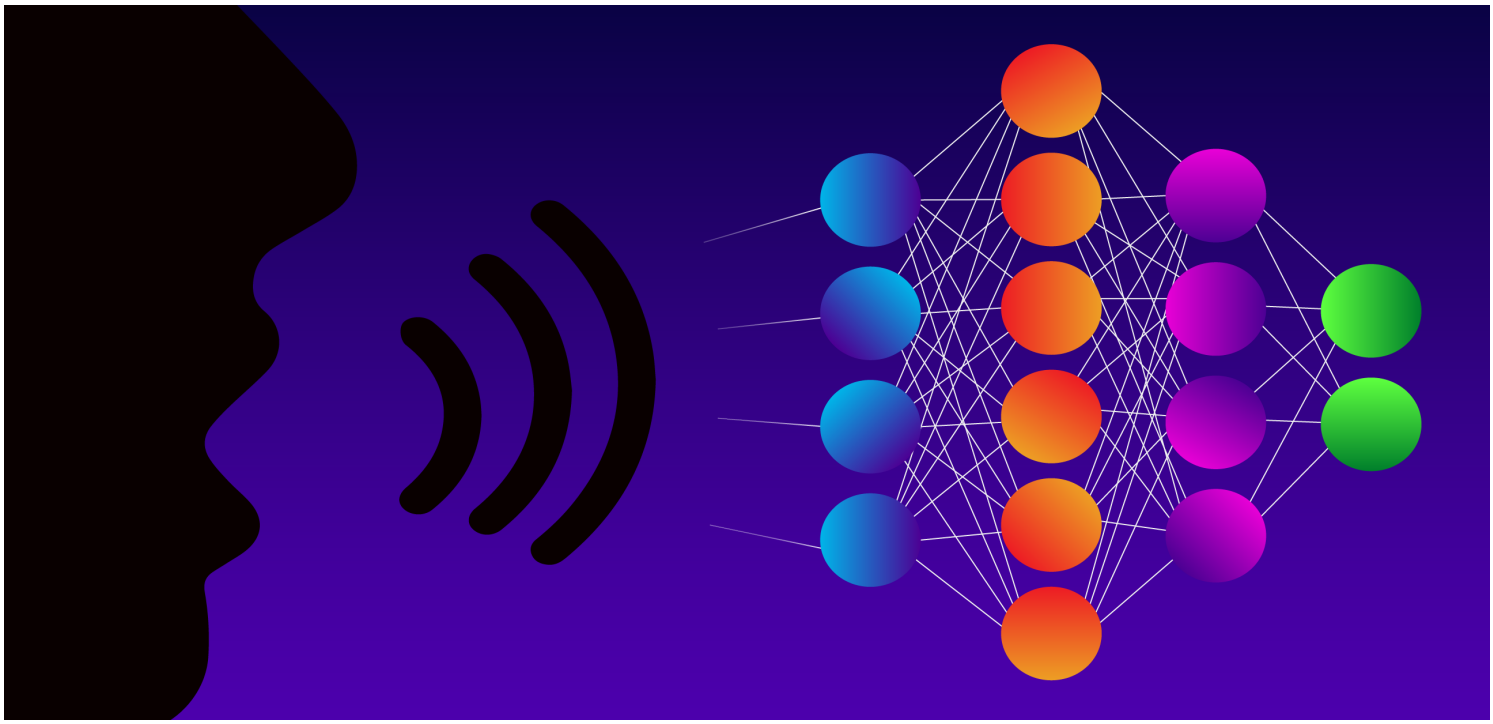
Deep Learning for NLP: An Overview of Recent Trends

Elvis Follow
Aug 23, 2018 · 14 min read



In a timely new paper, Young and colleagues discuss some of the recent trends in deep learning based natural language processing (NLP) systems and applications. The focus of the paper is on the review and comparison of models and methods that have achieved state-of-the-art (SOTA) results on various NLP tasks such as visual question answering (QA) and machine translation. In this comprehensive review, the reader will get a detailed understanding of the past, present, and future of deep learning in NLP. In addition, readers will also learn some of the *current best practices* for applying deep learning in NLP. Some topics include:

- The rise of distributed representations (e.g., word2vec)
- Convolutional, recurrent, and recursive neural networks
- Applications in reinforcement learning
- Recent development in unsupervised sentence representation learning
- Combining deep learning models with memory-augmenting strategies

What is NLP?

