**Recent trends in deep learning based Natural Language Processing**

I. INTRODUCTION

Natural language processing (NLP) is a theory-motivated range of computational techniques for the automatic analysis and representation of human language. NLP enables computers to perform a wide range of natural language related tasks at all levels, ranging from parsing and part-of-speech (POS) tagging, to machine translation and dialogue systems.

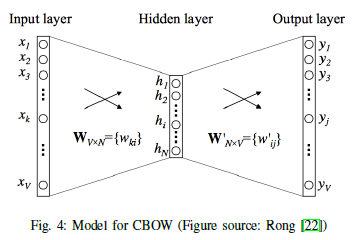
II. DISTRIBUTED REPRESENTATION

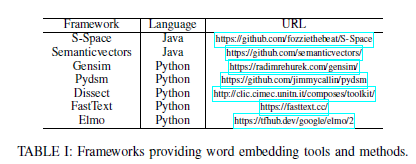
1. Word Embeddings

Word embeddings are often used as the first data processing layer in a deep learning model. These embeddings have proven to be efficient in capturing context similarity, analogies and due to its smaller dimensionality, are fast and efficient in processing core NLP tasks.

1. Word2vec

CBOW computes the conditional probability of a target word given the context words surrounding it across a window of size k. On the other hand, the skip-gram model does the exact opposite of the CBOW model, by predicting the surrounding context words given the central target word. The context words are assumed to be located symmetrically to the target words within a distance equal to the window size in both directions.



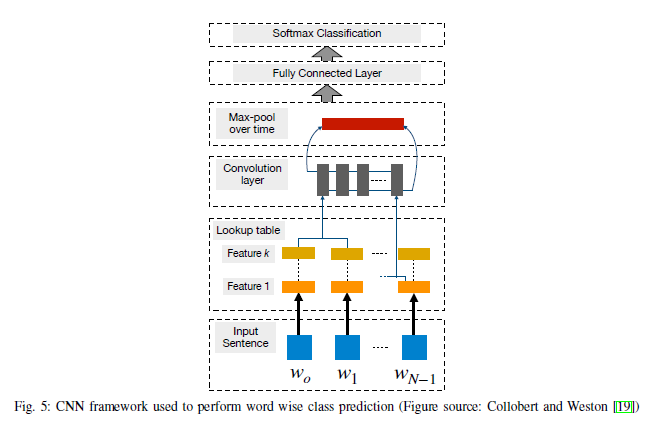


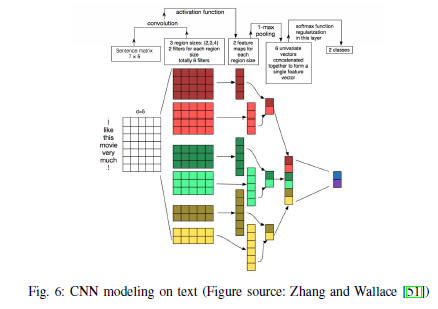
1. Character Embeddings

Word embeddings are able to capture syntactic and semantic information, yet for tasks such as POS-tagging and NER, intra-word morphological and shape information can also be very useful.

1. Contextualized Word Embeddings

III. CONVOLUTIONAL NEURAL NETWORKS

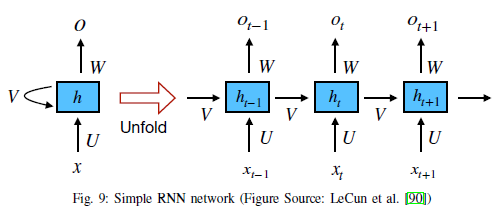


1. Basic CNN
2. Sentence Modeling
3. Window Approach
4. Applications

IV. RECURRENT NEURAL NETWORKS

RNNs use the idea of processing sequential information. The term “recurrent” applies as they perform the same task over each instance of the sequence such that the output is dependent on the previous computations and results. Generally, a fixed-size vector is produced to represent a sequence by feeding tokens one by one to a recurrent unit. In a way, RNNs have “memory” over previous computations and use this information in current processing.

1. Need for Recurrent Networks



Given that an RNN performs sequential processing by modeling units in sequence, it has the ability to capture the inherent sequential nature present in language, where units are characters, words or even sentences. Words in a language develop their semantical meaning based on the previous words in the sentence.

