Abstractive text summarization

Text summarization is an NLP grounded fashion which involves converting large number of paragraphs into simple accessible judgment, done by involving various grammatical connections, sentence matching etc. Analogy of the original paragraphs or texts is maintained. Let’s say that we are reading a review or newspaper or some kind of research paper where you come across huge paragraphs. This becomes a tedious task for anyone who, just want to focus on the main context of what they are reading and save their time. NLP involves creation of analogy and generating the texts. This method of generating the texts should be ensured that verbatim is not occurring.

**Verbatim** is basically called as the words that already have been used in the original paragraph or text should not be used again. We can ensure the percentage of the words that can be used in creating the summary. By text summarization we easily filter out the main context of any content provided, and take meaning full actions with that summary. As we humans can consume a finite amount of information provided at times it is important for the summarizers for deriving to meaningful conclusions with as minimum noise as possible.

Text summarization is helpful in providing useful information from the text documents given, spending less time in reading and getting useful information. News articles, fact sheets and mails fall under these categories. Sentences builds upon the previous, text summarization may not be helpful. Research journals, medical text are a good example where text summarizations are not very accurate. The summarized texts might help us giving the summarized results but they might lack the style and tone of the text that the author tries to convey.

In text summarization, the most important information of an original document is summarized in a way so that the main points of the document can be easily summarized. Multi-document summarization aims to produce a concentrated version of the document while keeping the key information. Due to the massive amount of data available these days, summarization has become increasingly important. As a final note, this paper is collected and the most recent and research in the field of text summarization to study and analyze for future research happens. In the future, it will provide a new direction for those who are interested in this field.

There's a huge quantum of data surfacing digitally, thus the significance of developing a punctuate procedure to dock long textbooks incontinently while keeping the main idea of it is necessary. Summarization helps mitigate the time needed for reading, provide quick search for information and to help get the most desired information on the topic. The central object of motorized textbook summarization is dwindling the reference textbook into a lower interpretation maintaining its knowledge alongside with its meaning. Several descriptions of text summarization are provided, for example explained the report as text that is generated from one or more documents that communicate relevant knowledge in the first text, and that is no higher than half of the primary text and usually significantly more limited than that.

Then again, it has been seen that the summarization task with different kinds has not been formalized as a multitarget streamlining task previously, despite that there are numerous goals which can be considered. The SI was not used before to help the continuous synopsis that draws near. Subsequently, another model has been proposed to be satisfactory for accomplishing numerous targets and to fulfil the constant needs.

In the long run, this investigation will enthuse analysts to further consider the different sorts of SI when tackling the synopsis assignments, especially, in the short content outline field. There is a huge amount of data surfacing digitally, therefore the importance of developing a punctuate procedure to shorten long texts immediately while keeping the main idea of it is necessary.

Summarization also helps dock the time demanded for reading, fasten the hunt for information and help to get the most quantum of information on one content. Text summarization is a fascinating literacy content that caught attention fleetly, as exploration increase, they're hoping to witness a advance that will affect this by furnishing immediate method in summarizing long texts.

The swell of information available through the internet and social networks and information technologies make the need of summarization more critical, especially with the massive quantum of data that's being spread due to the knowledge transfer among its druggies, which makes it delicate to separate between the right information from the wrong bones.

There are two principles for the textbook summarization systems extractive and abstractive. Extractive summarization creates synopsis by choosing remarkable rulings or expressions from the source content, while abstractive strategies paraphrase and rebuild rulings to form the summary. They concentrate around abstractive summarization in this work as it's decreasingly adaptable and hence can produce precipitously different summaries.

Recent neural system ways to deal with an outline are generally either sentence extractive, choosing a lot of sentences as the summary, or abstractive, creating the summary from a seq2seq model. In this work, they present a neural model for single-record summary dependent on joint extraction and pressure. Following latterly fruitful extractive models, they outline the summarization issues as a progression of original choices. This model picks sentences from the report and after that chooses which of a set of compression options to apply to each chosen sentence.

They cipher this arrangement of separate contraction rules dependent on syntactic constituency parses; still, the proposed methodology is measured and it could use for any accessible source of condensing. For learning, they build oracle extractive compressive summaries that reflect vulnerability over the proposed model’s decision sequence, and then learn both parts together with this supervision.

Test results on the CNN/ Daily Mail and New York Times datasets demonstrate that this model accomplishes the innovative prosecution on substance determination assessed by Cream. either, mortal, and homemade assessment demonstrate that the proposed model’s yield, for the utmost part, remains.

**Extractive**

Let us consider that you have exam tomorrow and you take out your text book and start marking the important sentences from each of the topics using a highlighter. Similarly extractive is a method of highlighting the existing words and phrases by ranking them based on the relevance and understanding of the texts. This method presents you with the most important sentences.

Extractive Summarization ways vary, yet they all partake the same introductory task:

* Construct an intermediate representation of the input text (i.e the text that is to be summarized).
* Score the sentences based on their constructed intermediate representation.
* Select the summary consisting of the top k most important sentences.

Step 2 and 3 are pretty simple and easy; in sentence scoring we want to determine how well each sentence relays important aspects of the text being summarized, while sentence selection is performed using some specific optimization algorithm. These algorithms are very simple they only follow 2 steps conceptually those steps are:

* Assign a score to each judgment using some metric.
* Elect from the stylish scored rulings via some well- defined judgment selection system.

For serving the purpose of summarization to being in a natural language some intermediate representation has to be made prior to the sentence scoring and selection. It follows two such categories of intermediate representation they are topic representation and indicator representation.

**1) Construction of an intermediate representation of the input textbook**

There are two types of representation- grounded approaches content representation and index representation. Content representation transforms the textbook into an intermediate representation and interprets the content bandied in the textbook. The ways used for this differ in terms of their complexity, and are divided into frequency- driven approaches, content word approaches, and idle semantic analysis and Bayesian content models. Indicator representation describes every judgment as a list of formal features (pointers) of significance similar as judgment length, position in the document, having certain expressions etc.

**2) Scoring the rulings grounded on the representation**

When the intermediate representation is generated, a significance score is assigned to each judgment. In content representation approaches, the score of a judgment represents how well the judgment explains some of the most important motifs of the textbook. In index representation, the score is reckoned by adding up the substantiation from different weighted pointers.

**3) Selection of a summary comprising of a number of rulings**

The summarizer system selects the top k most important rulings to produce a summary. Some approaches use greedy algorithms to elect the important rulings and some approaches may convert the selection of rulings into an optimization problem where a collection of rulings is chosen, considering the constraint that it should maximize overall significance and coherency and minimize the redundancy.

**Topic Representation**

Transformation that focuses on the context of the topic, these are broadly classified into major categories they are –

* Frequency driven approach
* Topic word approach
* Latent semantic analysis (LSA)
* Bayesian topic models – Latent Dirichlet allocation (LDA)

**Frequency driven approach**

Frequency driven method uses the TF-IDF method of vectorising the sentences by giving them weights. It calculates the probability of the word determined by the number of times the word occurs f(w) divided by the total number of words in the document and are included in the summary. TF-IDF also identifies the words that are most commonly occurring in the document and gives them the least weights (those words should not be considered). This has paved a way to another approach called as the centroid based approach that ranks the sentences based on the computations of set of features After the creation of TF- IDF vector representations of documents, the documents that describe the same content are clustered together and centroids are reckoned. Pseudo documents that consist of the words whose TF-IDF scores are higher than the actual threshold and they are clustered together. These centroids are used to identify rulings in each cluster that are central to the content.

* Pseudo documents – documents that consist of terms with higher frequencies.

**Topic words**

Fashion, that aims to identify words that describe the content of the input document. Luhn’s idea was to use log- liability rate test to identify explicatory words known as the “content hand”. Generally speaking, there are two ways to cipher the significance of a judgment.

* Using a function of the number of content autographs it contains.
* Using a proportion of the content autographs in the judgment.

From the above mentioned two methods the first method gives higher scores to longer sentences, the second one measure the density of the topic words.

**Latent Semantic Analysis**

Latent semantic analysis (LSA) is an unsupervised method and also a natural language processing technique for extracting a representation of text semantics based on observed words. The first step is to build a term-sentence matrix, where each row corresponds to a word from the input (n words) and each column corresponds to a sentence. Each entry of the matrix is the weight of the word i in sentence j computed by TF-IDF technique.

Then singular value decomposition (SVD) is used on the matrix that transforms the initial matrix into three matrices: a term-topic matrix having weights of words, a diagonal matrix where each row corresponds to the weight of a topic, and a topic-sentence matrix. If you multiply the diagonal matrix with weights with the topic-sentence matrix, the result will describe how much a sentence represent a topic, in other words, the weight of the topic i in sentence j. Features that are hidden in the data which cannot be directly measured. These features are essential to the data, but are not original features of the dataset.

The flow chart represents the steps of Latent semantic analysis –

Topic encoded data

Singular value decomposition

Document term matrix

Raw text data

Document term matrix is creation of vectorised features of the words or in simple terms we can say it as converting the document into bag of words

Singular value decomposition is similar to principle component analysis which uses a method of reducing the features into some specified components, generally speaking it is used in dimensionality reduction.

It reduces the dimensions of the dataset by encoding it with latent features with LSA these latent features represent topics in the original text data.

The by-products of the singular value decomposition gives two results they are Dictionary, encoding matrix.

Dictionary provides the information of the feature names that we have converted our document into encoding matrix is a data frame that is converted to visually see the decomposed results of each of the words in the document.