Natural language processing (NLP) deals with building computational algorithms to automatically analyze and represent human language. NLP-based systems have enabled a wide range of applications such as Google's powerful search engine, and more recently, Amazon's voice assistant named Alexa. NLP is also useful to teach machines the ability to perform complex natural language related tasks such as machine translation and dialogue generation.

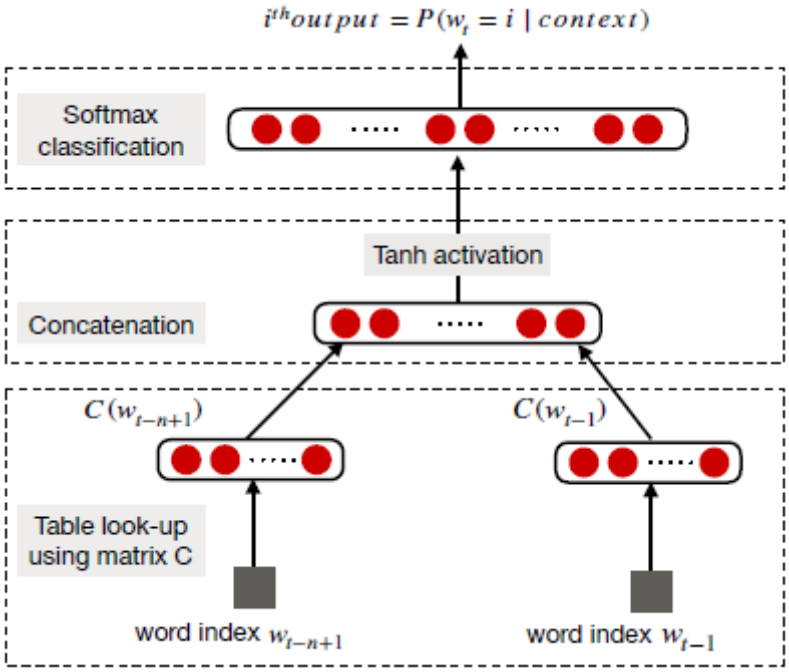
For a long time, the majority of methods used to study NLP problems employed shallow machine learning models and time-consuming, hand-crafted features. This lead to problems such as the curse of dimensionality since linguistic information was represented with sparse representations (high-dimensional features). However, with the recent popularity and success of word embeddings (low dimensional, distributed representations), neural-based models have achieved superior results on various language-related tasks as compared to traditional machine learning models like SVM or logistic regression.

Distributed Representations

As mentioned earlier, hand-crafted features were primarily used to model natural language tasks until neural methods came around and solved some of the problems faced by traditional machine learning models such as curse of dimensionality.

Word Embeddings: Distributional vectors, also called word embeddings, are based on the so-called distributional hypothesis — words appearing within similar context possess similar meaning. Word embeddings are pre-trained on a task where the objective is to

predict a word based on its context, typically using a shallow neural network. The figure below illustrates a neural language model proposed by Bengio and colleagues.



The word vectors tend to embed syntactical and semantic information and are responsible for SOTA in a wide variety of NLP tasks such as sentiment analysis and sentence compositionality.

Distributed representations were heavily used in the past to study various NLP tasks, but it only started to gain popularity when the continuous bag-of-words (CBOW) and skip-gram models were introduced to the field. They were popular because they could efficiently construct high-quality word embeddings and because they could be used for semantic compositionality (e.g., ‘man’ + ‘royal’ = ‘king’).

