Natural Language Processing: From Basics to using RNN and LSTM

A detailed introduction to all the concepts prevalent in the world of Natural Language Processing

vibhor nigam Follow
May 17, 2019 · 11 min read ★

One of the most fascinating advancements in the world of machine learning, is the development of abilities to teach a machine how to understand human communication. This very arm of machine learning is called as Natural Language Processing.

This post is an attempt at explaining the basics of Natural Language Processing and how a rapid progress has been made in it with the advancements of deep learning and neural networks.

Before we dive further into this it is necessary to understand the basics

What is Language?

A language, basically is a fixed vocabulary of words which is shared by a community of humans to express and communicate their thoughts.

This vocabulary is taught to humans as a part of their growing up process, and mostly remains fixed with few additions each year.

Elaborate resources such as dictionaries are maintained so that if a person comes across a new word he or she can reference the dictionary for its meaning. Once the person gets exposed to the word it gets added in his or her vocabulary and can be used for further communications.

How do computers understand language?

A computer is a machine working under mathematical rules. It lacks the complex interpretations and understandings which humans can do with ease, but can perform a complex calculation in seconds.

> *For a computer to work with any concept it is necessary that there should be a way to express the said concept in the form of a mathematical model.*

This constraint highly limits the scope and the areas of natural language a computer can work with. So far what machines have been highly successful in performing are classification and translation tasks.

A classification is basically categorizing a piece of text into a category and translation is converting that piece into any other language.

What is Natural Language Processing?

> *Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software.*

The study of natural language processing has been around for more than 50 years and grew out of the field of linguistics with the rise of computers.[1]

Basic Transformations

As mentioned earlier, for a machine to make sense of natural language(language used by humans) it needs to be converted into some sort of a mathematical framework which can be modeled. Below mentioned, are some of the most commonly used techniques which help us achieve that.

Tokenization,Stemming and Lemmitization

Tokenization is the process of breaking down a text into words. Tokenization can happen on any character, however the most common way of tokenization is to do it on space character.

Stemming is a crude way of chopping of an end to get base word and often includes removal of derivational affixes. A **derivational affix** is an **affix** by means of which one word is formed (derived) from another. The derived word is often of a different word class from the original.The most common algorithm used for the purpose is Porter's Algorithm.

Lemmatization performs vocabulary and morphological analysis of the word and is normally aimed at removing inflectional endings only. An **inflectional ending** is a group of letters added to the end of a word to change its meaning. Some **inflectional endings** are: -s. bat. bats.

> *Since stemming occurs based on a set of rules, the root word returned by stemming might not always be a word of the english language. Lemmatization on the other hand reduces the inflected words properly ensuring that the root word belongs to english language.*

N-Grams

N-grams refer to the process of combining the nearby words together for representation purposes where N represents the number of words to be combined together.

For eg, consider a sentence, "***Natural Language Processing is essential to Computer Science.***"

A **1-gram** or **unigram** model will tokenize the sentence into one word combinations and thus the output will be "**Natural, Language, Processing, is, essential, to, Computer, Science**"

A **bigram** model on the other hand will tokenize it into combination of 2 words each and the output will be "**Natural Language, Language Processing, Processing is, is essential, essential to, to Computer, Computer Science**"

Similarly, a trigram model will break it into "Natural Language Processing, Language Processing is, Processing is essential, is essential to, essential to Computer, to Computer Science" , and a n-gram model will thus tokenize a sentence into combination of n words together.

> *Breaking down a natural language into n-grams is essential for maintaining counts of words occurring in sentences which forms the backbone of traditional mathematical processes used in Natural Language Processing.*

Transformation Methods

One of the most common methods of achieving this in a bag of words representation is tf-idf

TF-IDF

> ***TF-IDF is a way of scoring the vocabulary so as to provide adequate weight to a word in proportion of the impact it has on the meaning of a sentence. The score is a product of 2 independent scores, term frequency(tf) and inverse document frequency (idf)***