**LSA Using python**

Consider the document –

text = ['the quick brown fox', 'the slow brown dog', 'the quick red dog', 'the lazy yellow fox']

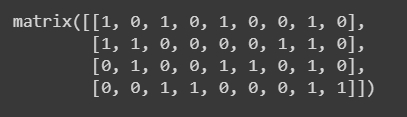
Using CountVectorizer function, transform the document into the bag of words.

vectors = CountVectorizer()

vectorize = vectors.fit(text)

vectors\_text = vectors.transform(text)

vectors\_text.todense()



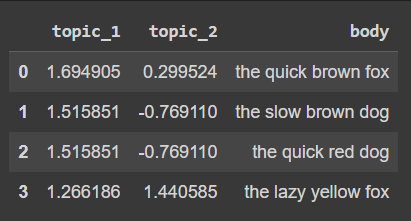
Using the singular value decomposition convert the sparse matrix into decomposed matrix with 2 components

svd = TruncatedSVD(n\_components=2)

decomposed = svd.fit\_transform(vectors\_text)

data = DataFrame(decomposed, columns=['topic\_1', 'topic\_2'])

data['body'] = text



To get the feature names of the decomposed document

vectorize.get\_feature\_names\_out()



Encode the matrix into a dataframe with each of the words having some composition

# encode\_matrix = DataFrame()

encode\_matrix = DataFrame(svd.components\_, index=['topic\_1', 'topic\_2'], columns=vectorize.get\_feature\_names\_out()).transpose()

encode\_matrix



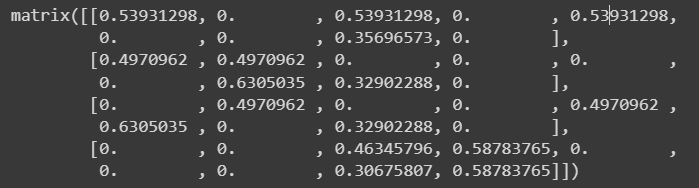
Using the term frequency inverse document frequency

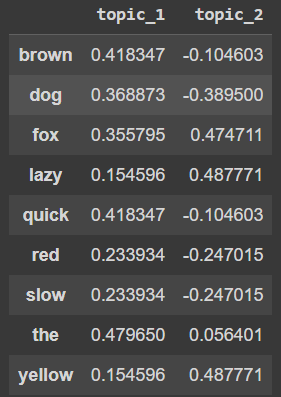
tfidf = TfidfVectorizer()

tf\_vectors = tfidf.fit(text)

tf\_data = tfidf.transform(text)

tf\_data.todense()





**Latent dirichlet allocation – A bayesian method of topic modelling**

Bayesian method is a probabilistic method of describing the topics in more detail that can represent the information that is lost in their own approaches. The goal in topic modelling is to infer to the words related to a certain topic and the topics discussed in a certain document, based on the prior analysis of a corpus of documents. With Bayesian models it is possible as it calculates the probabilities of an event based on a combination of common sense assumptions and the outcomes of previous related events. Bayesian model is constantly improved by going through many iterations, where a prior probability is updated with observational evidence to produce a new posterior probability. Latent dirichlet allocation is a topic model that generates topics based on words frequency from a set of documents used for finding reasonable accurate mixture of topics within a given a document

LDA is helpful in finding the most highlighted contents in dataset by assigning a prior probability and calculating the posterior probability finding the word at each new topic.

Assumptions in LDA –

* Convert the document into “bag of words”.
* By removing the stopwords from the document it is still relevant for considering. As stopwords do not incur to any possible meaning, they don’t affect the results of the LDA or any loss of information.
* K is the number of topics which is pre-defined.
* Our model generates the documents based off of our current word by updating the assignment, assuming it except for the current word in question are correct.

Working of LDA –

* The words that belong to a document, that we know already.
* We need to calculate the probability of words belonging into a topic.

**Indicator representation approaches**

The second large group of techniques aims to represent the text based on a set of features and use them to directly rank the sentences without representing the topics of the input text.

**Graph Methods**

It uses Pagerank algorithm, these styles represents the documents as a connected graph, where rulings form the vertices and edges between the rulings indicate how analogous the two rulings are. The similarity of two sentences is measured with the help of cosine similarity with TD-IDF weights for words and if it is greater than a certain threshold, these sentences are connected. This graph representation results in two issues thesub-graphs included in the graph produce motifs covered in the documents, and the important rulings are linked. Sentences that are connected to many other sentences in a sub-graph are likely to be the center of the graph and will be included in the summary Since this system don't need language-specific verbal processing, it can be applied to colorful languages.

At the same time, similar measuring only of the formal side of the judgment structure without the syntactic and semantic information limits the operation of the system.

**Page Rank Algorithm**

Graph-based ranking algorithms like Kleinberg’s HITS algorithm (Kleinberg, 1999) or Google’s PageRank (Brin and Page, 1998) have been successfully used in citation analysis, social networks, and the analysis of the link-structure of the World Wide Web. Arguably, these algorithms can be singled out as crucial rudiments of the paradigm- shift touched off in the field of Web hunt technology, by furnishing a Web runner ranking medium that relies on the collaborative knowledge of Web engineers rather than individual content analysis of Web runners. In short, a graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information.

**Hits Algorithm**

Hyperlink- Induced Topic Search the idea behind capitals and Authorities stemmed from a particular sapience into the creation of web runners when the Internet was firstly forming; that is, certain web runners, known as capitals, served as large directories that weren't actually authoritative in the information that they held, but were used as compendiums of a broad roster of information that led druggies direct to other authoritative runners.

In other words, a good mecca represents a runner that refocused to numerous other runners, while a good authority represents a runner that's linked by numerous different capitals.

The scheme thus assigns two scores for each runner

• Its authority, which estimates the value of the content of the runner.

• Its hub value, which estimates the value of its links to other runners.

In the successes algorithm, the first step is to recoup the most applicable runners to the hunt query. This set is called the root set and can be attained by taking the top runners returned by a textbook- grounded hunt algorithm. A base set is generated by accelerating the root set with all the web runners that are linked from it and some of the runners that link to it.

The web runners in the base set and all hyperlinks among those runners form a focused subgraph. The successes calculation is performed only on this focused subgraph. According to Kleinberg the reason for constructing a base set is to insure that utmost( or numerous) of the strongest authorities are included.

Authority and hub values are defined in terms of one another in a collective recursion. An authority value is reckoned as the sum of the gauged mecca values that point to that runner. A mecca value is the sum of the gauged authority values of the runners it points to. Some executions also consider the applicability of the linked runners.

The algorithm performs a series of iterations, each consisting of two basic steps:

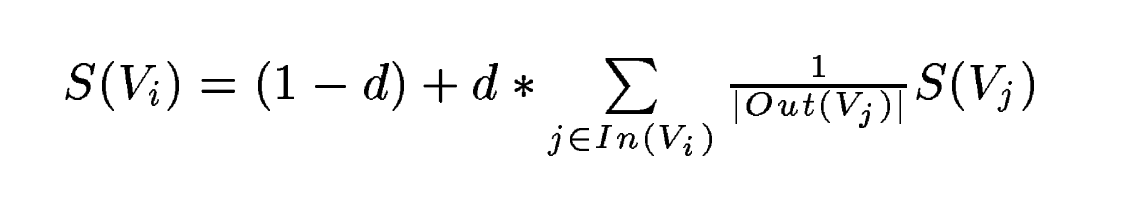
• **Authority update:** Update each node's authority score to be equal to the sum of the hub scores of each node that points to it. That is, a node is given a high authority score by being linked from runners that are honored as capitals for information**.**

• **Hub update**: Update each node's hub score to be equal to the sum of the authority scores of each node that it points to. That is, a node is given a high hub score by linking to bumps that are considered to be authorities on the subject.

Applying a analogous line of allowing to verbal or semantic graphs uprooted from natural language documents, results in a graph- grounded ranking model that can be applied to a variety of natural language processing operations, where knowledge drawn from an entire textbook is used in making original ranking/ selection opinions.

Similar textbook- acquainted ranking styles can be applied to tasks ranging from automated birth of key phrases, to extractive summarization and word sense disambiguation.

Graph- grounded ranking algorithms are basically a way of deciding the significance of a vertex within a graph, grounded on global information recursively drawn from the entire graph.

The basic idea implemented by a graph-based ranking model is that of “voting” or “recommendation”. When one vertex links to another one, it's principally casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex.

**V – be set of vertices**

**E – be set of edges**

**In(Vi) – be the set of vertices Vi points to the predecessors**

**Out(Vi) – be the set of vertices Vi points to the successors**

**d – be the damping factor which can only be in the range of 0 and 1.**

Usually the user enters into a new page when a link is clicked with a probability between 0 and 1, when the user is randomly surfing through the web, hence called as the “Random surfer model”. And the damping factor is usually put at .85.

**Text as graph** **in PageRank algorithm**

To enable the operation of graph- grounded ranking algorithms to natural language textbooks, we've to make a graph that represents the textbook, and interconnects words or other textbook realities with meaningful relations. Depending on the operation at hand, textbook units of colorful sizes and characteristics can be added as vertices in the graph, e.g. lexical or semantic relations, contextual overlap, etc.

**Keyword Extraction in PageRank algorithm**

The task of a keyword birth operation is to automatically identify in a textbook a set of terms that stylish describe the document. Similar keywords may constitute useful entries for building an automatic indicator for a document collection, can be used to classify a textbook, or may serve as a terse summary for a given document. Also, a system for automatic identification of important terms in a textbook can be used for the problem of language birth, and construction of sphere-specific wordbooks..