Word2vec: Around 2013, Mikolav et al., proposed both the CBOW and skip-gram models. CBOW is a neural approach to construct word embeddings and the objective is to compute the conditional probability of a target word given the context words in a given window size. On the other hand, Skip-gram is a neural approach to construct word embeddings, where the goal is to predict the surrounding context words (i.e., conditional probability) given a central target word. For both models, the word embedding dimension is determined by computing (in an unsupervised manner) the accuracy of the prediction.

One of the challenges with word embedding methods is when we want to obtain vector representations for phrases such as “hot potato” or “Boston Globe”. We can’t just simply combine the individual word vector representations since these phrases don’t represent the combination of meaning of the individual words. And it gets even more complicated when longer phrases and sentences are considered.

The other limitation with the word2vec models is that the use of smaller window sizes produce similar embeddings for contrasting words such as “good” and “bad”, which is not desirable especially for tasks where this differentiation is important such as sentiment analysis. Another caveat of word embeddings is that they are dependent on the application in which they are used. Re-training task specific embeddings for every new task is an explored option but this is usually computationally expensive and can be more efficiently addressed using *negative sampling*. Word2vec models also suffer from other problems such as not taking into account polysemy and other biases that may surface from the training data.

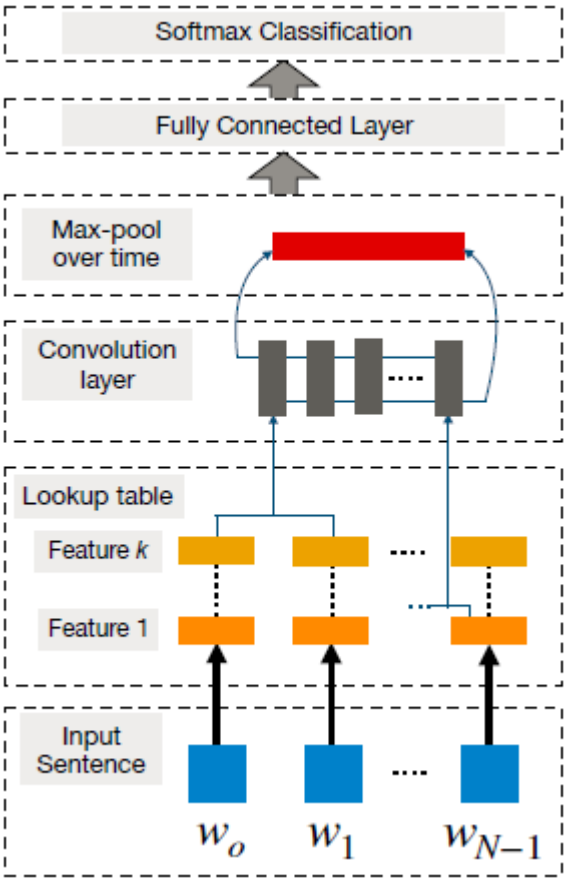
Character Embeddings: For tasks such as parts-of-speech (POS) tagging and named-entity recognition (NER), it is useful to look at morphological information in words, such as characters or combinations thereof. This is also helpful for morphologically rich languages such as Portuguese, Spanish, and Chinese. Since we are analyzing text at the character level, these type of embeddings help to deal with the unknown word issue as we are no longer representing sequences with large word vocabularies that need to be reduced for efficient computation purposes.

Finally, it’s important to understand that even though both character-level and word-level embeddings have been successfully applied to various NLP tasks, there long-term impact have been questioned. For instance, Lucy and Gauthier recently found that word vectors are limited in how well they capture the different facets of conceptual meaning behind words. In other words, the claim is that distributional semantics alone cannot be used to understand the concepts behind words. Recently, there was an important debate on meaning representation in the context of natural language processing systems.

