**INTRODUCTION**

The field of Natural Language Processing deals with the development of computational models in order to understand the structure and rules of natural human languages[1]. This includes, understanding different relations between words, paragraphs and even documents. This understanding is then used in many different applications. One major field which this kind of learning is impacting is that of artificial intelligence. Developments in this field are in pursuits of giving machines the ability to communicate with humans efficiently. One of the major hurdles for artificial intelligence has been its inability to interface in a language that comes naturally to humans, owing to the fact that human languages have a lot of ambiguity which can’t be overcome by following some predefined rules, rather are learnt with experience. That is where fields like machine learning come in, the basic premise of which is to allow the understanding of different patterns based on experience.

Machine learning is a branch of artificial intelligence that deals with understanding of patterns through historical data. Machine learning learns these patterns by processing huge amounts of data, collected and curated by practitioners and experts of different domains. Machine learning is revolutionising how humans have looked at different fields, and its use is leading to state of the art results. Natural Language Processing is no different. In applications ranging from, text summarisation to POS tagging, machine learning based techniques are changing everything.

But there are certain limitations to classical machine learning that allows it reach only a level of success, after which a plateau is reached [2]. The next level of success is driven by a sub-field of machine learning known by the name of deep learning. Deep learning is the use of neuron like-based structures, as imagined in a human brain to understand much more complex relationships between data[3]. They are the current drivers of change, and are expected to continue bringing in new insights for a long time to come. Techniques like ANNs, CNNs, RNNs are the most important techniques in this field.

This chapter tries to explain in as much detail as possible, the way textual data is represented and pre-processed for efficient computation in general followed by the basic structure of the machine learning process along with the need for deep learning approaches in the presence of conventional machine learning algorithms which have lesser computational and resource requirements; the concepts that form the basic building blocks of Deep Learning - Artificial Neural Networks and Back Propagation, Convolutional Neural Networks or ConvNets with Convolution layers, Pooling layers, Dropout layer, feature detection in CNNs followed by the concepts of Recurrent Neural Networks with GRU and LSTM. Next is a discussion about the major applications of these networks in Natural Language Processing highlighting the state of the art in Machine Learning and Deep Learning for the specific application . The chapter ends with a brief conclusion. The chapter also provides recommendations for subsequent steps to be taken after going through it. This chapter is meant to be an introduction to the vast field of Machine Learning and Deep Learning in Natural Language Processing and aims to be a good first resource for a beginner in the field. The authors advise the readers to consult the resources mentioned along with different topics in the future sections for a better understanding of the field.

**BACKGROUND**

**Word Representation**

Word is the fundamental unit in natural language processing semantic tasks, i.e. tasks that are concerned with and work around meanings of words, and a word happens to be the most basic unit which conveys meaning rather than the characters which they are made up of, which individually mean nothing. Now, words on computers are stored as strings or group of characters, which due to the ambiguity present in all human languages may or may not mean the same things in all different possible contexts, and hence don’t hold deterministic relevance on their own. For example, “*the house is on fire*”, and “*let’s fire the gun*” use the word fire in two different contexts, with two different meanings. While computers have historically dealt with programming languages, they are deterministic in the sense, that words don’t change meaning in different contexts. The word *“function”* is used in pre determined rule based form in a programming language like javascript,  and hence basic logical rules can be used to define different forms of behaviour in a programming language to a machine. As explained above, the basic ambiguous nature of human languages doesn’t allow the same approach to work accurately. Although, there are many words which share characters and hold similar or same meaning, but don’t hold true any generalisation for the complete field, which is dominated by words, which mean the same, but are spelled differently and also, which spell the same, but mean differently based on context. Plus working with strings is computationally expensive[4]. Since strings are based on characters, which themselves are stored as numbers. Individual numbers should be handled much better by machines than a group of numbers to be treated as one in the case of strings, and therefore, better representations are a prime need for faster computation.

**One hot encoded vectors:**

A solution to the inefficient handling of strings was the introduction of one hot encoded vectors [5]. In this, every word in the text or document involved in the application or task at hand, is replaced with a number. This is done by defining a vocabulary of size n which maps all possible words of the language to a number (could also be the corresponding index in the vocabulary). Words derived from the same lemma but may or may not have similar meaning called derivations (for example sing and singer) along with words with the same lemma but with different suffixes or prefixes based on use in different tense or plurality called inflections (for example cat and cats) are given different entries in the vocabulary. The words are mapped to their indices, and that  consequently gives a compact representation to the words. Vectors get involved here, as each word is represented using a n dimensional vector where, the entry corresponding to the index of the word is marked as one and others are marked as zero. Now, the question stands, why are vectors used instead of numbers? The answer to that question lies in the fact that numbers have an inbuilt relation between them, which might mislead algorithms to interpret a relation between two words which are entirely different. Such incorrect learnings are avoided by using vectors, since they do not have any inbuilt relation among them. Another issue that the reader may have noticed is with the sparsity of the current representation of words, which can be dealt with more compact representation of vectors, but another major issue with one hot encodings is that of the inability of the vector to store semantic and contextual relationships between words. It even loses the knowledge of semantic relations owing to similar lemmas. Therefore, a better representation is again required.