**Ranking the text after Keyword Extraction in PageRank algorithm**

The anticipated end result for this operation is a set of words or expressions that are representative for a given natural language textbook. The units to be ranked are thus sequences of one or further verbal units uprooted from textbook, and these represent the vertices that are added to the textbook graph.

**Machine Learning**

Approaches for the text summarization have been significantly been applied to get the most text-to-life summaries from the given inputs like the paragraphs or docs. As the text summarization mainly focuses on the classification side, we use the machine learning models like SVM, random forests, decision trees, naïve bayes, hidden markov models or the Conditional random fields.

Out of the mentioned machine learning models hidden markov models and conditional random fields out preform the others.

**Hidden Markov Model**

Hidden markov model is the probabilistic classification model, mostly used in bioinformatics, NLP, speech recognition etc. Each of the nodes in the markov model is known as a state and each of these states has an outgoing transitions with some probabilities assigned. These probabilities can infer us the prediction of the unknown conditions.

In NLP each of these nodes are represented as the words and their transitions to the other nodes can tell us about the association of the word to the occurring word, hence these can be clustered in a on large group to make out the summary of the text or paragraphs.

The main thing of HMM is to learn about a Markov chain by observing its retired countries. Considering a Markov process X with retired countries Y then the HMM solidifies that for each time stamp the probability distribution of Y mustn't depend on the history of X according to that time.

**POS tagging in hidden markov model**

POS tagging is a very useful part of text pre-processing in NLP as we know that NLP is a task where we make a machine able to communicate with a human or with a different machine. So it becomes mandatory for a machine to understand the part of speech.

Classifying words in their part of speech and furnishing their markers according to their part of speech is called part of speech trailing or POS tagging OR POST. Hence the set of markers markers is called a label- set. In the composition, we've seen how we can apply the part of speech tagging or POS tagging OR POST. Hence the set of labels/tags is called a tag-set. In the article, we have seen how we can implement the part of speech at a beginning level using the NLTK where the tag-sets package of NLTK was helping us to provide the part of speech tag to our documents.

**Conditional random fields**

It’s a generalized multiclass logistic regression, it has increased flexibility for sequence labelling.

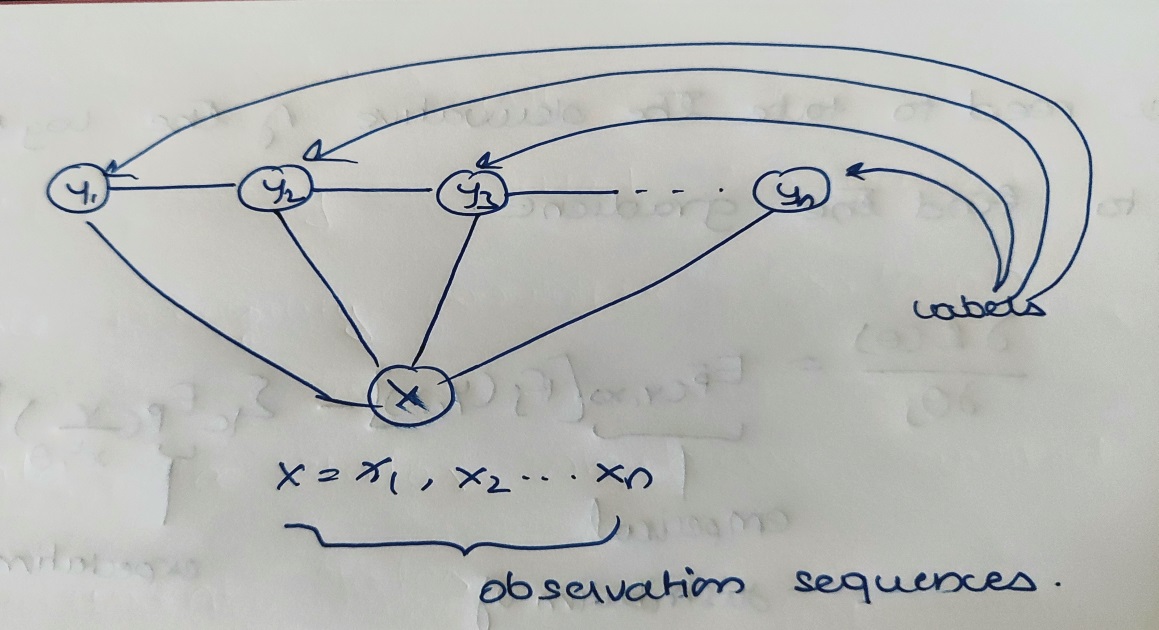
Hidden markov model – Joint probability – ranging over the observations and corresponding labels. Can leads to rigid independent assumptions.

Conditional random fields – Conditional probability - over label sequences given specific sequence of observations.

CRF’s are special case of hidden markov random fields which are undirected graphs that satisfy the markov property. CRF’s are mainly used for the entity recognition, part of speech tagging, gene prediction, noise reduction and object detection.

For general graphs, the problem of exact conclusion in CRFs is intractable. The conclusion problem for a CRF is principally the same as for an MRF and the same arguments hold. still, there live special cases for which exact conclusion is doable

* If the graph is a chain or a tree, communication passing algorithms yield exact results. The algorithms used in these cases are similar to the forward-backward and Viterbi algorithm for the case of HMMs.
* If the CRF only contains pair-wise potentials and the energy is submodular, combinatorial min cut/max flow algorithms yield exact solutions.



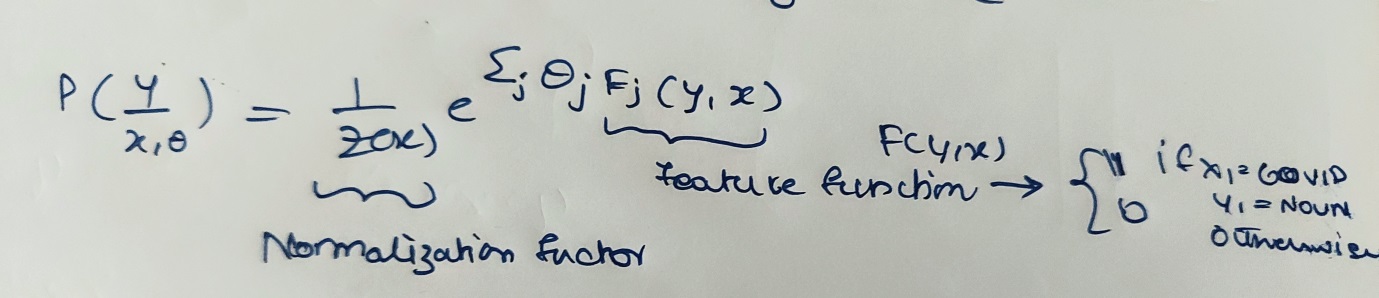
Let’s consider from the above graph that the connections be of two kinds; sequential connections between each of the label nodes and all these nodes be connected to the observational sequences; the other connection be namely that non sequential connections.

This inference tells us that CRF is truly a special case of hidden markov model. Each of the next states are dependent on the current states, as the connections can tell us from.

We can tell it as y2, is not connected to the, yn which is true for reasons, they infer as the conditionally independent labels cannot appear in the same potential function. Hence these labels can also be referred to as potential functions.

The other sequence example can be yn-1, yn are connected which means they are fully connected graphs. They instead require potential functions to operate only on random variables forming a maximal clique.

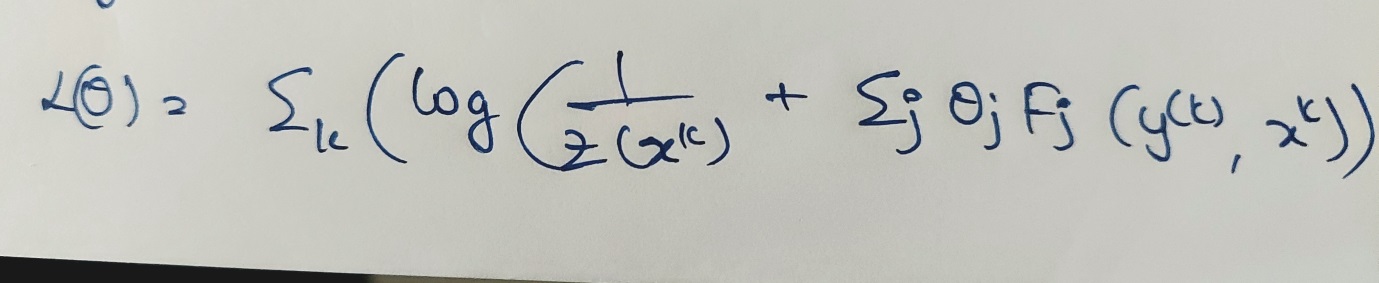
Probability of a label sequence y given observation sequence x is then a normalized product of feature functions.