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

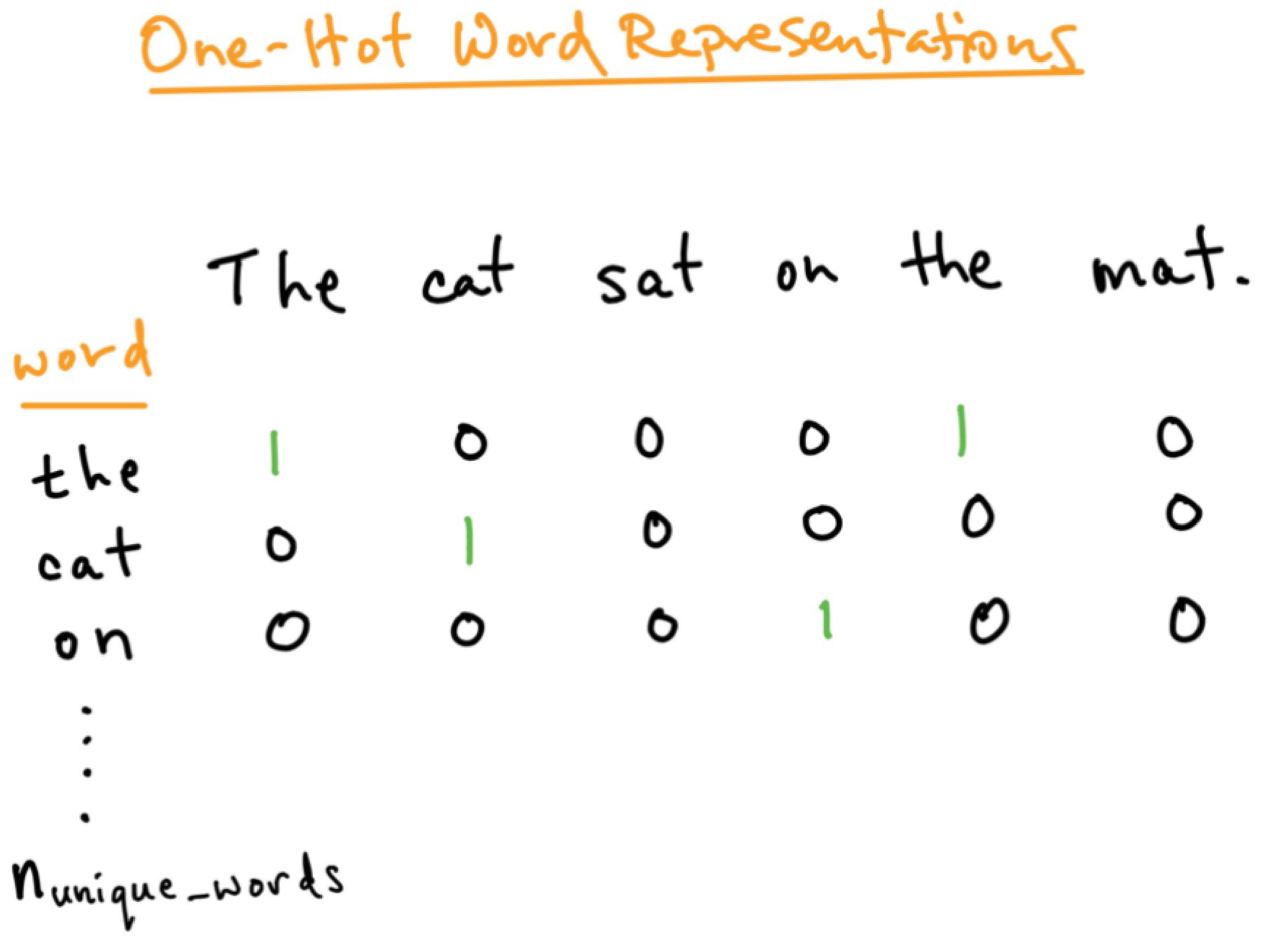
Picture Credit- Google

Term Frequency (TF): Term frequency is defined as frequency of word in the current document.

Inverse Document Frequency(IDF): is a measure of how much information the word provides, i.e., if it’s common or rare across all documents. It is calculated as log (N/d) where, N is total number of documents and d is the number of documents in which the word appears.

One-Hot Encodings

> *One hot encodings are another way of representing words in numeric form.* ***The length of the word vector is equal to the length of the vocabulary, and each observation is represented by a matrix with rows equal to the length of vocabulary and columns equal to the length of observation, with a value of 1 where the word of vocabulary is present in the observation and a value of zero where it is not.***



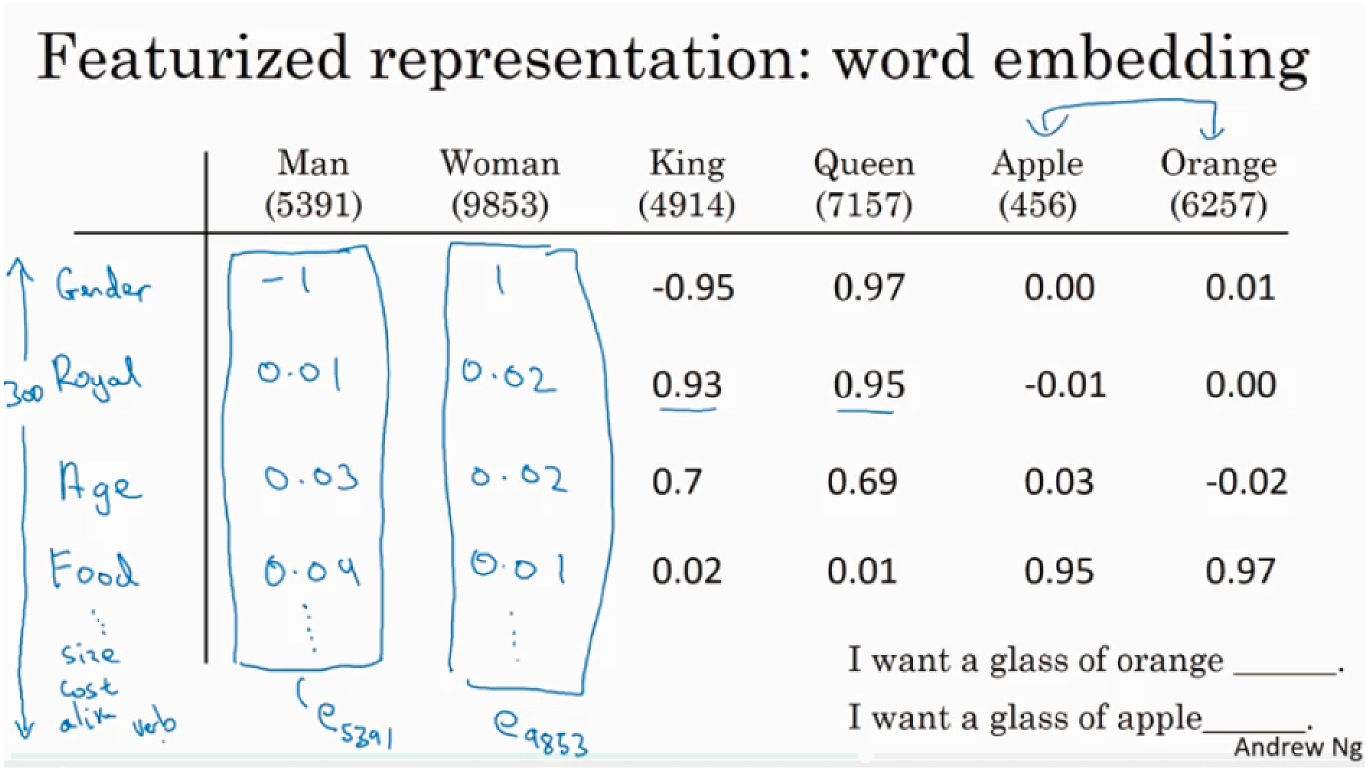
Picture Credit — Jon Krohn (picked via google search)

Word Embeddings

> ***Word embedding is the collective name for a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. The technique is primarily used with Neural Network Models.***

Conceptually it involves projecting a word from a dimension equivalent to the vocabulary length to a lower dimensional space, with the idea being that similar words will be projected closer to each other.