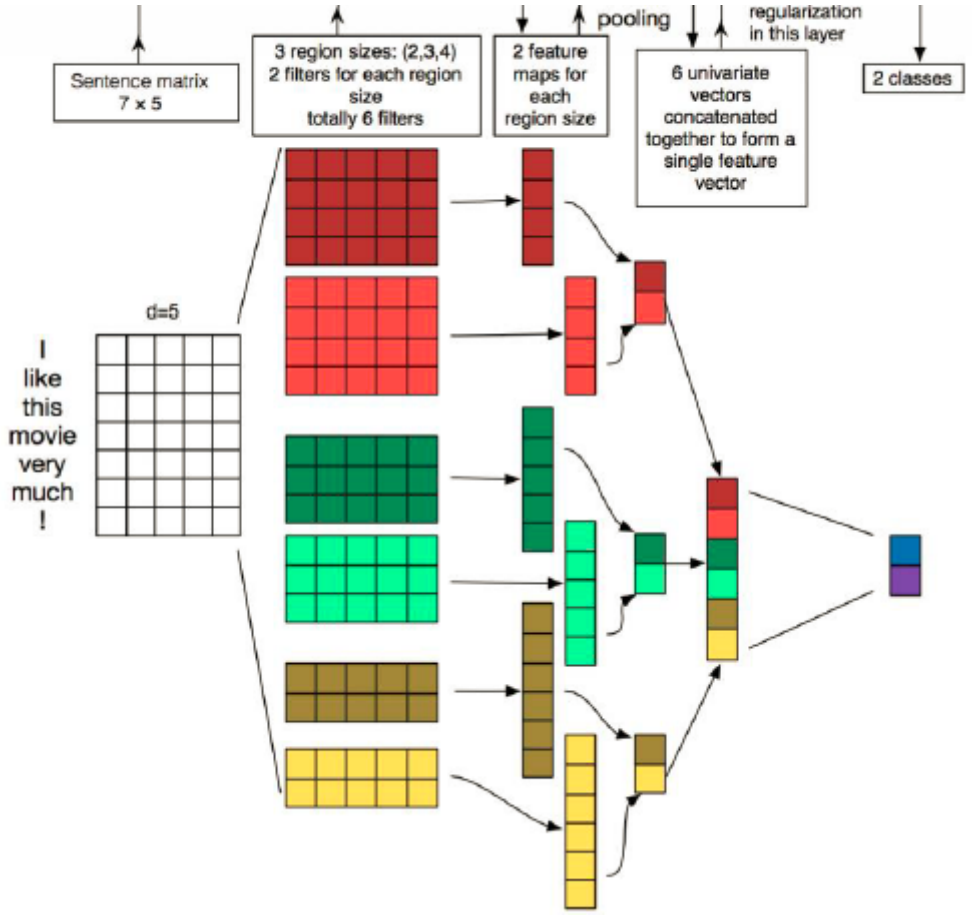
Convolutional Neural Network (CNN)

A CNN is basically a neural-based approach which represents a feature function that is applied to constituting words or n-grams to extract higher-level features. The resulting

abstract features have been effectively used for sentiment analysis, machine translation, and question answering, among other tasks. Collobert and Weston were among the first researchers to apply CNN-based frameworks to NLP tasks. The goal of their method was to transform words into a vector representation via a look-up table, which resulted in a primitive word embedding approach that learn weights during the training of the network (see figure below).



In order to perform sentence modeling with a basic CNN, sentences are first tokenized into words, which are further transformed into a word embedding matrix (i.e., input embedding layer) of *d* dimension. Then, convolutional filters are applied on this input embedding layer which consists of applying a filter of all possible window sizes to produce what's called a *feature map*. This is then followed by a *max-pooling* operation which applies a max operation on each filter to obtain a fixed length output and reduce the dimensionality of the output. And that procedure produces the final sentence representation.