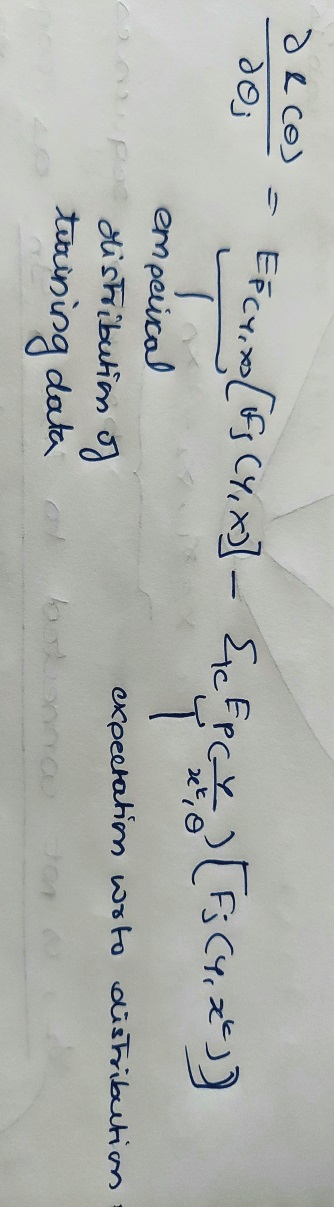
**Training CRF’s**

Seeks to find the model distribution with maximum entropy distribution is as uniform as possible.

Parameters can be optimized by minimizing cross entropy loss it can be given by the log likelihood of a CRF.



We can also find the gradient of the log likelihood to find the gradient



Yet, the problem with classifiers is that if we use supervised literacy styles for summarization, we need a set of labeled documents to train the classifier, meaning development of a corpus. A possible way- eschewal is to applysemi-supervised approaches that combine a small quantum of labeled data along with a large quantum of unlabeled data in training.

Overall, machine literacy styles have proved to be veritably effective and successful both in single andmulti-document summarization, especially in class-specific summarization similar as drawing scientific paper objectifications or biographical summaries.

Though abundant, all the summarization styles we've mentioned couldn't produce summaries that would analogous to mortal- created summaries. In numerous cases, the soundness and readability of created summaries aren't satisfactory, because they fail to cover all the semantically applicable aspects of data in an effective way and latterly they fail to connect rulings in a natural way.

Hence these are all various extractive methods that can be used for the text summarization. The mentioned methods are easy as compared to the abstractive text summarization, for better conclusion on extractive text summarization. Extractive text summarization is one of the methods for text summarization which uses two types of methods based on the topic representation and indicator representation.

Topic representation uses the method of TF-IDF which uses the documents words as frequencies and ranks the words based on their weights, which is similar to frequency driven method, LSA (latent semantic analysis) is an unsupervised learning method which uses the hidden words to measure them using the singular value decomposition, which is commonly used as dimensionality reduction. LDA (latent dirichelt allocation) is a Bayesian method that generates the topics based on the words frequency from a set of documents used for finding reasonable accurate mixture of topics within a given document. Indicator representation methods use the various graph and machine learning based methods that can be used in text summarization and have been much more efficient than the topic representation methods. Graph based method involves the use of pagerank algorithm based on hits algorithm and also uses the text rank. The words in the form of nodes and the association in the form of the links to the other nodes these are then clustered based on their similarity between the documents. Machine learning models that are used for classification for the text summarization can be used.

Algorithms like SVM, Naïve Bayes, logistic regression can be used but apart from these ML models Hidden markov model and conditional random fields have outperformed the above mentioned models. These models have higher intuition levels and are best at performing the NLP tasks as they use probabilistic calculation based on the current scenarios to predict the upcoming scenarios, hence these models are best suited for the text summarization in indicator representation methods.

**Abstractive Text Summarization**

The main topic of summarization has finally come into play. Abstractive text summarization is used in the advanced text summarization methods where new words are derived from the given words in the document. These new words gives much more meaning and semantic to the summarized topic. Hence generating these new words are advanced techniques.

Abstractive text summarization methods employ more powerful natural language processing techniques to interpret text and generate new summary text, as opposed to selecting the most representative existing excerpts to perform the summarization. On the other hand, it tries to guess the meaning of the whole textbook and presents the meaning to you.

It creates words and expressions, puts them together in a meaningful way, and along with that, adds the most important data set up in the textbook. This way, abstractive summarization ways are more complex than extractive summarization ways and are also computationally expensive.

While both are valid approaches to textbook summarization, it shouldn't be delicate to move you that abstractive ways are far more delicate to apply. In fact, the maturity of summarization processes moment are birth- grounded. This does not mean that abstractive styles should be blinked or ignored; on the negative, exploration into their perpetration and true semantic understanding of mortal language in general — is a good pursuit, and important work is demanded before we can confidently say that we've gained a true base in this bid.

**Sequences**

Sequenced data is data that takes the form of a list of varying length. Sequences can be difficult for traditional neural networks to process since there is the idea of an order, and the length may vary.

For example, consider the lyrics of a song, a sequence of words. The idea of an order means that certain words naturally come “ before ” others. It is easy to remember the words in the normal order, but much harder to recall the lyrics backwards.

In the real world, sequences can be any kind of data of varying length and has a general idea of an order. Some examples are texts, audio recordings, and video recordings.

Additionally, we may want to use sequences in the input, affair, or indeed both, in a machine learning operation.

Still, it's challenging to perform computations on them with normal neural networks. We cannot capture the idea of order, and we don't know how numerous bumps will be demanded to represent a sequence.

Sequential data can be dealt with the sequenced neural networks which have a separate architecture.

**Recurrent Neural Network**

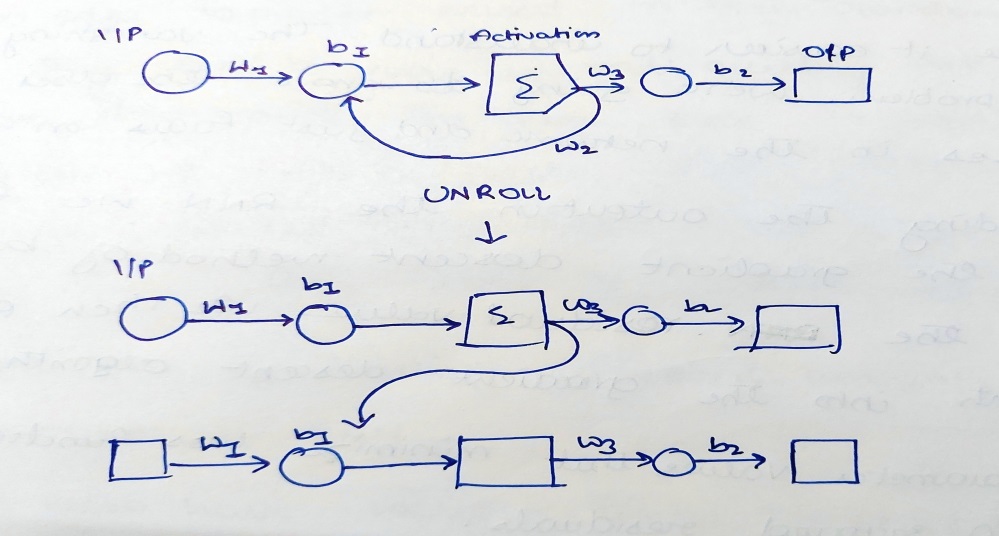
To understand Recurrent neural network we will go with an example, We need to predict the stock prices for a company using the neural network. But there is some complication using the neural network as the stocks are not a random trend they change throughout their time. Using regular neural network would cause complications in predicting the prices.

The data used in predicting the stocks would of course not be a static one, so we need a neural network which would be good at dealing the dynamic change of data this is where recurrent neural networks comes to play. Recurrent neural networks deals with the sequential data which help in dealing with the new data that enters by retraining using the feedback loops.

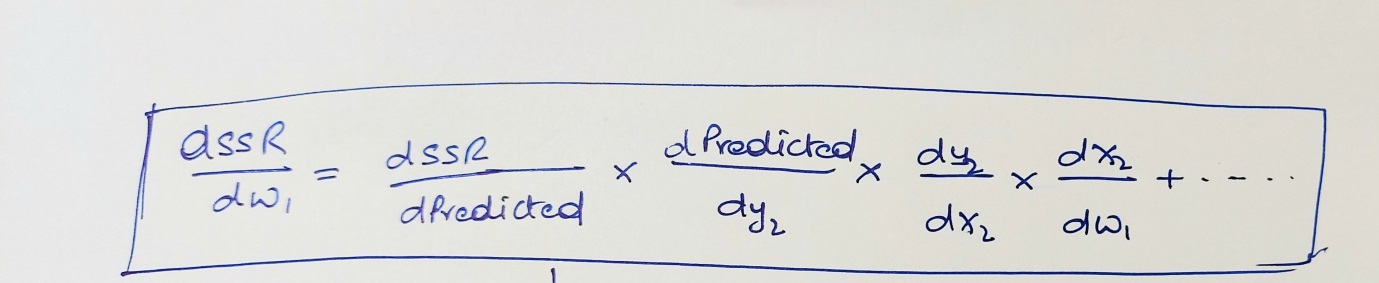
Feedback loops are the recurrent loop which calls them when they encounter a new data; these feedback loops are present at every output of the activation function. These activation functions output is fed to the previous nodes and weights are again updated in the process. This makes it possible for the recurrent neural network possible to use the sequential input values, prices collected over time to make predictions. Recurrent neural network uses the current and past states to predict the tomorrow’s case.

Unrolling the neural network would help us understand the working principle. As from the above the diagram we can see that the output of the previous node is fed to the activation function and that output is used to feed the previous node including the next node. This tells us that network can predict two values but the one that has currently occurred after the retrain with new weight. The recurrence of the node can be given as the number of conditions that a particular problem has.