For the sake of understanding we can think of embeddings as each word being projected to a space of characteristics as shown in the image below.



Featurized representation: word embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.62	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97

I want a glass of orange ______.

I want a glass of apple______.

300
size
cost
alive
verb
e_{5391}
e_{9853}
Andrew Ng

Each word being represented in a space of characteristics (Gender, Royal, Age, Food) etc

> ***However, in reality these dimensions are not that clear or easily understandable. This does not concur a problem as the algorithms train on the mathematical relationships between the dimensions. What is represented by the dimension is meaningless for a neural network from training and prediction point of view.***

If one is interested in getting a visual understanding of Linear Algebra , projections and transformations which are the core mathematical principles behind a number of machine learning algorithms i will highly encourage them to visit “**Essence of Linear Algebra**” by 3Blue1Brown.

Vectors, what even are they? | Essence of linear algebra, chapter 1

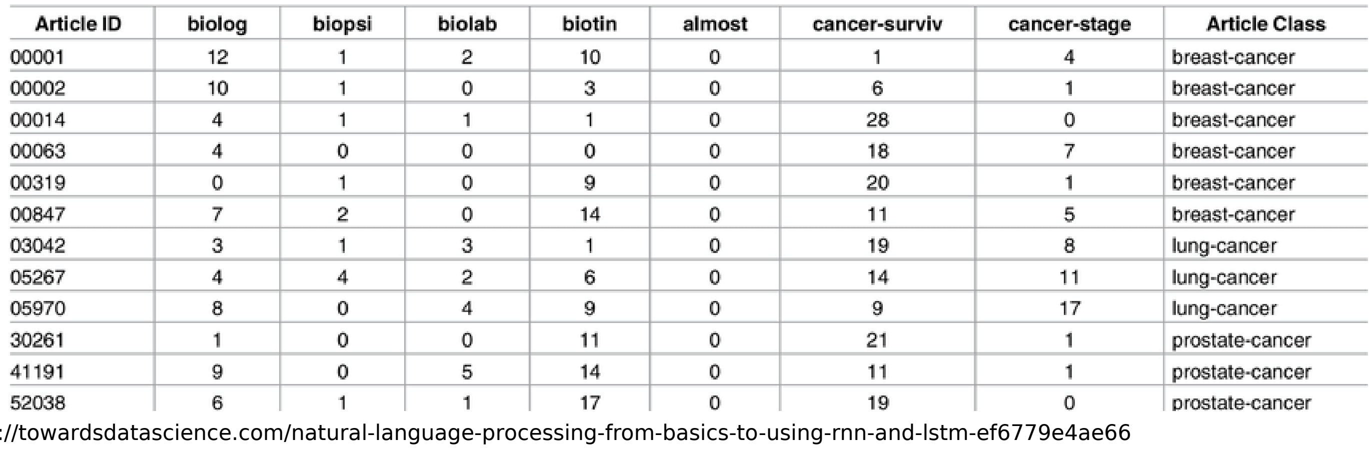
Representation Methods

Bag of Words

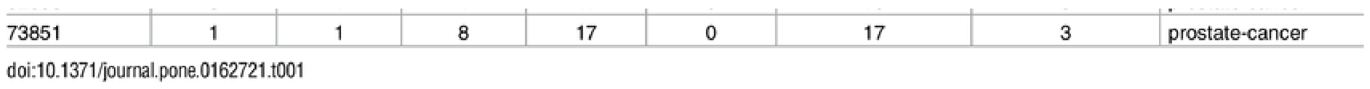
For an algorithm to derive relationships amongst the text data it needs to be represented in a clear structured format.

> ***Bag of words is a way to represent the data in a tabular format with columns representing the total vocabulary of the corpus and each row representing a single observation. The cell (intersection of the row and column) represents the count of the word represented by the column in that particular observation.***
>
> ***It helps a machine understand a sentence in an easy to interpret paradigm of matrices and thus enables various linear algebraic operations and other algortihms to be applied on the data to build predictive models.***

Below is an example of Bag of Words Model for a sample of articles from medical journals



Article ID	biolog	biopsi	biolab	biotin	almost	cancer-surviv	cancer-stage	Article Class
00001	12	1	2	10	0	1	4	breast-cancer
00002	10	1	0	3	0	6	1	breast-cancer
00014	4	1	1	1	0	28	0	breast-cancer
00063	4	0	0	0	0	18	7	breast-cancer
00319	0	1	0	9	0	20	1	breast-cancer
00847	7	2	0	14	0	11	5	breast-cancer
03042	3	1	3	1	0	19	8	lung-cancer
05267	4	4	2	6	0	14	11	lung-cancer
05970	8	0	4	9	0	9	17	lung-cancer
30261	1	0	0	11	0	21	1	prostate-cancer
41191	9	0	5	14	0	11	1	prostate-cancer
52038	6	1	1	17	0	19	0	prostate-cancer



73851	1	1	8	17	0	17	3	prostate-cancer

doi:10.1371/journal.pone.0162721.t001

Pictrue Credit -Google

This representation has worked really well, and has been responsible for churning out models for some of the most commonly used machine learning tasks such as spam detection, sentiment classifier and others.