**Distributional vectors :**

According to the distributional hypothesis[6], words that are encountered in similar contexts, tend to have similar meanings, making the company of a word, one of its most important characteristic. For example, the words *truthful* and *honest*, both have high probability of holding a company of words like *statement, man, woman & individual*. This concept is used to represent words keeping in mind the contexts they were used in, a problem of one hot encoded vectors. A method based on this kind of learning which goes by the name of “*counting methods*”, rather than storing one at the index of the word in the n dimensional vector, this concept utilises an unsupervised learning technique wherein a window of a predefined size is used. A parser moves word by word and increases the frequency of all the words in the nearby window of the word currently being analysed. This, as the reader can comprehend, involves words both on the right and left of the current word. The vector can be normalised post parsing, and that gives us a probabilistic distribution of different words on how likely they are of being found near to the word, the vector represents. The main idea hence is that similar words, will have similar words used in their neighbourhood, which would lead to similar vector representation for similar words. A very effective method of measuring similarity between vectors is cosine similarity[5], where the cosine is found out using the inner product between the vectors at hand. The closer the cosine is to one, the more similar the vectors are. But the other issue that was present with one hot encodings, of being computationally too expensive due to their n dimensional size still persists in distributional vectors. If n sized matrix is required to represent one word, then n\*n sized matrix is required to represent the entire vocabulary, which is computationally too expensive. This constraint needs to be elevated for more efficient computation.

**Word Embeddings:**

The computation based issues in distributed vectors are resolved using word embeddings[6]. Word embeddings are used to store the same contextual information of the distributed vectors but using a low dimensional vector. The basic idea behind this is to employ an unsupervised learning technique to start with a random valued vector for each word in the vocabulary where the size of the vector can be predefined based on the computational abilities at hand for the user. As each word is parsed, a means can be employed to increase the cosine similarity between vectors encountered near to each other and correspondingly to reduce similarity of the vectors not in the immediate neighbourhood of the word. As more and more scans are done for the same word based on the data at hand, the more accurate this representation gets. Since the amount of non-annotated textual data is enormous, creating word embeddings is not that hard a task. Word embeddings bring the best of both worlds, in word representation that is a very compact representation, avoiding any bias and keeping the context based similarity in tact. One thing to note here is the impact of the vector at hand. The bigger the vector chosen, the more complex the relationship the unsupervised algorithm will be able to learn between the words of the vocabulary. Hence word meanings are represented as vectors which are built keeping intact the con

**Word2vec**:

Created by Google in 2013, Word2vec is an unsupervised method for efficiently learning a word embedding from a text corpus and creating dense vector representations of words in order to capture their semantic and contextual similarity. Usually the size of the word embedding vectors can be specified. Hence, the dimensionality of the vector space in this case is much lower than the sparse vector space obtained using the conventional Bag of words model. The representations of word2vec can be obtained using the model architectures known as Common bag of words (CBOW) model and the Skip Gram model, introduced in the works of Mikolov et al. [7]

**CBOW Mode**l:

Input to this model is the context of each word. By taking the context or surrounding words, it tries to predict the target word corresponding to it. For example, in the given sentence: “The temperature is expected to rise by four degrees.” If we consider a context window of size two, the (context\_window, target\_word) pair for the the target word “four” will look like this: ([by, degrees], four). Implementing the CBOW model [8] involves creating a vocabulary containing all the unique words in a given corpus of text and assigning a unique numerical identifier to each one of them. Building the context and target word pairs is the step that follows. The deep learning architecture takes the context words as input to an embedding layer, which returns their word embeddings. These are averaged out when further passed on to a hidden layer- thereby disregarding the order in which the context words appear. These averaged dense embeddings are passed on to the soft-max layer to predict the target word. The output received is then matched with the actual expected output word to compute the loss, which is back propagated to the initial embedding layer for updation of weights. The process is repeated for multiple epochs.

**Skip-Gram:**

In the skip gram model[9], the neural network is trained to receive an input word (in the form of a one-hot encoded vector) and to predict (in the form of a single vector) the probability of every word in the vocabulary being the neighbouring word based on a parameter known as window size. Training pairs from a corpus are fed into the neural network for training. Common pairs and phrases are hence identified by the number of times each pair shows up in the training data. For example, the probability of occurrence of the word ‘Office’ is significantly higher when the input word is ‘Microsoft’.

**Basic Structure of Machine Learning and the Need for Deep Learning**

Humans learn from experience, whereas machines have historically been operated on hard coded logic. Learning from experience allows humans to perform, tasks presumed very hard for machines very easily which includes effortlessly communicating with each other and navigating around the world using the sense of vision. Being able to allow machines to learn from these experiences is the basic premise behind machine learning.