By increasing the complexity of the aforementioned basic CNN and adapting it to perform word-based predictions, other NLP tasks such as NER, aspect detection, and POS can be studied. This requires a window-based approach, where for each word a fixed size window of neighboring words (sub-sentence) is considered. Then a standalone CNN is applied to the sub-sentence and the training objective is to predict the word in the center of the window, also referred to as word-level classification.

One of the shortcoming with basic CNNs is there inability to model *long distance dependencies*, which is important for various NLP tasks. To address this problem, CNNs have been coupled with time-delayed neural networks (TDNN) which enable larger contextual range at once during training. Other useful types of CNN that have shown success in different NLP tasks, such as sentiment prediction and question type classification, are known as dynamic convolutional neural network (DCNN). A DCNN uses a *dynamic k-max pooling strategy* where filters can dynamically span variable ranges while performing the sentence modeling.

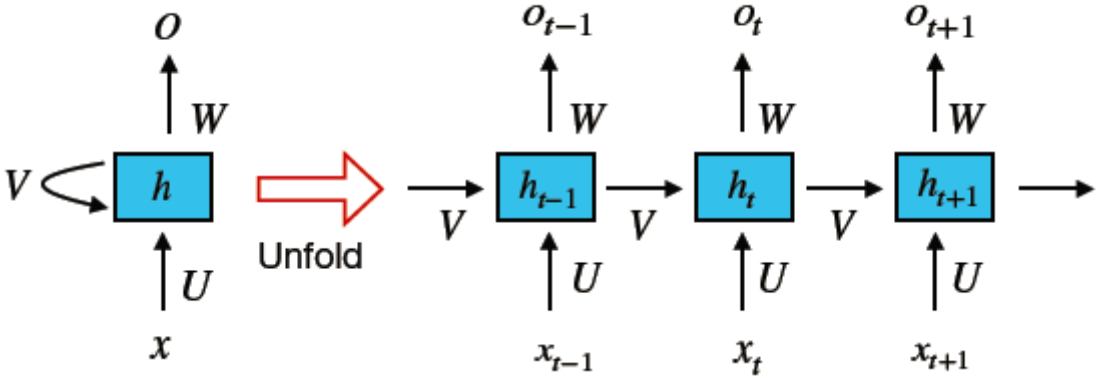
CNNs have also been used for more complex tasks where varying lengths of texts are used such as in aspect detection, sentiment analysis, short text categorization, and

sarcasm detection. However, some of these studies reported that external knowledge was necessary when applying CNN-based methods to microtexts such as Twitter texts. Other tasks where CNN proved useful are query-document matching, speech recognition, machine translation (to some degree), and question-answer representations, among others. On the other hand, a DCNN was used to hierarchically learn to capture and compose low-level lexical features into high-level semantic concepts for the automatic summarization of texts.

Overall, CNNs are effective because they can mine semantic clues in contextual windows, but they struggle to preserve sequential order and model long-distance contextual information. Recurrent models are better suited for such type of learning and they are discussed next.

Recurrent Neural Network (RNN)

RNNs are specialized neural-based approaches that are effective at processing sequential information. An RNN recursively applies a computation to every instance of an input sequence conditioned on the previous computed results. These sequences are typically represented by a fixed-size vector of tokens which are fed sequentially (one by one) to a recurrent unit. The figure below illustrates a simple RNN framework below.



The main strength of an RNN is the capacity to memorize the results of previous computations and use that information in the current computation. This makes RNN models suitable to model context dependencies in inputs of arbitrary length so as to create a proper composition of the input. RNNs have been used to study various NLP tasks such as machine translation, image captioning, and language modeling, among others.

As it compares with a CNN model, an RNN model can be similarly effective or even better at specific natural language tasks but not necessarily superior. This is because they model very different aspects of the data, which only makes them effective depending on the semantics required by the task at hand.