However, there are two major drawbacks of this representation:

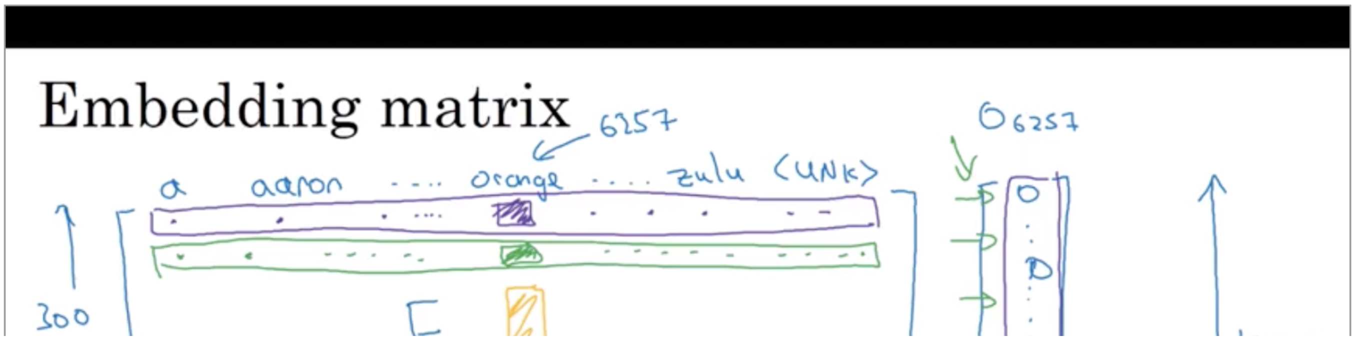
1. **It disregards the order/grammar of the text** and thus looses the context in which a word is being used
2. **The matrix generated by this representation is highly sparse** and more biased towards the most common words. Think about it, the algorithms primarily work on the count of the words, whereas in language the importance of word is actually inversely proportional to frequency of occurrence. Words with higher frequency are more general words like the, is, an, which do not alter the meaning of sentence significantly. Thus it becomes important to weigh the words appropriately so as to reflect there adequate impact on the meaning of a sentence.

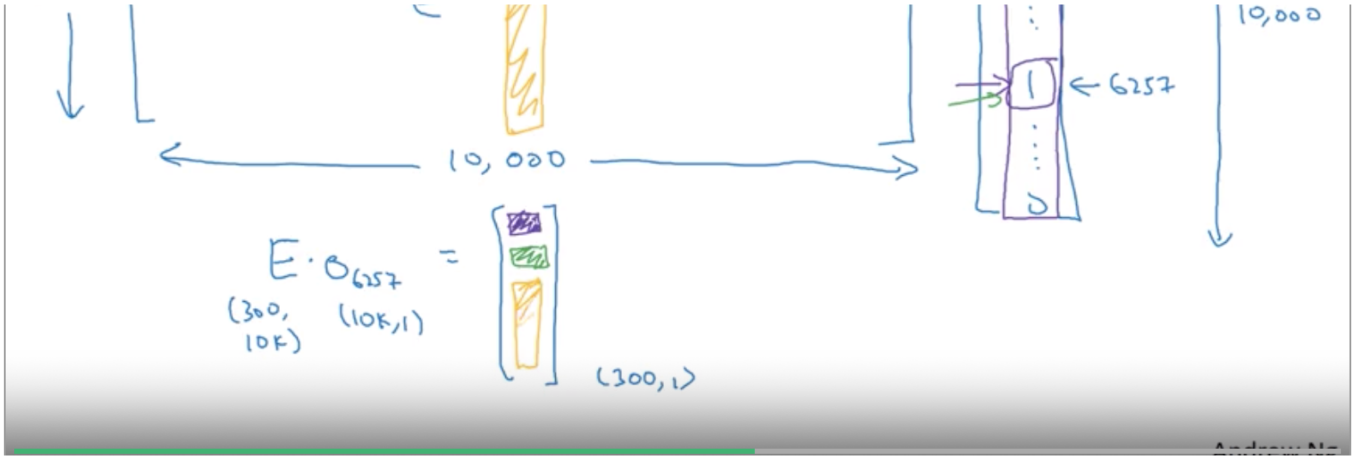
Embedding Matrix

Embedding matrix is a way to represent the embeddings for each of the words present in the vocabulary. Rows represent the dimensions of the word embedding space and columns represent the words present in the vocabulary.

> ***To convert a sample into its embedding form, each of the word in its one hot encoded form is multiplied by the embedding matrix to give word embeddings for the sample.***

Embedding matrix





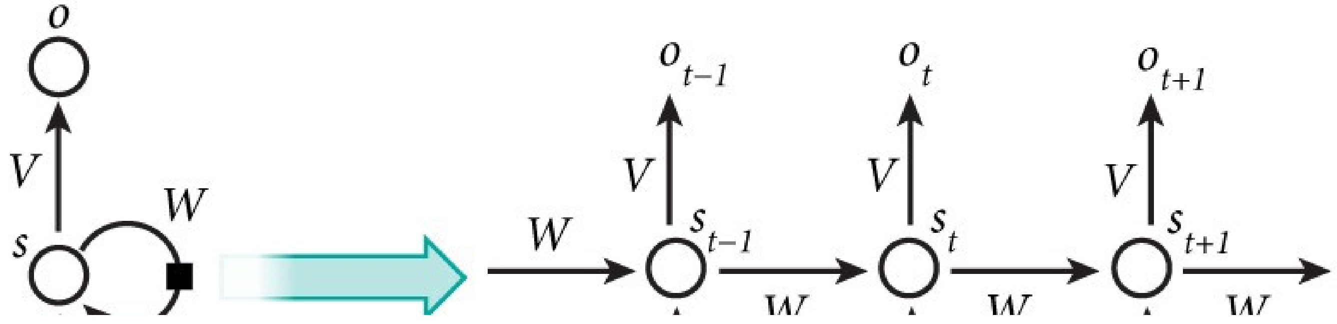
One thing to keep in mind is that here a one-hot encoding merely refers to an n-dimensional vector with a value 1 at the position of word in the vocabulary, where n is the length of the vocabulary. These one-hot encodings are drawn from the vocabulary instead of the batch of observations.