Any machine learning application involves approximating a function or model which is then used to make predictions or decisions or grouping things together for a given input represented using a feature vector which can be thought to be analogous to generalised properties being used to define somebody [10][11]. This approximation is done on the basis of a data set (set of historically collected feature vectors to define a group of inputs, undergone some basic pre processing to be suitable for processing) collected by experts, practitioners, or researchers. These models can be simple linear equations or composite functions of other non linear functions such as sin, tan, or log, and are basically different weights associated to different features as a measure of information provided by the feature in facilitating the application at hand, which could range from classification or regression. Approximating this model involves majorly two phases, which are named training and testing. Training is the part during which the system (the machine) learns the equation based on a subset of the dataset (randomly sliced, for example if the programmer were to decide that the model was to be approximated using 75% data, then 75% random chosen feature sets will be picked to train on or learn from), starting with random weights for each feature, slowly fine tuning these dependencies to highlight the real patterns and relationships between the feature sets in the dataset. The model approximated in training phase is then tested on the remaining data to see how the model performs on the remaining data that has never been encountered by it before, to really test how well the model performs. The performance is measured by comparing the made predictions with the set of true values marked by field experts or researchers of the field (referred to as ground truth in some texts of the literature concerning machine learning) for all input feature sets for both training and testing datasets. To check the performance of a model, the error or loss function is also used, which is an indicator of the deviation of the model from the true values of the dataset. The minimisation of the error function is one way to move towards a better model.

Generally it is not the case that linear models can be used to reach an acceptable level of accuracy. Classical machine learning learns complex non linear models using a technique of increasing dimensions or features of the feature set by introducing dummy features, which are higher order functions of the current features itself. For example, adding a dummy variable x2 for a given feature x i.e. values of x are squared and put into a separate column for the dataset which is then treated as an individual feature itself and the process of approximating the function is repeated. These dummy features in the dataset are generally added by the researchers or domain experts. This process is heavily dependent on the researcher’s knowledge of the dataset and may not always be correct. It may also happen that the best features for the model to train on are simply not taken into account due to some bias or misconceptions on the expert’s side. In simple words, operating on dummy features to find complex models is a tough ask and researchers look for techniques and algorithms which are able to find appropriate decision boundaries without involving dummy features to the feature set.

This is where the concept of neural network(s) is of utility. They are able to develop complex models for the problem without having to add dummy features into the dataset. Being a major building block in the field of deep learning, this concept has led to a great amount of growth and development in machine learning and artificial intelligence. For a more detailed understanding of this process, the authors recommend [13].

The following section deals with understanding different types of neural networks i.e. ANNs (Artificial Neural Networks), CNNs (Convolutional Neural Networks), RNNs (Recurrent Neural Networks) which are the major basis of understanding deep learning and how it impacts different areas of natural language processing

**Artificial Neural Network and Back Propagation**

A neural network[14] can be regarded as a group of individual systems or models which when combined in a layered architecture or structure, can be used to define a composite system with a very complex boundary for a given dataset, while also avoiding the model from overfitting. Overfitting is the concept where the model approximated using the training set performs very well on the training data but performs significantly worse on encountering new test data. More information about overfitting can be found here []. This structure is modelled on the neural network structure of the brain which is a complex and composite structure of elementary units called neurons, which are said to use weighted impulses. The initial layer of this architecture is called the input layer, and the final layer involved in giving the output for the problem is called the output layer. The layers in between the input and output layers are referred to as hidden layers. The neurons in one layer are connected to all neurons in the layer ahead, and each neuron has a corresponding output function which when fed an input (generally the weighted sum of outputs of the neurons connected to it), outputs a corresponding value when a function referred to as the activation function is applied to it. An example of a simple neural network is shown in Fig. 1.

ANN main-1.tiff

Figure 1. Artificial Neural Network

The weighted sums of the outputs of the units in the previous layer connected to a neuron in the current layer, depends on the weights given to the respective output of the previous layer neurons. These weights together decide the final model of the system and these need to be tuned in order to increase the accuracy or to decrease the loss associated to a model.

The working of a neural network is basically development of different linear or non-linear models in the neurons of the first hidden layer, the linear combination of which can be used to come up with more complex models, which would be higher order non-linear function and this process goes on as hidden layers are added to the architecture of the output layer which depending on the type of the classification i.e. binary or multi, gives an appropriate output increasing complexity in the network which allows approximation of a much more complex function. This has a performance trade off as well which is that since all neurons in one layer are connected to the neurons in the next layer, therefore there are a lot of weights to tune which requires a lot of computation power, and slows down performance, which needs to be kept in mind while training models.