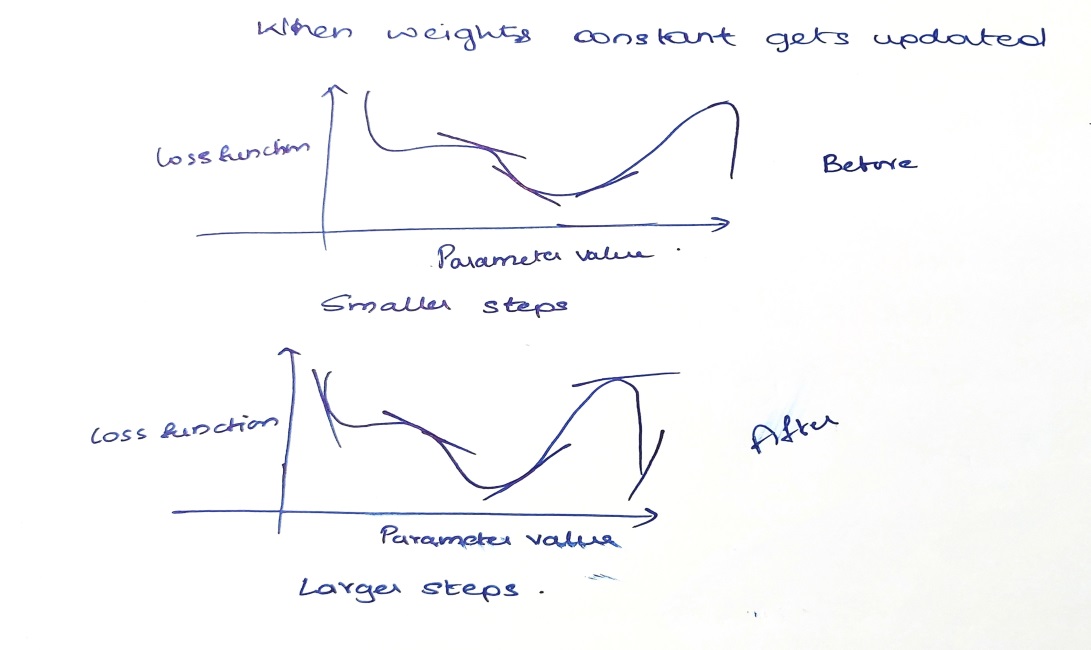
The feedback loops can also be visualized as such:



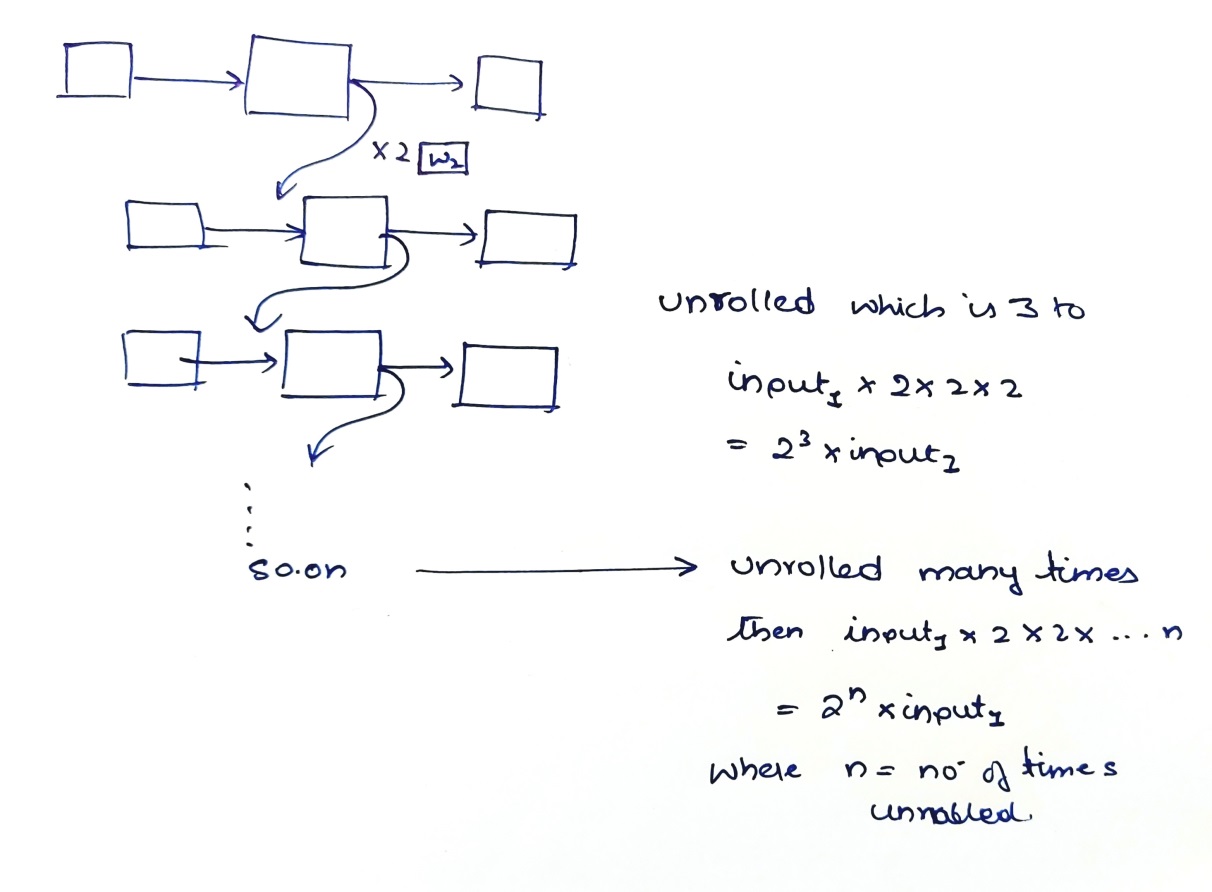
Based on these conditions the recurrent neural network’s feedback loop runs. The constants before any weight or biases should be controlled. The number of times we unroll a recurrent neural network it gets harder to train the neural network this problem is called the Vanishing Gradient Problem. As this has to with the squiggle that we copy each time, we unroll the network. To make it easier to understand the vanishing/ exploding gradient problem we’re going to focus on the feedback loops weight . Before finding the output in the RRN we first go with the residual value, we then plug those gradients into the gradient descent algorithm to find the parameter values that maximizes loss function, like sum of squared residuals.



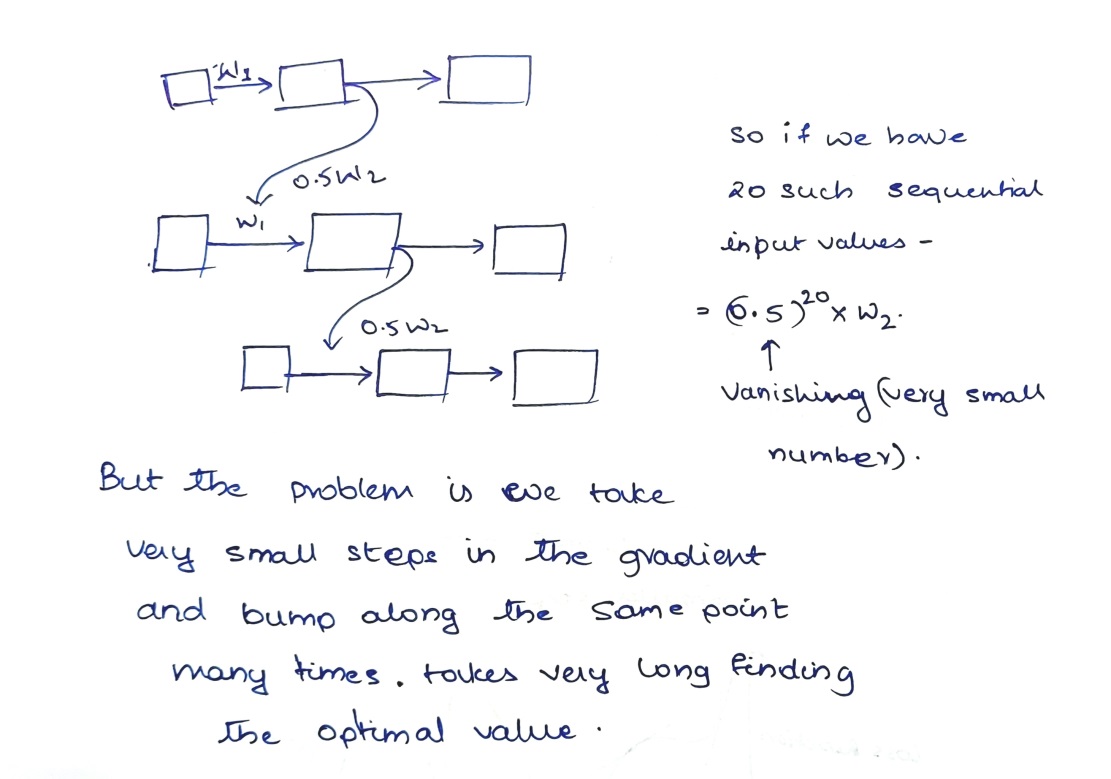
Exploding Gradient problem occurs when the gradient becomes too large, which makes the model unstable. In this case, larger error gradients accumulate, and the model weights become too large. This issue can cause longer training times and poor model performance.



From the above graph we can tell that the gradient has very larger steps and has chances to increase the error rate in the model.



The above problem can be over by keeping the constant less than 1 which makes the gradient much smaller. This results in the vanishing gradient problem.