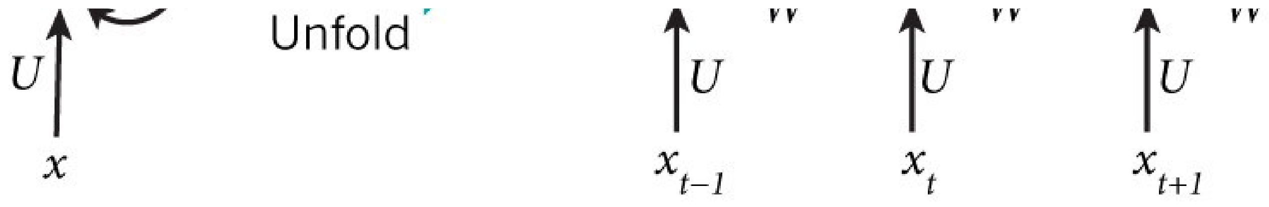
Recurrent Neural Networks (RNN)

Recurrent Neural Networks or RNN as they are called in short, are a very important variant of neural networks heavily used in Natural Language Processing.

> *Conceptually they differ from a standard neural network as the standard input in a RNN is a word instead of the entire sample as in the case of a standard neural network. This gives the flexibility for the network to work with varying lengths of sentences, something* ***which cannot be achieved in a standard neural network*** *due to it's fixed structure. It also provides an additional advantage of* ***sharing features learned across different positions of text*** *which can not be obtained in a standard neural network.*

A RNN treats each word of a sentence as a separate input occurring at time 't' and uses the activation value at 't-1' also, as an input in addition to the input at time 't'. The diagram below shows a detailed structure of an RNN architecture.

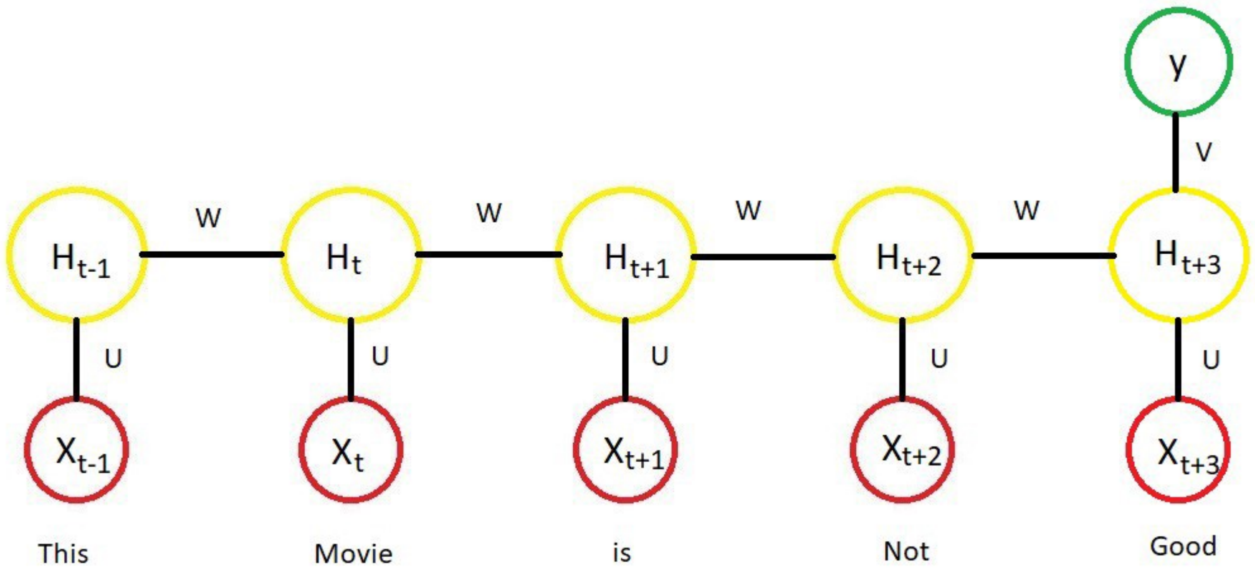




The architecture described above is also called as a many to many architecture with (Tx = Ty) i.e. number of inputs = number of outputs. Such structure is quite useful in Sequence modelling.

Apart from the architecture mentioned above there are three other types of architectures of RNN which are commonly used.

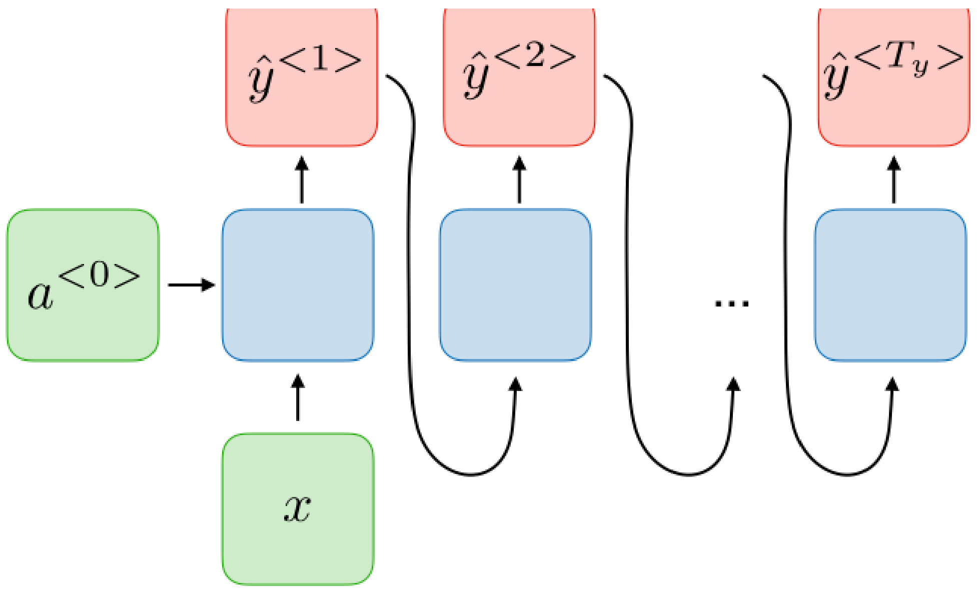
1. **Many to One RNN** : Many to one architecture refers to an RNN architecture where many inputs (Tx) are used to give one output (Ty). A suitable example for using such an architecture will be a **classification task**.