The input expected by a RNN are typically one-hot encodings or word embeddings, but in some cases they are coupled with the abstract representations constructed by, say, a CNN model. Simple RNNs suffer from the vanishing gradient problem which makes it difficult to learn and tune the parameters in the earlier layers. Other variants, such as long short-term memory (LSTM) networks, residual networks (ResNets), and gated-recurrent networks (GRU) were later introduced to overcome this limitation.

RNN Variants: An LSTM consist of three gates (input, forget, and output gates), and calculate the hidden state through a combination of the three. GRUs are similar to LSTMs but consist of only two gates and are more efficient because they are less complex. A study shows that it is difficult to say which of the gated RNNs are more effective, and they are usually picked based on the computing power available. Various LSTM-based models have been proposed for sequence to sequence mapping (via encoder-decoder frameworks) that are suitable for machine translation, text summarization, modeling human conversations, question answering, image-based language generation, among other tasks.

Overall, RNNs are used for many NLP applications such as:

- Word-level classification (e.g., NER)
- Language modeling
- Sentence-level classification (e.g., sentiment polarity)
- Semantic matching (e.g., match a message to candidate response in dialogue systems)
- Natural language generation (e.g., machine translation, visual QA, and image captioning)

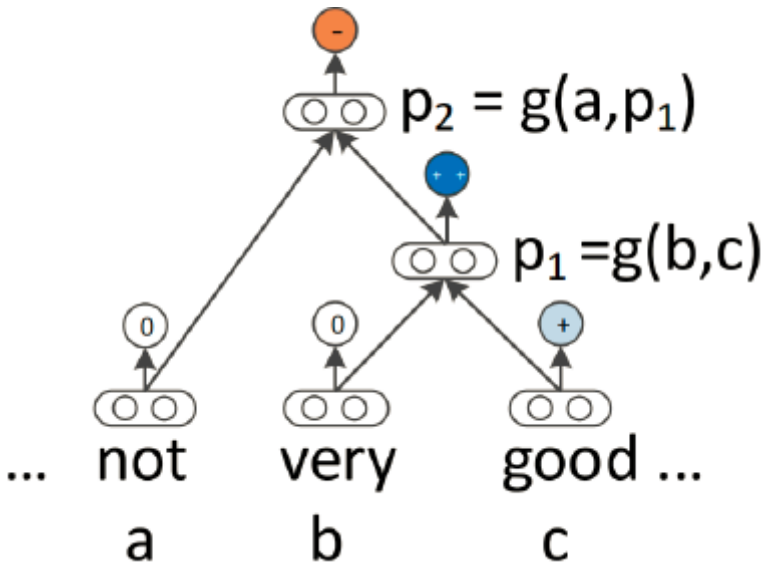
Attention Mechanim

Essentially, the attention mechanism is a technique inspired from the need to allow the decoder part of the above-mentioned RNN-based framework to use the last hidden state along with information (i.e., context vector) calculated based on the input hidden state sequence. This is particularly beneficial for tasks that require some alignment to occur between the input and output text.

Attention mechanisms have been used successfully in machine translation, text summarization, image captioning, dialogue generation, and aspect-based sentiment analysis. Various different forms and types of attention mechanisms have been proposed and they are still an important research area for NLP researchers investigating various applications.

Recursive Neural Network

Similar to RNNs, recursive neural networks are natural mechanisms to model sequential data. This is so because language could be seen as a recursive structure where words and sub-phrases compose other higher-level phrases in a hierarchy. In such structure, a non-terminal node is represented by the representation of all its children nodes. The figure below illustrates a simple recursive neural network below.



In the basic recursive neural network form, a compositional function (i.e., network) combines constituents in a bottom-up approach to compute the representation of higher-level phrases (see figure above). In a variant, MV-RNN, words are represented by both a matrix and a vector, meaning that the parameters learned by the network represent the

matrices of each constituent (word or phrase). Another variation, recursive neural tensor network (RNTN), enables more interaction between input vectors to avoid large parameters as is the case for MV-RNN. Recursive neural networks show flexibility and they have been coupled with LSTM units to deal with problems such as gradient vanishing.