The above problem for the vanishing gradient we cannot find the optimal k value which makes it difficult. LSTM can help in finding the solution to the above problem.

**Long Short Term Memory**

LSTM is a stepping stone to learn about the Transformers LSTM is a type of vanilla recurrent neural network that was designed to overcome the problem of exploding/ vanishing gradient problem; when unrolling the feedback networks the coefficients of the weights or biases happen to be increasing exponentially when the coefficients are greater than positive ‘1’ the exponential increase causes the recurrent neural network to explode or if the coefficient is lesser than the ‘1’ the exponential decrease tends to be much closer to zero which is known to be as the vanishing gradient problem.

To overcome this problem in recurrent neural network LSTM was introduced which had 2 different networks for storing the values, stored long back and stored shortly. This excludes the concept of feedback loops. Instead of using the same feedback loop for the events that happened long ago and events that just happened yesterday to make a prediction about tomorrow LSTM uses two separate paths to make predictions about tomorrow.

* One path is for long term memory.
* One path is for short term memory.

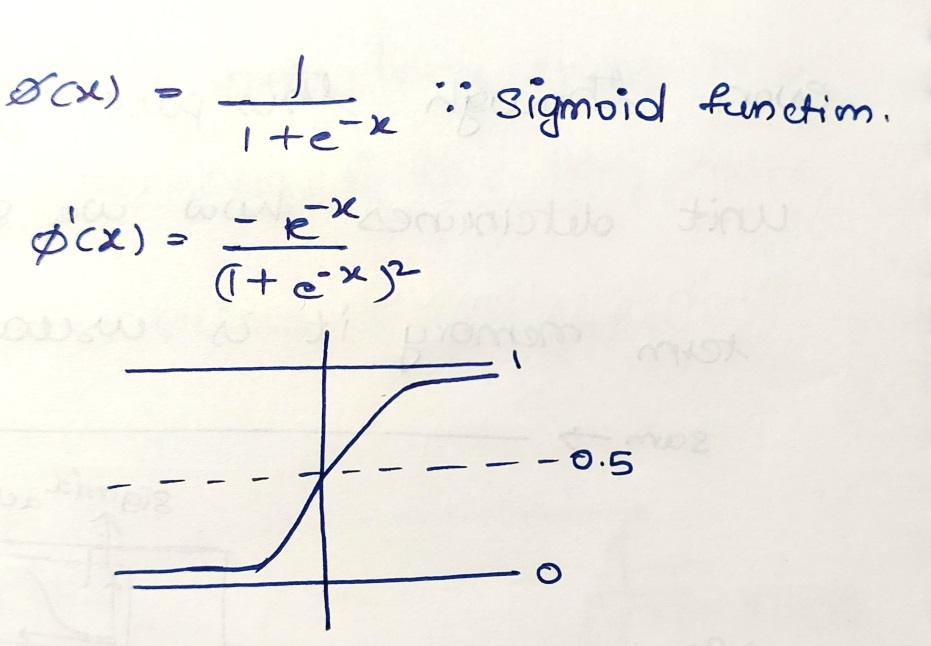
LSTM is much more of a complicated unit. LSTM uses only two activation functions they are sigmoid and tanh.

**Sigmoid Activation Function**

Sigmoid activation function is used for outputs that range in between 0 and 1. Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

The function is differentiable. That means, we can find the slope for this equation at any two points. The function is monotonic but its derivative is not.

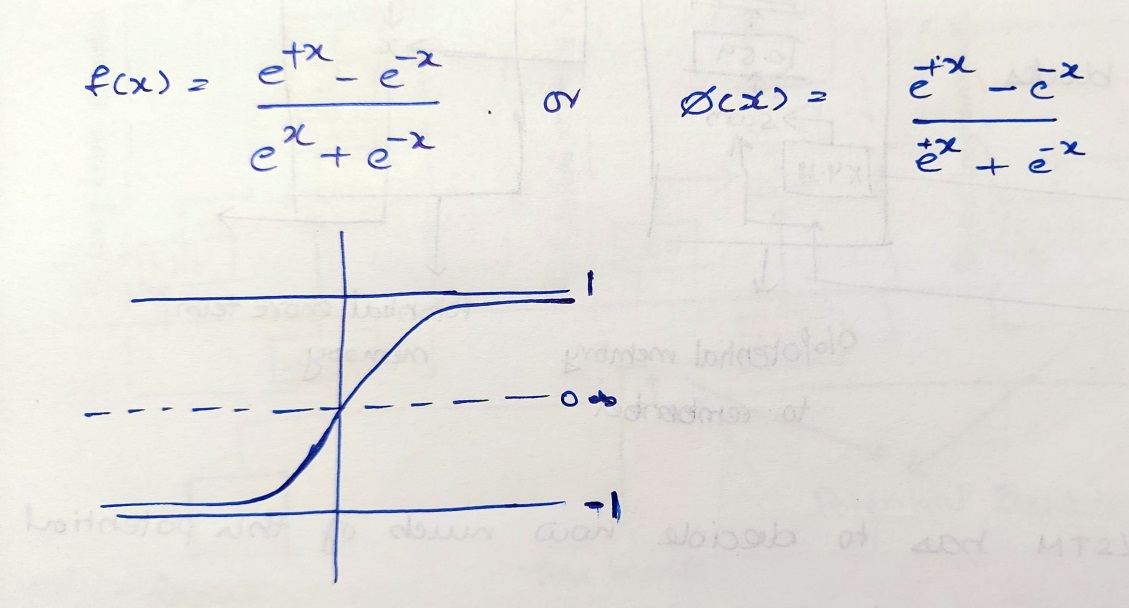
Mostly used for binary classification problem. Gives smooth gradient while converging, normalized functions, prediction either 0 or 1. It is computationally expensive, not a zero centric function, prone to vanishing gradient problem.



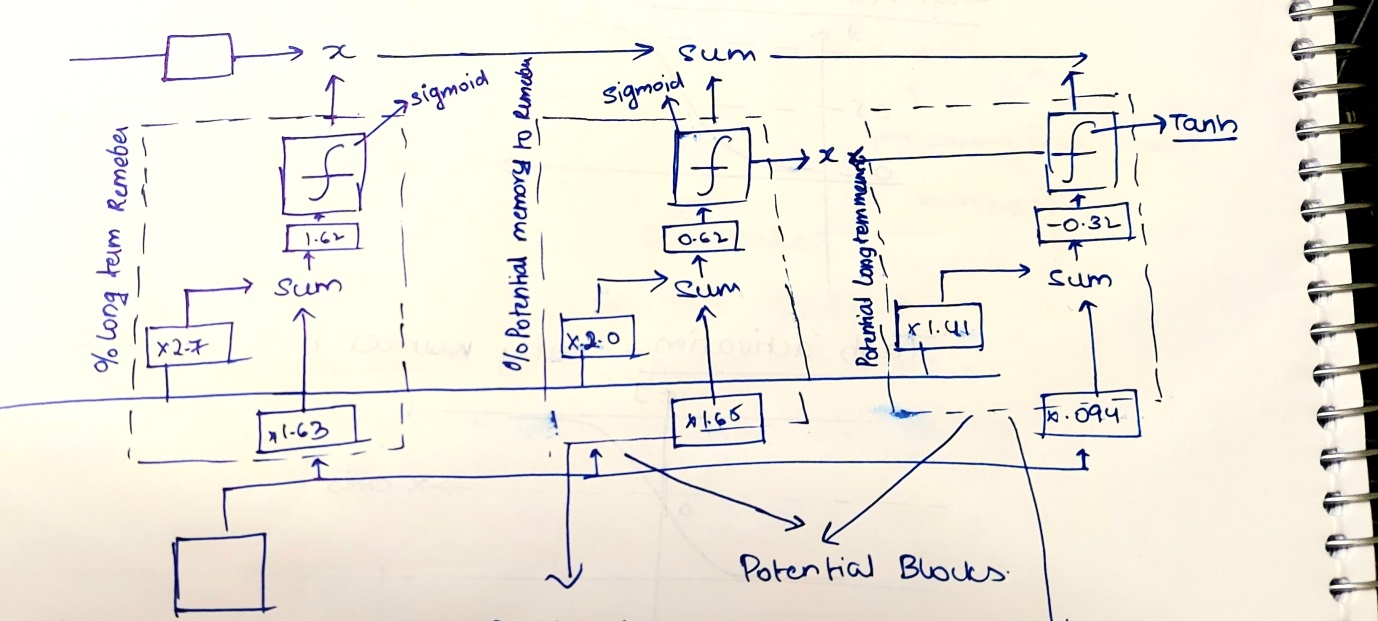
**Tanh Activation Function**

Tanh activation function is used for outputs that range in between -1 and 1. Therefore, it is especially used for models where we have to predict the probability as an output. Normally used as the input of a binary probabilistic function.

It is a zero centric unlike sigmoid, has smooth gradient converging function. But it is prone to vanishing gradient function, computationally expensive.