Picture Credit — Google

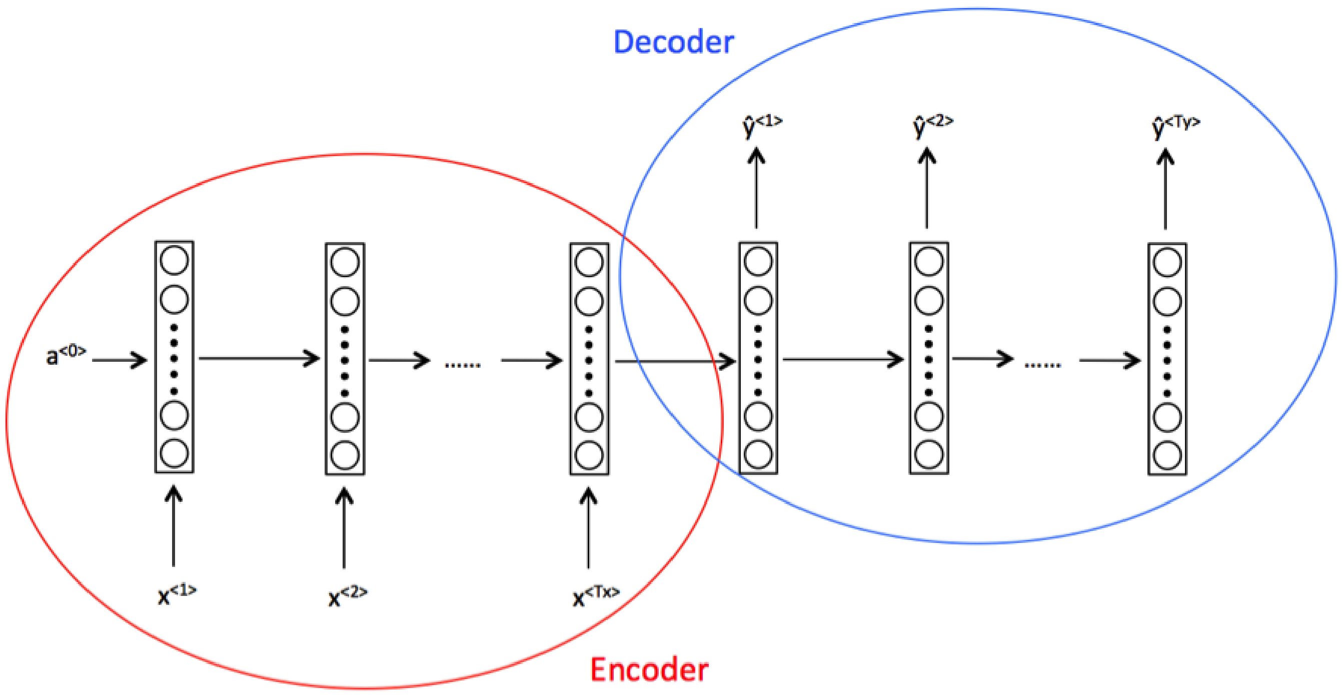
In the image above H represents the output of the activation function.

2. **One to Many RNN**: One to Many architecture refers to a situation where a RNN generates a series of output values based on a single input value. A prime example for using such an architecture will be a **music generation** task, where an input is a jounre or the first note.



Picture Credit — Google

3. Many to Many Architecture (Tx not equals Ty): This architecture refers to where many inputs are read to produce many outputs, where the length of inputs is not equal to the length of outputs. A prime example for using such an architecture is **machine translation** tasks.



Picture Credit- Google

Encoder refers to the part of the network which reads the sentence to be translated, and, **Decoder** is the part of the network which translates the sentence into desired language.

Limitations of RNN

Apart from all of its usefulness RNN does have certain limitations major of which are :