1. RNN models
2. Simple RNN:

Calculation of st is based as per the equation:



Thus, st is calculated based on the current input and the previous time step’s hidden state. The function f is taken to be a non-linear transformation such as tanh, ReLU and U; V;W account for weights that are shared across time. In practice, however, these simple RNN networks suffer from the infamous vanishing gradient problem, which makes it really hard to learn and tune the parameters

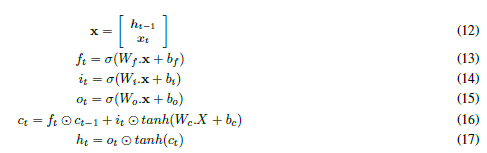
of the earlier layers in the network.

This limitation was overcome by various networks such as long short-term memory (LSTM), gated recurrent units (GRUs), and residual networks (ResNets), where the first two are the most used RNN variants in NLP applications.

1. Long Short-Term Memory:

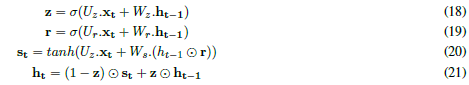
LSTM has additional “forget” gates over the simple RNN. Its unique mechanism enables it to overcome both the vanishing and exploding gradient problem.

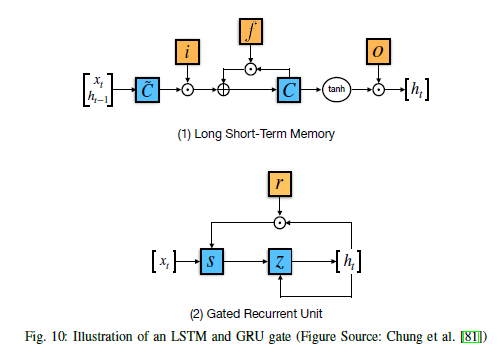
Unlike the vanilla RNN, LSTM allows the error to back-propagate through unlimited number of time steps. Consisting of three gates: input, forget and output gates, it calculates the hidden state by taking a combination of these three gates as per the equations below:



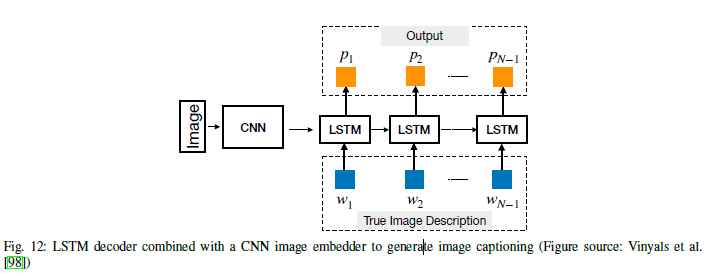
1. Gated Recurrent Units:

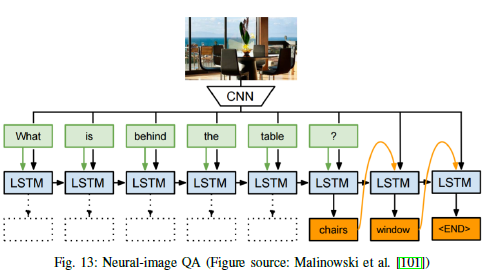
Another gated RNN variant called GRU [82] (Fig. 10) of lesser complexity was invented with empirically similar performances to LSTM in most tasks. GRU comprises of two gates, reset gate and update gate, and handles the flow of information like an LSTM sans a memory unit. Thus, it exposes the whole hidden content without any control. Being less complex, GRU can be a more efficient RNN than LSTM. The working of GRU is as follows:





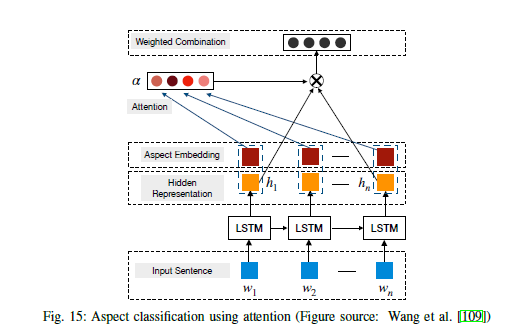
1. Applications
2. RNN for word-level classification:
3. RNN for sentence-level classification:
4. RNN for generating language:





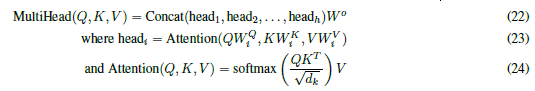
1. Attention Mechanism

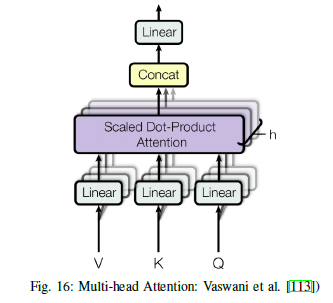
The attention mechanism can be broadly seen as mapping a query and a set of key-value pairs to an output, where all the mentioned components are vectors. The output is a combination of the values whose weights are determined by the compatibility between the query and the corresponding keys. This output amounts to the “context” of the input used in decoding the output.



