SICK Cure for the Evaluation of Compositional Distributional Semantic Models. In Proceedings of the 9th Language Resources and Evaluation Conference, Reykjavik, Iceland, 26–31 May 2014; pp. 216–223. Available online: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf (accessed on 29 October 2020).
18. Bowman, S.R.; Angeli, G.; Potts, C.; Manning, C.D. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Lisbon, Portugal, 17–21 September 2015.
19. Williams, A.; Nangia, N.; Bowman, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1 (Long Papers)), New Orleans, LA, USA, 1–6 June 2018; pp. 1112–1122.
20. Dolan, B.; Quirk, C.; Brockett, C. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In Proceedings of the 20th international conference on Computational Linguistics, Geneva, Switzerland, 23–27 August 2004; p. 350.

21. Agirre, E.; Banea, C.; Cardie, C.; Cer, D.; Diab, M.; Gonzalez-Agirre, A.; Guo, W.; Mihalcea, R.; Rigau, G.; Wiebe, J. Semeval-2014 task 10: Multilingual semantic textual similarity. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Dublin, Ireland, 23–24 August 2014; pp. 81–91.
22. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
23. Mikolov, T.; Yih, W.T.; Zweig, G. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–14 June 2013; pp. 746–751.
24. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.; Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642.
25. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
26. Pang, B.; Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Ann Arbor, MI, USA, 17 June 2005; pp. 115–124.
27. Hu, M.; Liu, B. Mining and summarizing customer reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22 August 2004; pp. 168–177.
28. Rosenthal, S.; Farra, N.; Nakov, P. SemEval-2017 Task 4: Sentiment Analysis in Twitter. In Proceedings of the SemEval '17 11th International Workshop on Semantic Evaluation, Vancouver, BC, Canada, 3–4 August 2017.
29. Voorhees, E.M.; Harman, D. Overview of TREC 2002. In Proceedings of the Eleventh Text REtrieval Conference, Gaithersburg, MD, USA, 19–22 November 2002. Available online: <https://trec.nist.gov/pubs/trec11/papers/OVERVIEW.11.pdf> (accessed on 29 October 2020).
30. Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv* **2016**, arXiv:1606.05250.
31. Wiebe, J.; Wilson, T.; Cardie, C. Annotating expressions of opinions and emotions in language. *Lang. Resour. Eval.* **2005**, *39*, 165–210. [[CrossRef](#)]
32. Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; et al. Natural questions: A benchmark for question answering research. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 453–466. [[CrossRef](#)]
33. Berant, J.; Chou, A.; Frostig, R.; Liang, P. Semantic parsing on freebase from question-answer pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1533–1544.
34. Marcus, M.; Santorini, B.; Marcinkiewicz, M.A. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguist.* **1993**, *19*, 313–330.
35. Wang, J.; Wang, Z.; Zhang, D.; Yan, J. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In Proceedings of the International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 2915–2921.
36. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv* **2018**, arXiv:1804.07461.
37. Lai, G.; Xie, Q.; Liu, H.; Yang, Y.; Hovy, E. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 785–794. [[CrossRef](#)]
38. Pang, B.; Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, Barcelona, Spain, 21–26 July 2004; p. 271.
39. Jelinek, F. *Statistical Methods for Speech Recognition*; MIT Press: Cambridge, MA, USA, 1998.

40. Huang, E.H.; Socher, R.; Manning, C.D.; Ng, A.Y. Improving word representations via global context and multiple word prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Jeju Island, Korea, 8–14 July 2012; pp. 873–882.
41. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.
42. Akbik, A.; Blythe, D.; Vollgraf, R. Contextual string embeddings for sequence labeling. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 21–25 August 2018; pp. 1638–1649.
43. Shuang, K.; Zhang, Z.; Loo, J.; Su, S. Convolution–deconvolution word embedding: An end-to-end multi-prototype fusion embedding method for natural language processing. *Inf. Fusion* **2020**, *53*, 112–122. [[CrossRef](#)]
44. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 29 October 2020).
45. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*; MIT Press: Vancouver, BC, Canada, 2019; pp. 5754–5764.
46. Botha, J.; Blunsom, P. Compositional morphology for word representations and language modelling. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1899–1907.
47. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and Their Compositionality. 2013. Available online: <https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf> (accessed on 29 October 2020).
48. Kim, Y.; Jernite, Y.; Sontag, D.; Rush, A.M. Character-aware neural language models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
49. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [[CrossRef](#)]
50. Gage, P. A new algorithm for data compression. *C Users J.* **1994**, *12*, 23–38.
51. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. 2017. Available online: <https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf> (accessed on 29 October 2020).
52. Kudo, T.; Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv* **2018**, arXiv:1808.06226.
53. Schuster, M.; Nakajima, K. Japanese and korean voice search. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 5149–5152.
54. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
55. Goldberg, Y. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.* **2016**, *57*, 345–420. [[CrossRef](#)]
56. Shi, T.; Liu, Z. Linking GloVe with word2vec. *arXiv* **2014**, arXiv:1411.5595.
57. Levy, O.; Goldberg, Y. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*; MIT Press: Montréal, QC, Canada, 2014; pp. 2177–2185.
58. Harris, Z.S. Distributional structure. *Word* **1954**, *10*, 146–162. [[CrossRef](#)]
59. Firth, J. A Synopsis of Linguistic Theory 1930–1955. In *Studies in Linguistic Analysis*; Palmer, F., Ed.; Philological Society: Oxford, UK, 1957.
60. Levy, O.; Goldberg, Y. Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, MD, USA, 22–27 June 2014; pp. 302–308.
61. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.

62. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1188–1196.
63. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
64. Kiros, R.; Zhu, Y.; Salakhutdinov, R.R.; Zemel, R.; Urtasun, R.; Torralba, A.; Fidler, S. Skip-thought vectors. In *Advances in Neural Information Processing Systems*; Mit Press: Montréal, QC, Canada, 2015; pp. 3294–3302.
65. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
66. Hill, F.; Cho, K.; Korhonen, A. Learning distributed representations of sentences from unlabelled data. *arXiv* **2016**, arXiv:1602.03483.
67. McCann, B.; Bradbury, J.; Xiong, C.; Socher, R. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*; Mit Press: Long Beach, CA, USA, 2017; pp. 6294–6305.
68. Subramanian, S.; Trischler, A.; Bengio, Y.; Pal, C.J. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv* **2018**, arXiv:1804.00079.
69. Artetxe, M.; Schwenk, H. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 597–610. [[CrossRef](#)]
70. Auli, M.; Galley, M.; Quirk, C.; Zweig, G. Joint language and translation modeling with recurrent neural networks. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1044–1054.
71. Conneau, A.; Lample, G.; Rinott, R.; Williams, A.; Bowman, S.R.; Schwenk, H.; Stoyanov, V. XNLI: Evaluating cross-lingual sentence representations. *arXiv* **2018**, arXiv:1809.05053.
72. Socher, R.; Pennington, J.; Huang, E.H.; Ng, A.Y.; Manning, C.D. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Edinburgh, UK, 27–31 July 2011; pp. 151–161.
73. Socher, R.; Huval, B.; Manning, C.D.; Ng, A.Y. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, Jeju Island, Korea, 12–14 July 2012; pp. 1201–1211.
74. Zhao, H.; Lu, Z.; Poupart, P. Self-adaptive hierarchical sentence model. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015.
75. Tai, K.S.; Socher, R.; Manning, C.D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv* **2015**, arXiv:1503.00075.
76. Yogatama, D.; Blunsom, P.; Dyer, C.; Grefenstette, E.; Ling, W. Learning to compose words into sentences with reinforcement learning. *arXiv* **2016**, arXiv:1611.09100.