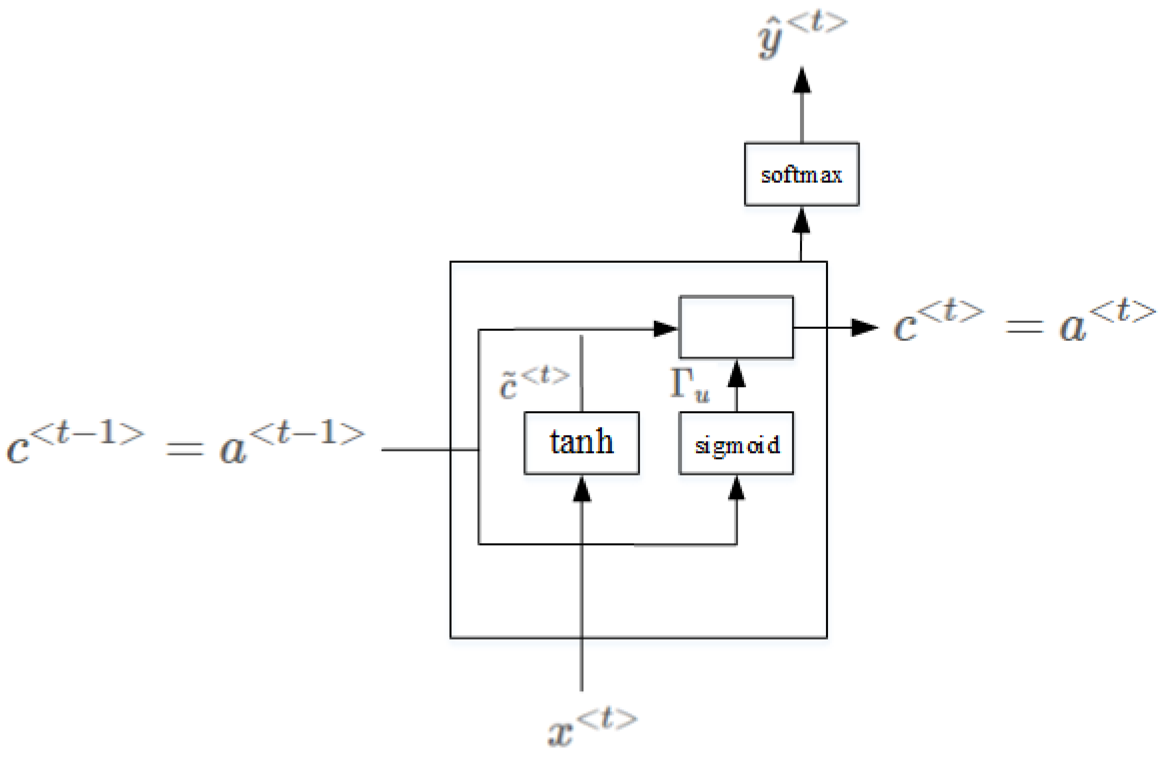
1. Examples of RNN architecture stated above are capable of capturing the dependencies in only one direction of language. Basically in case of Natural Language Processing it assumes that the word coming after has no effect on the meaning of the word coming before. With our experience of languages we know that it is certainly not true.

2. RNN are also not very good in capturing long term dependencies and the problem of vanishing gradients resurface in RNN.

Both these limitations give rise to new types of RNN architectures which are being discussed below.

Gated Recurrent Unit

It is a modification in the basic recurrent unit which helps to capture long range dependencies and also help a lot in fixing vanishing gradient problem.



Picture Courtesy Google

> *GRU consists of an additional memory unit commonly referred as an update gate or a reset gate.* ***Apart from the usual neural unit with sigmoid function and softmax for output it contains an additional unit with tanh as an activation function****. Tanh is used since its output can be both positive and negative hence can be used for both scaling up and down. The output from this unit is then combined with the activation input to update the value of the memory cell.*

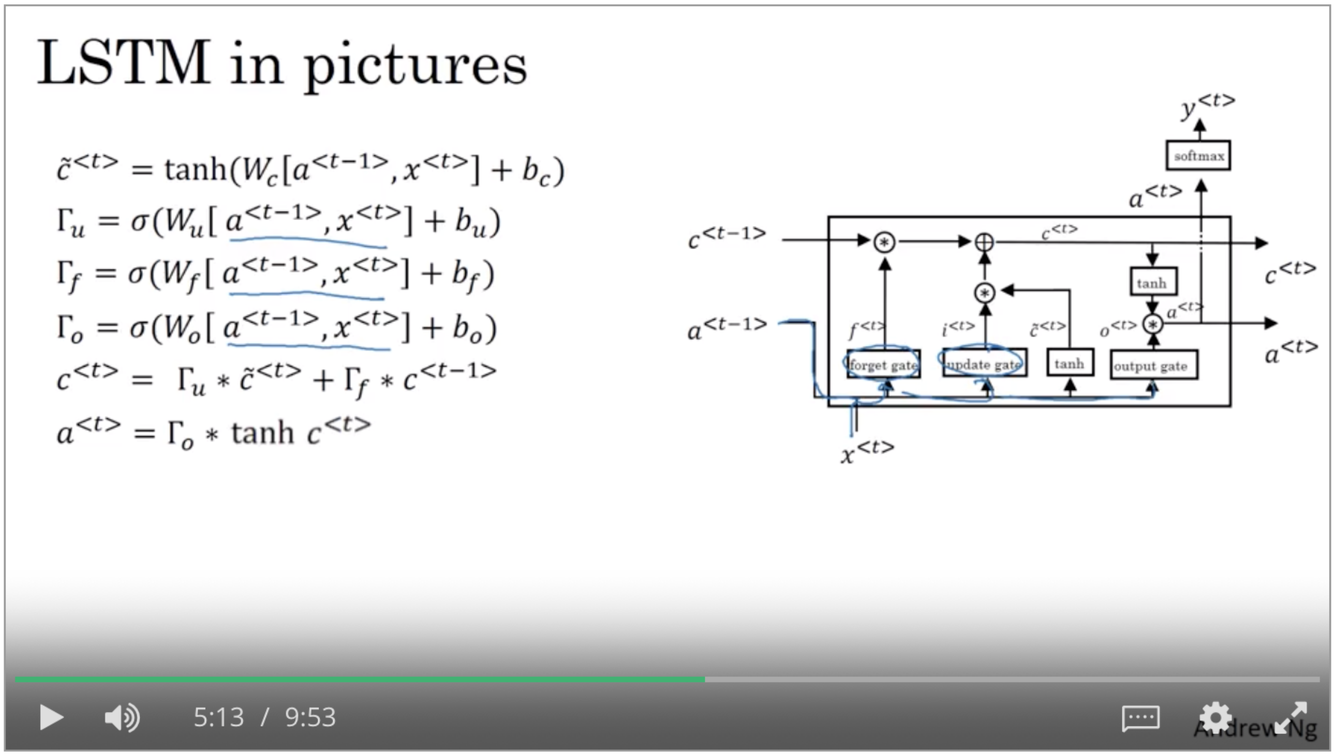
Thus at each step value of both the hidden unit and the memory unit are updated. The value in the memory unit, plays a role in deciding the value of activation being passed on to the next unit.