1. Parallelized Attention: The Transformer

The Transformer consists stacked layers in both encoder and decoder components.

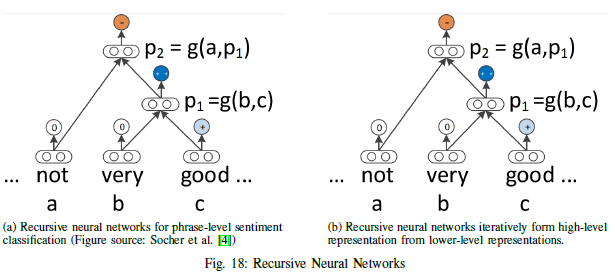




V. RECURSIVE NEURAL NETWORKS

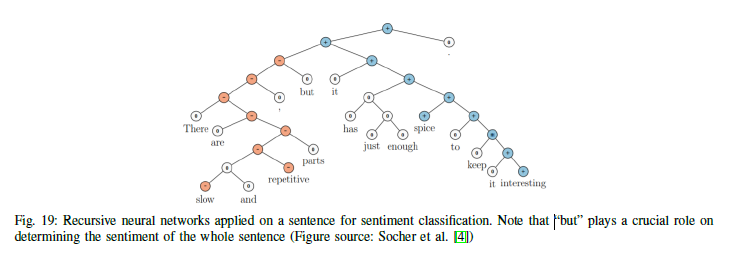
Recurrent neural networks represent a natural way to model sequences. Arguably, however, language exhibits a natural recursive structure, where words and sub-phrases combine into phrases in a hierarchical manner. Such structure can be represented by a constituency parsing tree. Thus, tree-structured models have been used to better make use of such syntactic interpretations of sentence structure. Specifically, in a recursive neural network, the representation of each non-terminal node in a parsing tree is determined by the representations of all its children.

1. Basic model



1. Applications

One natural application of recursive neural networks is parsing. A scoring function is defined on the phrase representation to calculate the plausibility of that phrase. Beam search is usually applied for searching the best tree. The model is trained with the max-margin objective.



VI. DEEP REINFORCED MODELS AND DEEP UNSUPERVISED LEARNING

1. Reinforcement learning for sequence generation

Reinforcement learning is a method of training an agent to perform discrete actions before obtaining a reward. In NLP, tasks concerning language generation can sometimes be cast as reinforcement learning problems.

In its original formulation, RNN language generators are typically trained by maximizing the likelihood of each token in the ground-truth sequence given the current hidden state and the previous tokens. Termed “teacher forcing”, this training scheme provides the real sequence prefix to the generator during each generation (loss evaluation) step. At test time, however, ground truth tokens are then replaced by a token generated by the model itself. This discrepancy between training and inference, termed “exposure bias”, can yield errors that can accumulate quickly along the generated sequence.

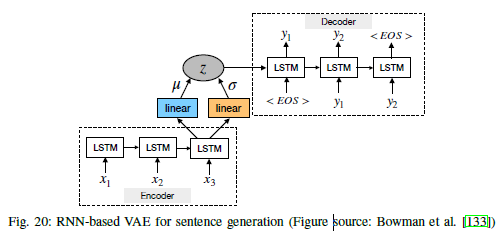
Another problem with the word-level maximum likelihood strategy, when training auto-regressive language generation models, is that the training objective is different from the test metric.

There are two well-known shortcomings of reinforcement learning. To make reinforcement learning tractable, it is desired to carefully handle the state and action space, which in the end may restrict expressive power and learning capacity of the model. Secondly, the need for training the reward functions makes such models hard to design and measure at run time.

1. Unsupervised sentence representation learning

Similar to word embeddings, distributed representation for sentences can also be learned in an unsupervised fashion.

1. Deep generative models



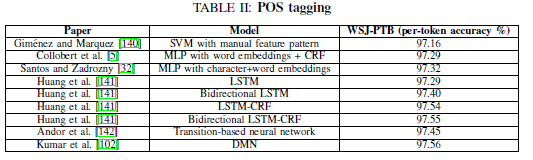
The VAE imposes a prior distribution on the hidden code space which makes it possible to draw proper samples from the model. It modifies the autoencoder architecture by replacing the deterministic encoder function with a learned posterior recognition model. The model consists of encoder and generator networks which encode data examples to latent representation and generate samples from the latent space, respectively. It is trained by maximizing a variational lower bound on the loglikelihood of observed data under the generative model.