The LSTM diagram can be given as

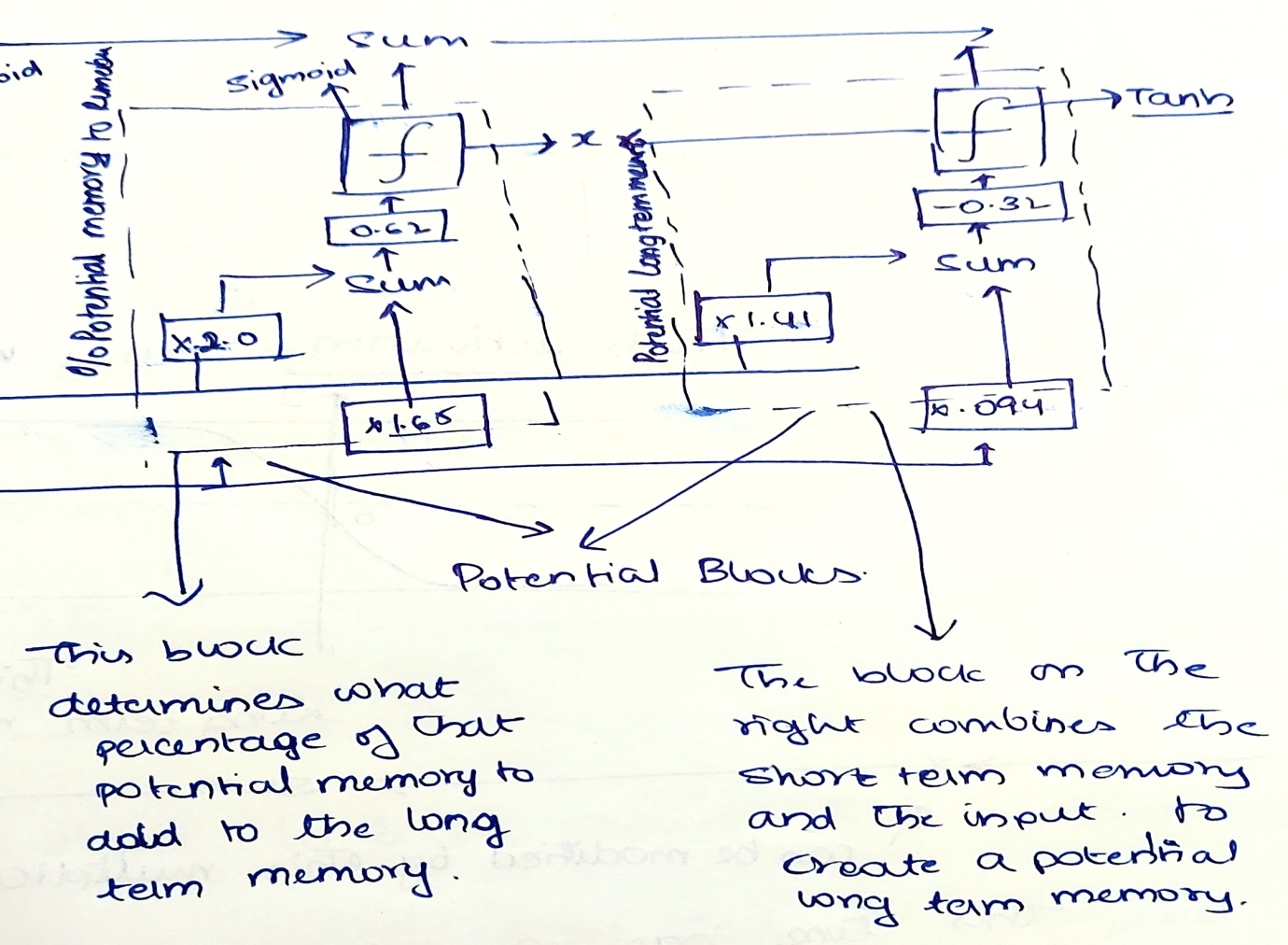


The line at the top represents the long term memory that runs across all the cell state. This line has no weights and biases through a series. The lack of weights and allows long term memories to flow through a series of unrolled units without causing the gradients to explode or vanish.

The inputs flowing nearer to the cells units containing the activation functions are called the hidden state they represent short term memory. They are directly connected to the weights that can modify them. Hence this all comes under the stage 1.

To summarize the first stage in a long short term memory unit determines what percentage of the long term memory is memorized. It is called the forget gate.

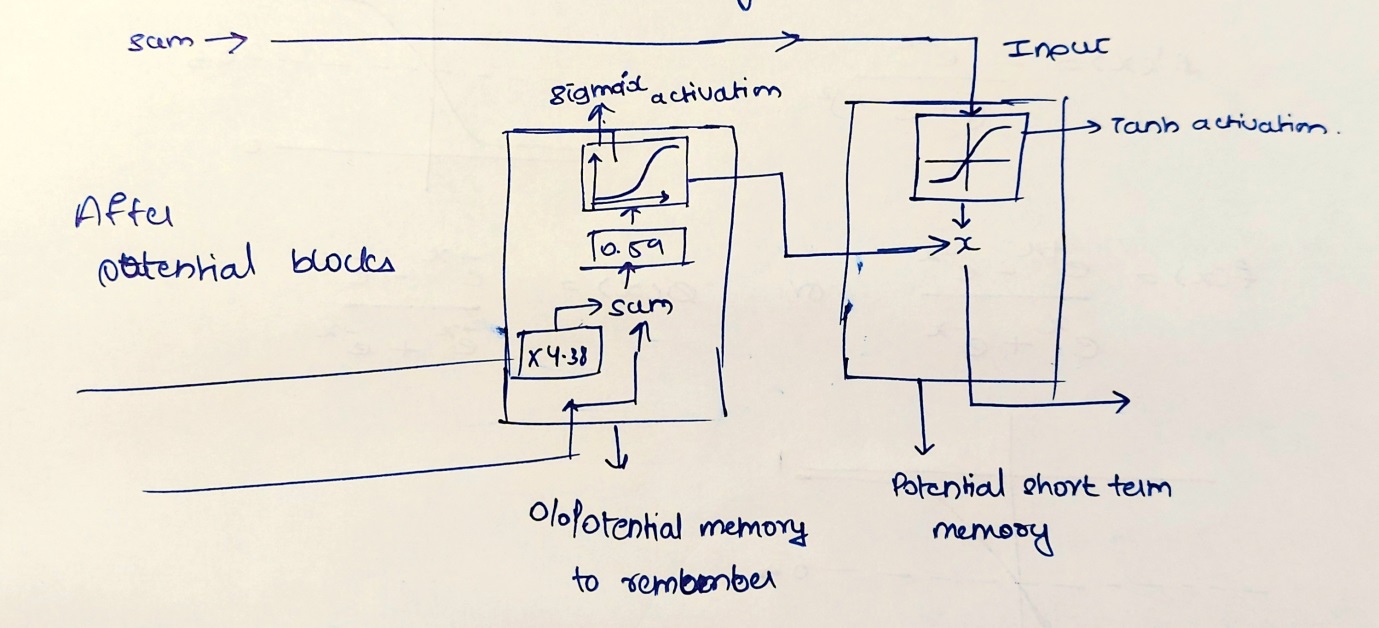
When an input is passed to the long term and short term memory lines the stage 1 calculates the percentage of long term memory remembered. Let’s say that we have passed inputs to both long and short term memory lines. Calculations are made in the long term remembered with activation function sigmoid. If the inputs are positive then there is some chance that the inputs are being remembered in the long term unit. If the inputs passed are negative in the short term unit then there is no chance that the long term memory would be remembered.



This is the potential unit the LSTM that calculates the potential for remembering the long term inputs. The block on the right combines the short term memory and the input to create a potential long term memory. The block on the left determines what percentage of that potential memory to add to the long term memory. The right potential block uses the Tanh activation function and left block uses the sigmoidal activation function. The outputs for both the blocks are combined together to form a potential input for the long term memory line which is then combined. After calculation of the activation functions the LSTM has to decide how much of this potential memory to save and this is done using the exact same method we used earlier when we determined what percentage of the long term memory to remember.

When the input value is positive then it has some remembering potential. When the input value is -10 then the potential memory to remember would be 0 when the activation function sigmoid is used which has zero potential of memory storage for long term memory. Even though this input of the long short term memory unit determines how we should update the long term memory, it is usually called the input gate. This all goes under the stage 2 where we have to decide on the potential for the long term memory.

The stage 3 includes the potential for calculating the short term and giving out the output.



This is similar to the steps that were involved in calculation of the potential for long term memory. Now the LSTM has to decide how much of the potential short term memory to pass on.

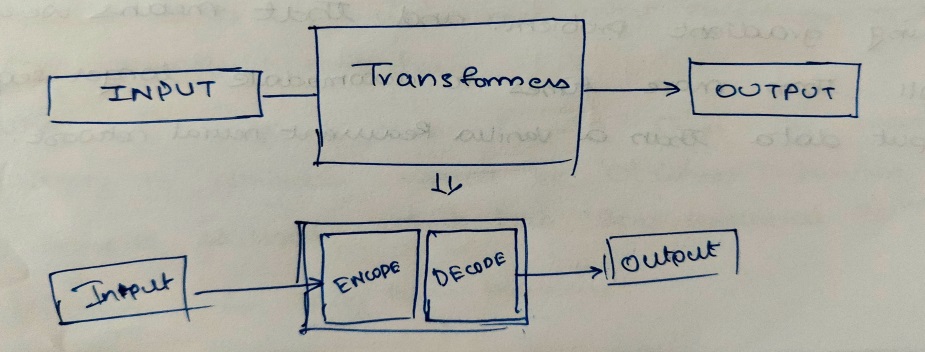
The output of the this potential short term memory is the same output for the LSTM also. Because the new short term memory is the output from the entire LSTM unit this stage is called the output gate.

Long short term memory networks avoid the exploding/ vanishing gradient problem and that means we can unroll them more and more number of times to accommodate longer sequences of input data than a vanilla recurrent neural network. Hence this was all about the LSTM. This gives the new concept called the Transformers.