Now, the process of outputting for an input feature set is called forward propagation and requires a single traversal of the network. As mentioned before, these outputs require weights on the connections between the neurons. The network starts with random weights for the connections and then the amount of error being induced in the network due to the random weights using a loss or cost function, is evaluated. If the error being induced in the network can be reduced to a satisfactory level, then it can be said that the system is optimisable. The error is reduced using the process of back propagation. This process involves the updating of weights of the network by evaluating the dependency of the error on the weight, for which the derivative of the error function with respect to the weight is calculated[15]. Once this dependency is evaluated, the weight is updated by subtracting the derivative times the learning rate. This process of gradually tuning parameters in the direction of maximum error reduction or minimum loss is called gradient descent. The process of finding the derivative holds the key to the whole process of back propagation. The basic premise behind the process of back propagation is explained here as depicted in Fig 2.

ANN.tiff

Figure 2. Artificial Neural Network for Back Propagation

The derivative of the error function with respect to any weight can be rewritten as the derivative of error with respect to the net output of the network multiplied by the derivative of the output with respect to the weights. This derivative of the output with respect to the weights can be written as the derivative of the output with respect to the input multiplied with the derivative of the input with respect to the specific weight.

Note : wij refers to the ith unit in the current layer, connected to the jth unit in the layer ahead.

The derivative of the error function (assuming error function to be mean squared) with respect to the output is where the summation is over the whole dataset is given by :

The derivative of the final output (considering the activation function to be sigmoid) with respect to the input is :

The derivative of the input to the final output for a specific weight is :

The product of these three factors, gives the derivative dependency of the error on the weight and then the updating can happen which will allow the error to recede, increasing the accuracy of the model.

Note : The model considered here is comparatively simple than most models encountered in machine learning every day, but should be enough to give an intuition of back propagation. For readers looking for more detailed explanation for complex models, authors recommend [16].

An artificial neural network is very useful in classification problems and will be used extensively in further type of network architectures (CNNs and RNNs) as well due to this utility.

**Convolutional Neural Networks**

While working on images, the type of processing discussed in artificial neural network will treat each pixel as a different feature. Pixels in the same neighbourhood in images are known to have similar pixel values, and these similar valued pixels together form different shapes and curves and this correlation is not accounted for [17]. The human brain on which the ANN is based, accounts for these correlations and forms a suitable boundary to recognise objects with a combination of pixels processed together as is done in the visual cortex of the brain.These correlations can be detected using different image processing techniques, one of the most important of which is convolution, which involves applying a mask over the image to come up with a modified image. It is based on the principle that correlation can be attributed to a combination of pixels i.e. being able to form a super pixel from a combination of sub pixels. To form this super pixel, it is necessary to find a logical combination of the sub pixels. One such combination can be the weighted sum of these values, since the contribution of one pixel will be more than the other in deciding the value of the super pixel. The net should be able to learn these weights itself as it does in ANN. These weights should be assigned in such a way that it is able to identify patterns and shapes in the image. This is the basic concept and strength of Convolution Neural Networks (abbreviated ConvNets) i.e. given an image it should be able to output the correlated features in an image which can then be fed into an ANN which can then classify the images as per requirements of the system.

Convolutional Neural Networks are essentially deep artificial neural networks that find applications in the fields of Object Detection, Object Detection and Localisation, Image Classification, Scene Labelling, Video Activity Recognition, etc. ConvNets are well qualified for image processing tasks.

*The convolution operation*

A grayscale image is represented as a two dimensional matrix. If a filter is applied to the image, the input image is said to be convolved with the filter matrix. The filter may be a vertical edge detector - to detect all vertical edges in the given image - for example, a 3 x 3 vertical edge detection matrix is shown in Fig. :

Edge Detector.tiff

Figure 3. Filters for (a)vertical and (b)horizontal edge detection

The filter may be a horizontal edge detector, like the one shown in the following or a filter to detect edges at 45 degrees or 70 degrees.

On an input matrix of size 4 x 4, the convolution of the 3 x 3 vertical edge detection matrix is illustrated below in Fig..

Conv Filter Combined.tiff

Figure 4. Convolution operation over the given matrix

The steps involve multiplying each element of the filter matrix (shown in superscripts) with the input matrix elements that it overlaps with and storing the sum as the result in the corresponding position.

In step 1, the result is obtained as 0(1)+45(0)+4(-1)+18(1)+0(0)+21(-1)+5(1)+32(0)+0(-1) which is equal to -2, which forms the first element of the result matrix. Similarly, the result has been computed for the other three steps.

Conv Results Comibed.tiff

Figure 5. Corresponding results of the convolution operation

For a greyscale input image of size n x n convolved with a filter matrix (or convolution matrix) of size f x f, the resulting image is a matrix of size : (n-f+1) x (n-f+1). Applying the formula on the above example where n=4, f=3, the resulting matrix is of size (4-3+1) x (4-3+1) or 2 x 2.

*Padding*

As observed in the formula given above, the output image keeps shrinking as filters are applied, disregarding information from the edges of the input image. In such cases, it may be useful to pad the image with an additional border of say, p pixels so that the output image is now of the dimensions: (n+2p-f+1 x n+2p-f+1). This allows taking all pixels into consideration equal number of times while evaluating the convoluted image.

greyscale padding.tiff

Figure 6. Padding applied to a matrix

Based on the padding, convolutions may be classified into:

(a) **Valid convolutions:** No padding is done on the input image (p= 0). The n x n image is convolved with an f x f filter to give an output image of size n-f+1 x n-f+1.  
(b) **Same convolutions:** Padding is performed in a manner that the output size is the same as the input size. So, n+2p-f+1 = n => p= (f-1)/2 and since filters are usually odd number matrices for most practical applications, the value of padding comes out to be a whole number.