Recursive neural networks are used for various applications such as:

- Parsing
- Leveraging phrase-level representations for sentiment analysis
- Semantic relationships classification (e.g., topic-message)
- Sentence relatedness

Reinforcement Learning

Reinforcement learning encompass machine learning methods that train agents to perform discrete actions followed by a reward. Several natural language generation (NLG) tasks, such as text summarization, are being investigated by employing reinforcement learning.

The applications of reinforcement learning on NLP problems are motivated by a few problems. When using RNN-based generators, ground-truth tokens are replaced by tokens generated by the model which quickly increases the error rates. Moreover, with such models, the word-level training objective differs from the test metric, such as n-gram overlap measure, BLEU, used in machine translation and dialogue systems. Due to this discrepancy, current NLG-type systems tend to generate incoherent, repetitive, and dull information.

To address the problems mentioned above, a reinforcement algorithm called REINFORCE was employed to address NLP tasks such as image captioning and machine translation. This reinforcement learning framework consists of an agent (RNN-based generative model) which interacts with the external environment (input words and context vectors seen at every time step). The agent picks an action based on a policy (parameters) which involves predicting the next word of a sequence at each time step. The agent then updates its internal state (hidden units of RNN). This continues until

arriving at the end of the sequence where a reward is finally calculated. Reward functions vary by task; for instance, in a sentence generation task, a reward could be information flow.

Even though reinforcement learning methods show promising results, they require proper handling of the action and state space, which may limit the expressive power and learning capacity of the models. Keep in mind that standalone RNN-based models strive on their expressive power and their natural capability to model language.

Adversarial training has also been used to train language generators, where the objective is to fool a discriminator trained to distinguish generated sequences from real ones. Consider a dialogue system, with a policy gradient it is possible to frame the task under a reinforcement learning paradigm, where the discriminator acts like a human Turing tester. The discriminator is essentially trained to discriminate between human and machine-generated dialogues.

Unsupervised Learning

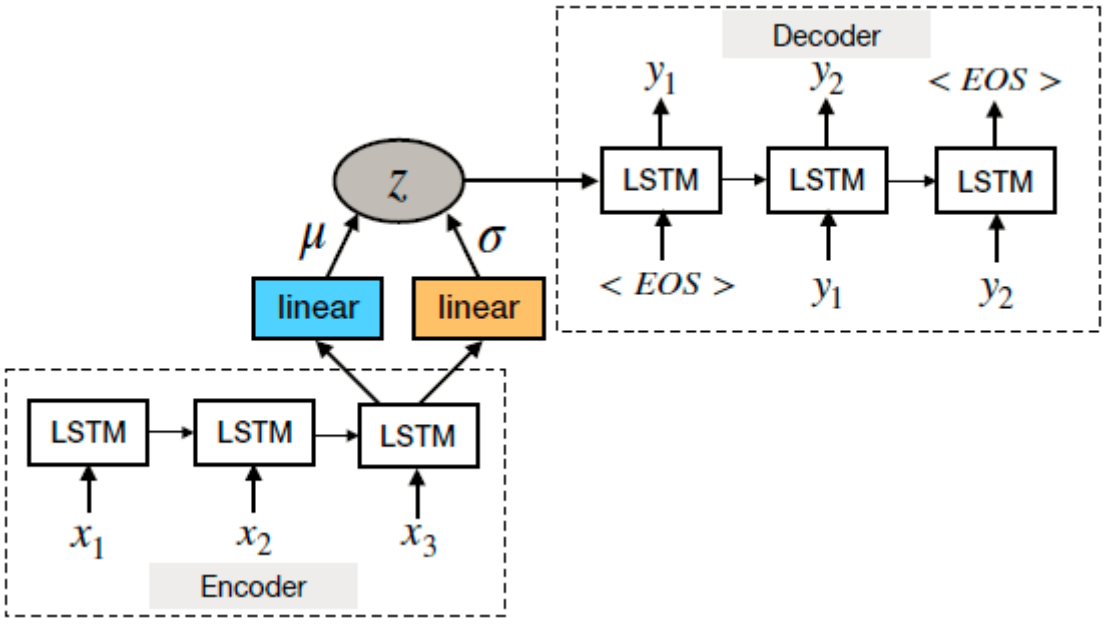
Unsupervised sentence representation learning involves mapping sentences to fixed-size vectors in an unsupervised manner. The distributed representations capture semantic and syntactic properties from language and are trained using an auxiliary task.