For a more detailed explanation one can refer to https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

LSTM

In LSTM architecture instead of having one update gate as in GRU there is an update gate and a forget gate.

Long Short Term Memory (LSTM)



LSTM in pictures

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[\underline{a^{<t-1>}, x^{<t>}}] + b_u)$$
$$\Gamma_f = \sigma(W_f[\underline{a^{<t-1>}, x^{<t>}}] + b_f)$$
$$\Gamma_o = \sigma(W_o[\underline{a^{<t-1>}, x^{<t>}}] + b_o)$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

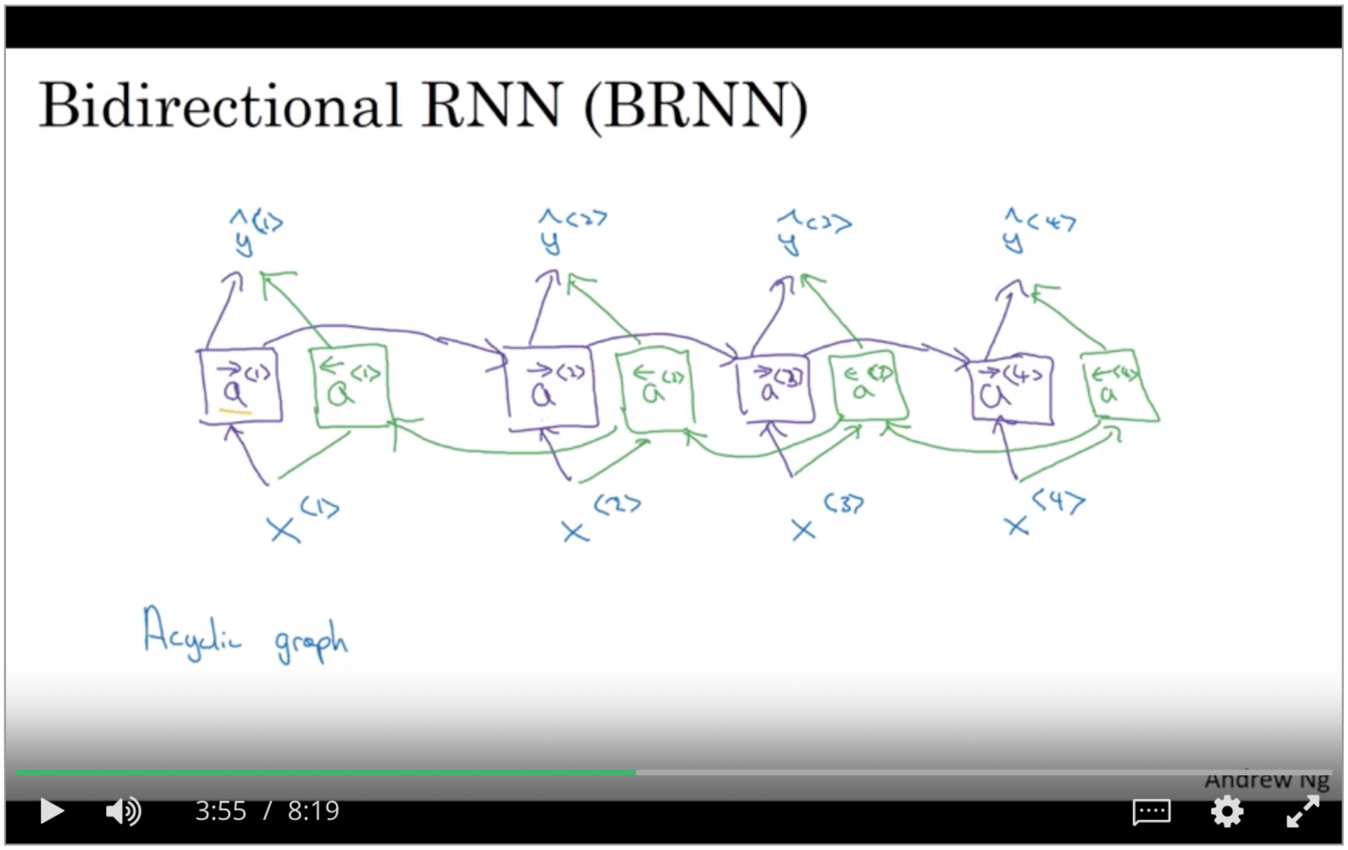
This architecture gives the memory cell an option of keeping the old value at time t-1 and adding to it the value at time t.

A more detailed explanation of LSTM is available at **http://colah.github.io/posts/2015-08-Understanding-LSTMs/**

Bidirectional RNN

In the above RNN architectures effects of occurrences at only the previous time stamps can be taken into account. In the case of NLP it means that it takes into account the effects of the word written only before the current word . But this is not the case in a language structure and thus Bi-directional RNN come to the rescue.

Bidirectional RNN



A bi-directional RNN consists of a forward and a backward recurrent neural network and final prediction is made combining the results of both the networks at any given time t, as can be seen in the image.

. . .

In this blog I have tried to cover all the relevant practices and neural network architectures prevalent in the world of Natural Language Processing. **For those interested in in-depth understanding of a neural network i will highly encourage to go through Andrew Ng Coursera course**.

References

1. https://machinelearningmastery.com/natural-language-processing/
2. Deep Learning Coursera, Andrew Ng
3. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
4. 3Blue1Brown series You Tube
5. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be
6. **http://colah.github.io/posts/2015-08-Understanding-LSTMs/**

Machine Learning Neural Networks Data Science NLP Naturallanguageprocessing