*Stride*

In the above example, for every subsequent computation, the filter shifts by one column. For such cases, the stride (denoted by s) is 1. Given a stride of 2 (s=2), the filter shifts by two columns in the next computation. So, for an n x n image, with a padding of p, an f x f filter and a stride s, the dimensions of the final image will be

*floor* {((n+2p-f)/(s+1))  x ((n+2p-f)/(s+1))}

For computation, the filter must lie within the image or the (image + padding)

*Convolution over volume*

For most practical applications these days, the input to a convolutional neural network is a multi-channelled image comprising of three channels - RGB. For example, the dimensions of the image may be 4 x 4 x 3, where 3 corresponds to the Red, Green and Blue colour channels. While the height of the image is 4 pixels and the width is 4 pixels. This input image can be visualized as a stack of three 2D images of size 4 x 4. It is also important to note that the number of channels in the filter are taken to be the same as the number of channels in the input image.   
  
If dimensions of the image are n x n x nc and dimensions of the filter are f x f x fc, then the resulting image is of the dimensions: (n-f+1) x (n-f+1) x nc’ (*or number of channels (nc) of the next layer, which is just equal to the number of filters applied*).

*One layer of a CNN*

The image at hand is convolved with n filter matrices and the output is summed with a bias (for each entry in the matrix). This is passed into an activation function like ReLU, resulting in the final matrix. All these operations together form the convolution layer of CNN.

Combined Pooling.tiff

Figure 7. (a) Average Pooling and (b) Max Pooling

The Convolution layer, pooling layer and fully connected layer constitute the basic building blocks of a CNN. Other than convolution layers, ConvNets often use pooling layers to reduce the spatial size of representation in order to reduce the number of parameters and computation in the network. It also makes some of the features it detects more robust. The popular types of pooling are:

1. **Max pooling:** For each of the grids highlighted in the vector below, the maximum value is selected as output. Pooling does not have to learn parameters. Instead, it has a fixed set of hyperparameters and fixed calculations. On the other hand, a large number of parameters exist in fully connected layers.
2. **Average or mean pooling:** The input is divided into smaller regions and average values of each region are computed.

*Dropout layer*

Convolution networks also employ a layer by the name of dropout layer to increase the randomness factor in the network to avoid overfitting. This layer exists between the dense and output layer, allowing only certain units to go through depending on some probability factor.

Convolutional layers have two major advantages over fully connected layers. One is parameter sharing as stated before is based on the idea that a feature detector useful in one part of the image is applicable as it is, to another part of the image - which is why the same 3 x 3 filter matrix can be applied at different positions of the input image. The second advantage is its sparsity of connections. The output values in each layer depend on only a small number of inputs.

**Recurrent Neural Networks**

For many problems, input data is sequential or has a factor of context associated with it. For example, sequence of words, audio signals or videos. But till now, the discussion has consisted of models that do not take this factor into account i.e. the models are provided input in one go and they generate outputs in one go without taking into consideration the temporal context between the features. Examples of such problems include music generation, named entity recognition, machine translation, speech recognition, video activity recognition, etc.

Depending on the application, both the input and the output can be variable sized, with a sequential nature to them. The basic idea behind RNNs [18] is that when a unit receives some input feature, it stores the information corresponding to it and that information is used while processing the next input feature and all future input features i.e. RNNs have a sort of memory element associated to it which allows the network to keep track of a context while processing features encountered at different time steps. RNNs are famously known to be able to learn shared features across different positions in the sequence. There are three different types of units that are used in RNNs with different capabilities and varying complexities i.e. the basic unit, the Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM). GRUs and LSTMs are based off the basic unit of RNNs, and these basic units are instrumental in understanding RNNs. RNNs generally operate on vectors.

*The basic unit*

RNN base4.tiff

Figure 8. Basic RNN unit

The features[19] are inputted to the basic unit in a sequential order, where an activation is calculated for each feature at different time steps in the unit, by applying an activation function on the multiplication of the input feature and the weights summed to the activation from previous time step. The output for each time step in the unit may be calculated by applying another activation function on the activation obtained by using the unit. In some applications the output might be taken after processing all features of the input, and some might output parallel to the input features being processed. This output is referred to as y’ henceforth, and the activation at a certain time step referred to as at . To maintain consistency an activation is also added at the 0th time step which has a vector of zeros.

An obvious weakness of such an approach is that it only takes into consideration features of previous time step into consideration while making a decision for the current input feature which might not always be the best approach for a problem. For example named entity recognition.