VII. MEMORY-AUGMENTED NETWORKS

The attention mechanism stores a series of hidden vectors of the encoder, which the decoder is allowed to access during the generation of each token. Here, the hidden vectors of the encoder can be seen as entries of the model’s “internal memory”.

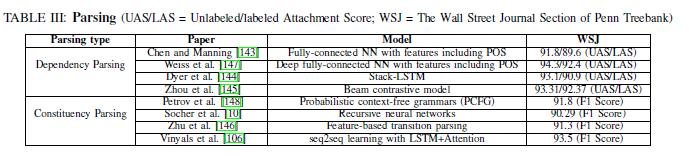
VIII. PERFORMANCE OF DIFFERENT MODELS ON DIFFERENT NLP TASKS

1. POS tagging

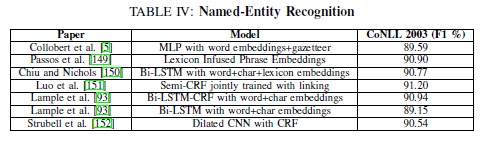


1. Parsing

There are two types of parsing: dependency parsing, which connects individual words with their relations, and constituency parsing, which iteratively breaks text into sub-phrases.



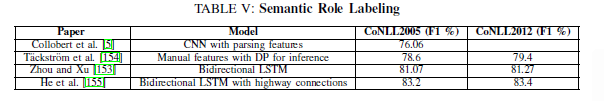
1. Named-Entity Recognition



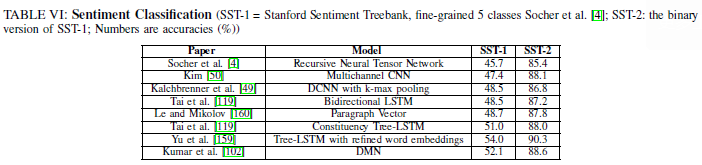
1. Semantic Role Labeling

Semantic role labeling (SRL) aims to discover the predicate-argument structure of each predicate in a sentence.

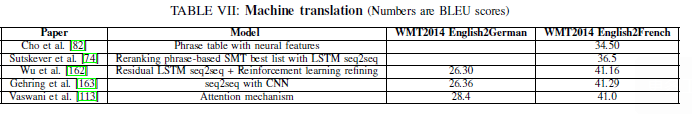
Traditional SRL systems consist of several stages: producing a parse tree, identifying which parse tree nodes represent the arguments of a given verb, and finally classifying these nodes to determine the corresponding SRL tags. Each classification process usually entails extracting numerous features and feeding them into statistical models.



1. Sentiment Classification

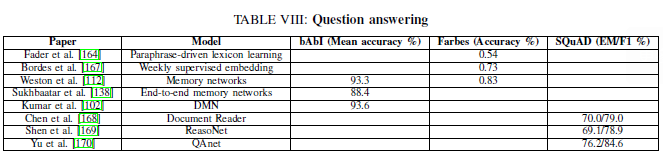


1. Machine Translation



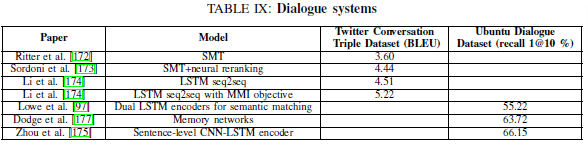
1. Question answering

QA problems take many forms. Some rely on large KBs to answer open-domain questions, while others answer a question based on a few sentences or a paragraph (reading comprehension).

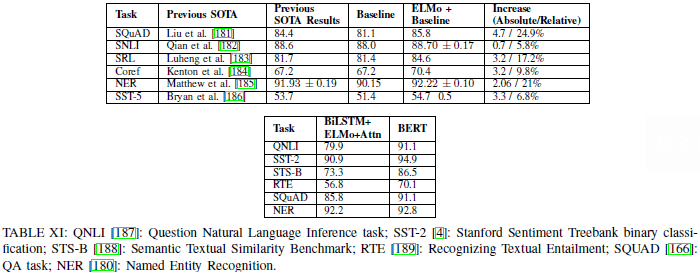


1. Dialogue Systems

Two types of dialogue systems have been developed: generation-based models and retrieval-based models.



1. Contextual Embeddings



IX. CONCLUSION

Deep learning offers a way to harness large amount of computation and data with little engineering by hand.