77. Choi, J.; Yoo, K.M.; Lee, S.G. Learning to compose task-specific tree structures. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
78. Drozdov, A.; Verga, P.; Yadav, M.; Iyyer, M.; McCallum, A. Unsupervised latent tree induction with deep inside-outside recursive autoencoders. *arXiv* **2019**, arXiv:1904.02142.
79. Nakagawa, T.; Inui, K.; Kurohashi, S. Dependency tree-based sentiment classification using CRFs with hidden variables. In Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, CA, USA, 2–4 June 2010; pp. 786–794.
80. Bowman, S.R.; Gauthier, J.; Rastogi, A.; Gupta, R.; Manning, C.D.; Potts, C. A fast unified model for parsing and sentence understanding. *arXiv* **2016**, arXiv:1603.06021.
81. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 160–167.
82. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. *arXiv* **2014**, arXiv:1404.2188.

83. Kalchbrenner, N.; Espeholt, L.; Simonyan, K.; Oord, A.v.d.; Graves, A.; Kavukcuoglu, K. Neural machine translation in linear time. *arXiv* **2016**, arXiv:1610.10099.
84. Gan, Z.; Pu, Y.; Henaou, R.; Li, C.; He, X.; Carin, L. Learning generic sentence representations using convolutional neural networks. *arXiv* **2016**, arXiv:1611.07897.
85. Zaremba, W.; Sutskever, I.; Vinyals, O. Recurrent neural network regularization. *arXiv* **2014**, arXiv:1409.2329.
86. Chung, J.; Cho, K.; Bengio, Y. A character-level decoder without explicit segmentation for neural machine translation. *arXiv* **2016**, arXiv:1603.06147.
87. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
88. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.
89. Kim, Y.; Denton, C.; Hoang, L.; Rush, A.M. Structured attention networks. *arXiv* **2017**, arXiv:1702.00887.
90. Lin, Z.; Feng, M.; Santos, C.N.d.; Yu, M.; Xiang, B.; Zhou, B.; Bengio, Y. A structured self-attentive sentence embedding. *arXiv* **2017**, arXiv:1703.03130.
91. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Pan, S.; Zhang, C. Disan: Directional self-attention network for rnn/cnn-free language understanding. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
92. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Zhang, C. Bi-directional block self-attention for fast and memory-efficient sequence modeling. *arXiv* **2018**, arXiv:1804.00857.
93. Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Wang, S.; Zhang, C. Reinforced self-attention network: A hybrid of hard and soft attention for sequence modeling. *arXiv* **2018**, arXiv:1801.10296.
94. Liu, Y.; Lapata, M. Learning structured text representations. *Trans. Assoc. Comput. Linguist.* **2018**, *6*, 63–75. [[CrossRef](#)]
95. Lample, G.; Conneau, A. Cross-lingual language model pretraining. *arXiv* **2019**, arXiv:1901.07291.
96. Guo, Q.; Qiu, X.; Liu, P.; Shao, Y.; Xue, X.; Zhang, Z. Star-transformer. *arXiv* **2019**, arXiv:1902.09113.
97. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv* **2019**, arXiv:1901.02860.
98. Song, K.; Tan, X.; Qin, T.; Lu, J.; Liu, T.Y. Mass: Masked sequence to sequence pre-training for language generation. *arXiv* **2019**, arXiv:1905.02450.
99. Reimers, N.; Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv* **2019**, arXiv:1908.10084.
100. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv* **2019**, arXiv:1909.11942.
101. Joshi, M.; Chen, D.; Liu, Y.; Weld, D.S.; Zettlemoyer, L.; Levy, O. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 64–77. [[CrossRef](#)]
102. Clark, K.; Luong, M.T.; Le, Q.V.; Manning, C.D. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv* **2020**, arXiv:2003.10555.
103. Conneau, A.; Lample, G.; Ranzato, M.; Denoyer, L.; Jégou, H. Word translation without parallel data. *arXiv* **2017**, arXiv:1710.04087.
104. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).