*He said, “Rose is the best player in the NBA”.*

*He said, “Rose is the best gift for Ms. Anita’s birthday”.*

The process of back propagation here also deals with tuning the weights so as to minimise the loss function as much as possible. The loss function is taken as the aggregate of all time steps. Another problem that RNNs face is of vanishing gradients. Here the gradient refers to the gradient of the loss function with reference to the weights, calculated at the time of back propagation, which are used to update the weights. Vanishing gradients refers to the gradient of the loss function with reference to weights in the earlier layers becoming very small, due to the involvement of derivatives which can take very small values, and as discussed before, the updating of individual weights involve product of different derivatives. These can get very small for the earlier layers, and the weights don’t really change much and their hit on the error doesn’t reduce, making it feel as though the network is stuck. This is avoided by the use of other type of units of RNN i.e. GRUs and LSTMs.

*Gated recurrent unit*

GRU unit.tiff

Figure 9. GRU unit

GRU[20] is a modification to the basic unit of RNN, which makes it much better at capturing long range connections in the sequence. An example of long range connections is as shown below.

*The baby, which already slept — is awake*

*The babies, which already slept — are awake.*

There could be many words between slept and the helping verb before awake, so it could be important to remember the context that the baby was singular in the first sentence whereas it was plural in the second example to be able to make prediction of possible words in different language models.

GRUs have a memory unit referred to as c, which, as the name suggests, provides memory to remember context to the RNN units. In GRUs the content of the memory cell and the activation are same. Like in the example considered before, 1 can be stored in c if the word to take into context is plural or 0 if it is singular, and till it does not go out of context, it can be maintained in the memory cell, and once it does (depending on the rules defined by the designer of the model), it can be considered to overwrite it with a different context depending on the local active context. That is the basic principle behind GRUs. At each time step the model considers overwriting, the value of the memory cell with a value of c’. c’ refers to the candidate value that may replace the current value of the memory cell. The GRU also consists of a gate which decides whether the candidate value will actually replace the current value of c, or not. The equations involved in GRUs are as follows :

*Long Short Term Memory Unit*

LSTM unit.tiff

Figure 10. LSTM unit

The LSTM[21][22] can be considered an extension of the GRU where the activation and memory cell values are not taken to be same. Here the candidate value is calculated using the activation value as shown in the following equations.

The unit consists of two other gates other than the update gate i.e. forget gate and the activation output gate. The update gate provides weight to the new candidate value to be able to replace the older value, whereas the forget gate gives weight to the old candidate whether it should be forgot or not. Their dependency together decide the value of the memory cell in the current time step. The output gate operates on the activation from the previous time step, which is used in deciding the activation for the next time step. This whole relationship is explained in the equations as shown above.

*Bidirectional RNNs*

Bidirectional RNNs[23] solve the issue where the model was operating on current input by taking into consideration only the previous time steps’ input features. The activation from both the previous and future time steps are taken into consideration while deciding the activation for the current time step unit i.e. the model propagates from both directions. The major constraint with bi-directional RNNs is that the entire sequence is required before operating, which might not be appropriate for real time applications of RNN.

**APPLICATIONS OF MACHINE LEARNING & DEEP LEARNING IN NLP**

**Named Entity Recognition**

*High Level Definition*

Named Entity Recognition (NER) classifies the named entities present in the text into categories like places, individuals, organisations, dates etc.  which can also be used to understand the subject of any given article. NER can scan news articles and identify the relevant tags like people, places or organisations talked about in the article. This is done to ensure smooth discovery of content on news applications. To create efficient search algorithms, NER can be run on the relevant tags/entities instead of searching every word of every article. This is used by online journals or publication sites that host multiple research papers. For a company handling complaints on social media platforms, the text can be passed through a NER API, to filter out the location and the product name so it can be immediately assigned to the relevant department.

*State of the art*

The CoNNL 2003 has been used as the standard English dataset for NER which mainly focusses on four types of entities: people, locations, organisations and miscellaneous. Support Vector Machines used and randomised condition based fields have been employed for the same. In 2011, the works of Collobert et al. [24] used a multitask approach to solve NLP problems. Their work proposed a feed forward neural network - which operated on pre-processed text output of word embeddings having a context with a fixed size window around each word. The fast architecture enabled the authors to employ a database as large as 631 million words from Wikipedia and demonstrated that the network could be used for several tasks including NER, POS tagging, chunking etc. The work demonstrated that generalised performance can be improved by learning these tasks simultaneously. For Named Entity Recogniton, the work of Hammerton, 2003 [25] proposed the Long Short term Memory.

Dos Santos et al. [26] proposed a Named Entity Recognition System that was language independent. The model used was known as the CharWNN deep neural network (an extension of Collobert’s multitask network with an added convolution layer to extract character level embeddings) with word & character level embeddings generated again from pre processed data with an appropriate technique as input in order to perform classification on Portugese text contained in the HAREM I corpus and Spanish text in the SPA CoNLL 2002 corpus which was sequential in nature. They concluded that alone, neither word based embeddings nor character based embeddings could match the results obtained when these two are used jointly. Across the ten named entity classes, the proposed CharWNN model outperformed the state-of-the-art system by reaching the benchmark in F1 score of 7.9 points as well as achieving state of the art performance on the SPA CoNLL 2002 corpus for Spanish texts.