Similar to the algorithms used to learn word embeddings, a skip-thought model was proposed, where the task is to predict the next adjacent sentences based on a center sentence. This model is trained using the seq2seq framework where the decoder generate target sequences and the encoder is seen as a generic feature extractor — even word embeddings were learned in the process. The model essentially learns a distributed representation for the input sentences, analogous to how word embeddings were learned for every word in previous language modeling techniques.

Deep Generative Models

Deep generative models, such as variational autoenconders (VAEs) and generative adversarial networks (GANs), are also applied in NLP to discover rich structure in natural language through the process of generating realistic sentences from a latent code space.

It is well known that standard sentence autoencoders fail to generate realistic sentences due to the unconstrained latent space. VAEs impose a prior distribution on the hidden latent space, enabling a model to generate proper samples. VAEs consist of encoder and generator networks which encode an input into a latent space and then generate samples from the latent space. The training objective is to maximize a variational lower bound on the log likelihood of observed data under the generative model. The figure below illustrates an RNN-based VAE for sentence generation.



Generative models are useful for many NLP tasks and they are flexible in nature. For instance, an RNN-based VAE generative model was proposed to produce more diverse and well-formed sentences as compared to the standard autoencoders. Other models allowed structured variables (e.g., tense and sentiment) to be incorporated into the latent code, to generate plausible sentences.

GANs, composed of two competing networks — generator and discriminator — , have also been used to *generate realistic text*. For instance, an LSTM was used as the generator and a CNN was used as the discriminator which discriminated between real data and generated samples. The CNN represents a binary sentence classifier in this case. The model was able to generate realistic text after the adversarial training.

Besides the problem that gradients from the discriminator cannot properly back-propagate through discrete variables, deep generative models are also difficult to

evaluate. Many solutions have been proposed over the recent years but these have not yet been standardized.

Memory-Augmented Network

The hidden vectors accessed by the attention mechanism during the token generation phase represent the model's "internal memory". Neural networks can also be coupled with some form of memory to solve tasks such as visual QA, language modeling, POS tagging, and sentiment analysis. For example, to solve QA tasks, supporting facts or common sense knowledge are provided to the model as a form of memory. Dynamic memory networks, an improvement over previous memory-based models, employed neural networks models for input representation, attention, and answering mechanisms.

Conclusion

So far we now the capacity and effectiveness of neural-based models such as CNN and RNNs. We are also aware of the possibilities to apply reinforcement learning, unsupervised methods, and deep generative models to complex NLP tasks such as visual QA and machine translation. Attention mechanisms and memory-augmented networks are powerful at extending the capacity of neural-based NLP models. Combining all these powerful techniques provide compelling methods to understand the complexity of language. The next wave of language-based learning algorithms promises even greater abilities such as common sense and modeling complex human behaviors.

. . .

NOTICE: Thanks if you made it this far. We also put together an online version of this summary on a dedicated website (nlpoverview.com). Together with the community, we will keep updating the project here.

Reference: "Recent Trends in Deep Learning Based Natural Language Processing" — Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. IEEE Computational Intelligence Magazine, 2018.

Machine Learning Artificial Intelligence NLP Innovation Technology

About Help Legal