The work of Chiu and Nicols [27] presented a neural network based architecture to detect word and character state features without supervision. They used a hybrid Bi-directional LSTM based Network and CNN to replace the requirement for feature engineering. Lookup tables are used to convert discrete words and characters into continuous vector rrepresentations, which form the input to the neural network.

The work of Lample et al. [28] introduced bidirectional LSTM accompanied by random fields based on conditions to operate in a neural net architecture that labels segments inspired by the shifit-reduce parser. Without the use of any language specific expertise or external labelled resources or gazetteers, the models achieved state-of-the-art performance in Named Entity Recognition on four languages. The model used both character level inputs and word embeddings, which served as inputs to the bidirectional LSTM. The output of the bidirectional LSTM were fed to  a layer which performed CRF calculations.

**Text Classification**

*High Level Definition*

The ability to automatically classify text documents into specific categories is known as text classification. This could be categorising news articles or blogs into specific domains, making articles easier to find and sort. In supervised Machine Learning models, the training data set is pre labelled into categories and a classifier is trained on the dataset to enable it to learn and classify a new article from then onwards.

*Implementation*

In 1971, the Rocchio algorithm was proposed which was based on a method of relevance feedback found in information retrieval systems. A major limitation of the Rocchio model was that it failed to classify multimodal classed and relationships as the words having a similar origin may lie far apart in the vector space. Support Vector Machines and Decision Trees also emerged as useful methods for document classification.

*State of the art*

Kim [29] used CNNs to perform multiple experiments in order to better understand and analyse sentence classification tasks. The model comprised of a convolution layer with a soft-max layer followed by a dropout layer. The nets were trained on pre trained word vectors, while understanding task based vectors specifically and performing some fine tuning on the hyper parameters involved in the network enabled further performance improvement. The model proposed in the works of Kim was able to outperform the benchmark of the time in four text classification tasks out of seven, which also included question segregation and sentiment detection.

Conneau et al. [30] presented an architecture of VD CNNs or Very Deep Convolutional Neural Networks which uses only convolutions and max pooling operations with a window size of three at the character level. Very deep CNNs were applied to text processing tasks for the first time. On using upto 29 convolutional layers on eight large-scale public datasets, there was visible improvement in several text classification tasks.

In Jiang’s work on text classification [31] a hybrid model for text classification was proposed. It was based driven by deep belief networks accompanied by soft-max regression algorithm. Deep Belief Networks are essentially feedforward networks where hidden layer pairs resemble restricted Boltzmann machines. The deep belief network helped resolve the issue of sparse high dimensional matrix arithmetic of text data in feature extraction and soft-max regression was used to classify the text. After separately pre-training both parts - Deep belief Networks and Soft-max regression, they were combined and trained together like a deep neural network. Using few labelled samples- which were less than half of the training data, the algorithm was able to achieve an accuracy that was 8.51% higher than SVM on the entire set of training data.

**Text Summarisation**

*High level Definition*

Shortening a long piece of text to create a meaningful summary of only the main points in the document is known as text summarisation.

*State of the art*

Frequency based extraction technique had been proposed along with Naïve Bayes Algorithms and graph based algorithms. The works of Erkan and Radev [32] proposed a stochastic measure to calculate the relative importance of units of text in a document. They presented an approach known as LexRank for computing sentence importance.

Abstractive text summaries, studied by Ganesan et al. [33] presented a graph-based summarisation technique to create summaries of opinions considered to be highly expendable. Compared to baseline abstractive methods, these summaries were in better agreement with the human summaries. They were well formed and conveyed the necessary information from the article.

The work of Liu et al. [34] presented an abstractive summarisation method wherein the source text is parsed to form Abstract Meaning Representation (AMR) graphs- which are semantic graphs of input, which are subsequently transformed into a graph like structure meant to depict summary later used to generate text summary. In recent times, encoder-decoder architectures in deep learning methods are being employed for abstraction based text summarisation.

Rush et al. [35] presented an approach based on data for summarisation of sentences. The method utilises a local convolutional attention-based encoder and generative beam search decoder to generate each word of the summary, giving due consideration to context.

A deep reinforced model was introduced for abstractive summarisation by Paulus et al. [36] which was a Recurrent Neural Network based encoder-decoder with an attention mechanism. It achieved successful results when the input text was short. For longer texts, on the other hand, it was seen that the model produced some incoherent sentences. The training method involved supervised learning along with reinforcement learning. When evaluated over the CNN/Daily Mail and New York times datasets, the model was seen to outperform previous state-of-the-art works. Gehring et al. [37] presented a convolutional model with boosted accuracy owing to an attention mechanism at each  layer.

**Question Answering**

*High Level Definition*

An important application in the field of information retrieval and natural language processing, which enables the system to automatically answer questions posed by a user. This would involve the system’s ability to translate natural language sentences used in the queries  into a representation that is internal to the system. The Stanford Question Answering Dataset or SQuAD is an open comprehension dataset of questions asked on Wikipedia articles and the answers are sentences from the article itself. Another publicly available annotated dataset is the WikiQA that contains questions and answers. The NewsQA and TREC-QA are similar datasets for research work.

*Implementation*

Two modern day classifications of the solutions to Question Answering are- IR based factoid question answering and Knowledge based question answering. In the IR based factoid Question answering the question is analysed to extract key information and a relevant short segment of text or document is pulled out from the web. In knowledge based Question Answering, a structured database is utilised to map questions in natural language to a query. Ambiguity in words and phrases, multilingualism in web documents pose limitations to this task.

*State of the art*

Early approaches to Question Answering in the 1960s involved parsing the question and matching the same tokens in the documents retrieved. Current approaches to the problem involve methods similar to those used in text summarisation. Wang et al. [38] presented a gated attention based recurrent network and a self-matching attention mechanism for reading comprehension style Question Answering to present answers to a question from a given passage. In order to locate the position of the answers from the given passage, pointer networks were used. For single and ensemble model, their work marked the top position on the leaderboard of the SQuAD dataset with a 71.3% on the single model and 75.9% on the ensemble model. To understand questions, a Multi Column Convolutional Neural Network (MCCNN) was used in the work of Dong et al. [39]. Instead of relying on hand crafted features for an understanding of questions or ranking most appropriate answers. The system has also been trained on paraphrased questions. Raposo et al. [40] introduced Relation Networks or RNs, which are Multilayer Perceptron networks with a focus on object-relation reasoning, that is learning relationships among objects or entities in the data. Long Short Term Memory or LSTM representations of sentences of a document were fed as input to the Relation Network, which considered all permutations to arrive at relationships among the sentences of a document or the question and the sentences.

**Machine Translation**

*High level Definition*

The process of converting one natural language into another using mathematical and algorithmic techniques while preserving the meaning of the text given as input is known as Machine Translation. Even for humans, the task of translating a document or piece of text from one language to another requires a great level of expertise in both languages involving a good understanding of the syntax, context of the written piece and the semantics.

*Implementation*

The different phases or subtasks involved in the process are-

De-formatting the source text to remove all non-translatable items like diagrams etc. Once the entire text has been translated into another language, the document is reformatted to add these items back in order. Another step of pre-editing or pre-processing involves removing punctuation marks that do not require translation and possibly cutting long sentences short for ease of understanding and translation of the text.

Morphological Analysis: This is used to determine the tense, POS tag and form of the base word. A distinction between the subject and the object in a sentence is achieved through syntactic analysis and an understanding of the context and interpretation is known as semantic analysis.

In order to explain the basic structure of a language, grammar formalism is utilised, which is a basic framework provided by linguists and researchers to explain the structure of a language.

Identifying the linguistic properties of all the words in the text document is tagging and going through the text while understanding the relation among words is known as parsing.

A machine translation system usually has an analyser which produces an internal representation of the meaning of the given input text. With this given meaning, the other component- referred to as the synthesiser produces output text containing one or multiple sentences using the meaning it receives from the synthesiser.

*State of the art*

In the early 2000s, statistical approaches were employed by researchers in the process of word based translation, syntax based translation or phrase based translation. With the use of end to end Deep Neural Networks being used for the task, results have shown massive improvement through the use of convolutional and recurrent layers to convert source language into target language, while researchers like K Ahmed et al. [41] advocate removing the recurrent and convolution layers entirely to use self attention and feed forward layers only. The limitation with the proposal is that it needs a huge number of training iterations to converge but it has successfully achieved state of the art performance on several MT tasks and surpassed the state of the art in English to German and English to French translation. The proposed weighted transformer in their work outperformed the baseline and converged at a rate  15 to 40% faster in comparison.

**CONCLUSION**

In this chapter, the reader got an overview of the ways in which machine learning and deep learning are impacting the field of natural language processing. Even though machine learning and deep learning have brought in state of the art results, there is still a long way to go before real time human like communication can be said to be possible for machines. There are multiple issues at hand, one of which is the ever growing corpus of words in human languages due to the advent of social networks, which lead to the growth of informal forms of languages, which derail a lot of work already done. Another constraint is the amount of computational power as well time required to be able to conduct research in this field. Even after so many improvements in the field for computation of strings, there is still some distance to cover. But the road ahead, looks both hopeful and exciting at the same time, thanks to the advances in fields like machine learning and deep learning.

**MOVING FORWARD**

The authors recommend the readers to study deep learning in detail understanding the implementation of different famous frameworks like TensorFlow, Keras and PyTorch which make neural networks much more accessible. The authors also recommend going through other applications which have not been discussed in detail in this chapter such as POS tagging, Image and Video Captioning among many others., resources to which have been provided in the additional reading section.