**Transformers**

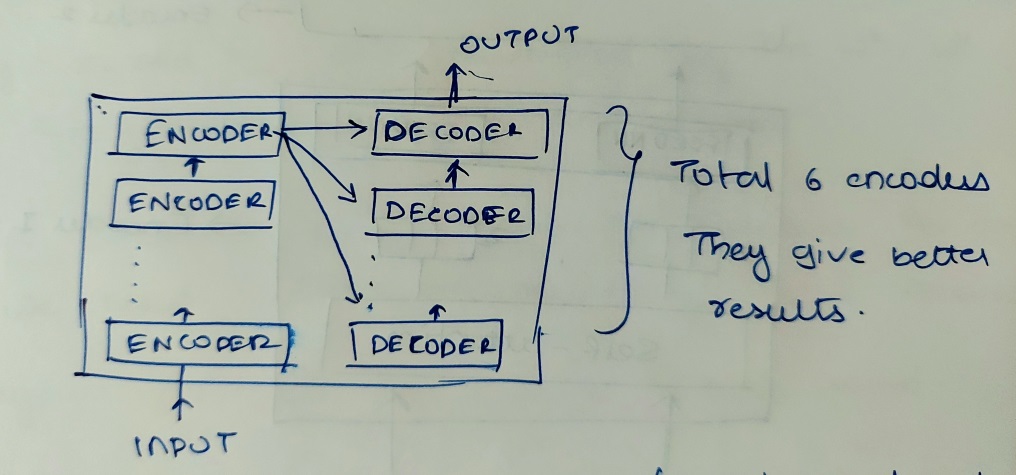
Transformers are the encoding and decoding composed units that are used to convert the words that are passed to it. It is a model that has uses attention to speed with which these models can be trained. The transformer outperforms the google neural machine translation model in specific tasks. Transformer uses the parallelization. It is the fact that google clouds recommendation also uses the transformer model as a reference to their cloud tpu offering.

Transformer has a encoding unit and decoding unit or components. Encoders take the inputs and decoders will decode them to give the output.



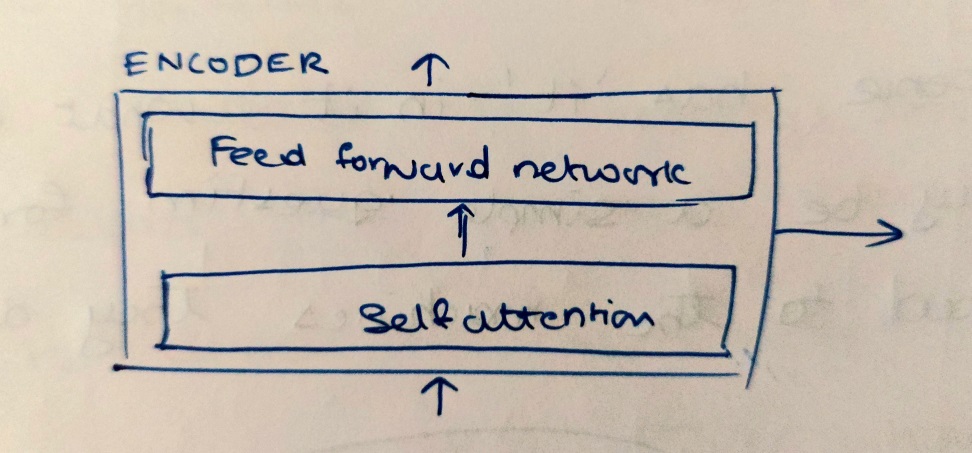
Number of encoders and decoders can be changed but the results will not be as better as they are currently. Using six encoders and decoders perform better. This can also be called as the hyper-parameter.

Each of these encoders are broken down to see the internal structure. They all are identical in structure (yet they do not share the same weights).



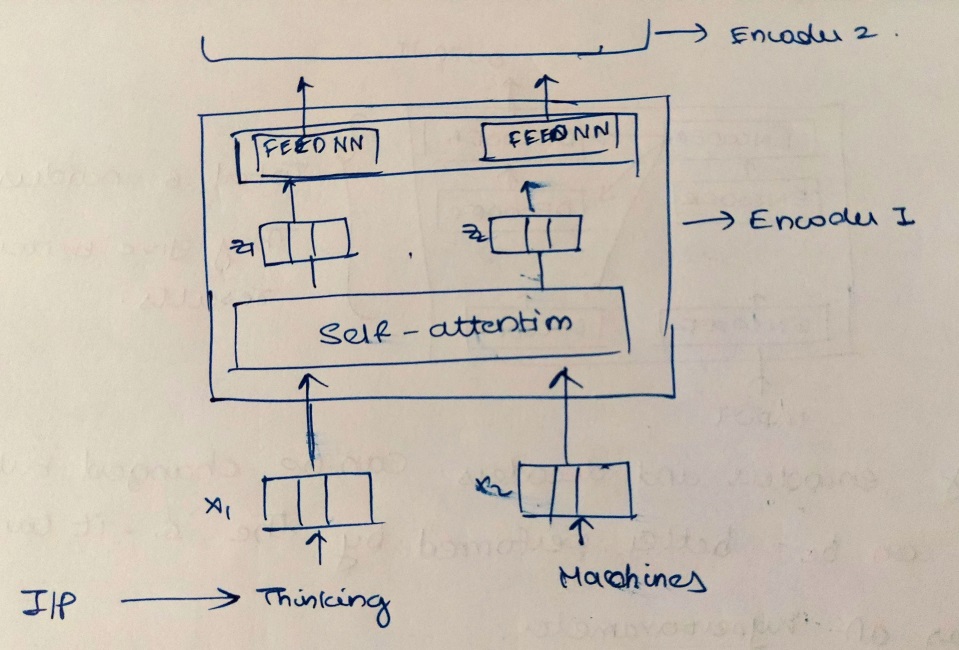
**Self Attention Layer**

The inputs are converted to some dimension using the word to vector converter which is 512 bytes. These converted dimensions are given to the self attention layer. All the words are passed parallel.

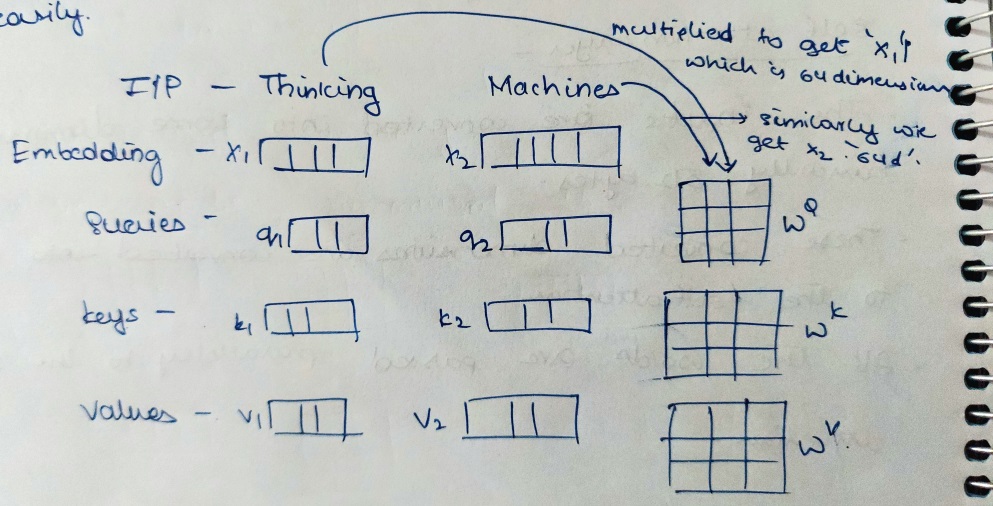


Let us take an example – ‘The animal didn’t cross the street because it was too tired.’

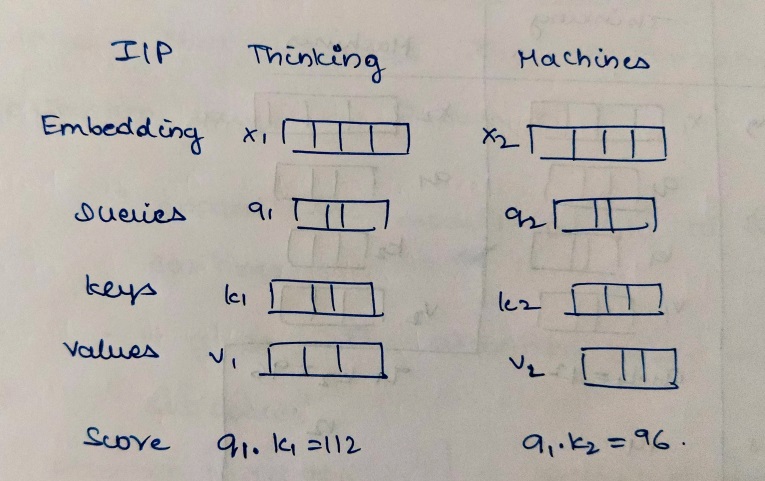
The above sentence has ‘it’ in it. What does this refer to? It may be a simple question for us humans but when asked to the machines they don’t understand it easily. So the converted words are sent in the form of x1 and x2 to the layer. These input vectors are then multiplied to the attentions to get the queries, keys, values. Hence these can be called as the parameters. After the generation of queries, key, values for each word we need to then multiply these queries and keys to get the score of each word.



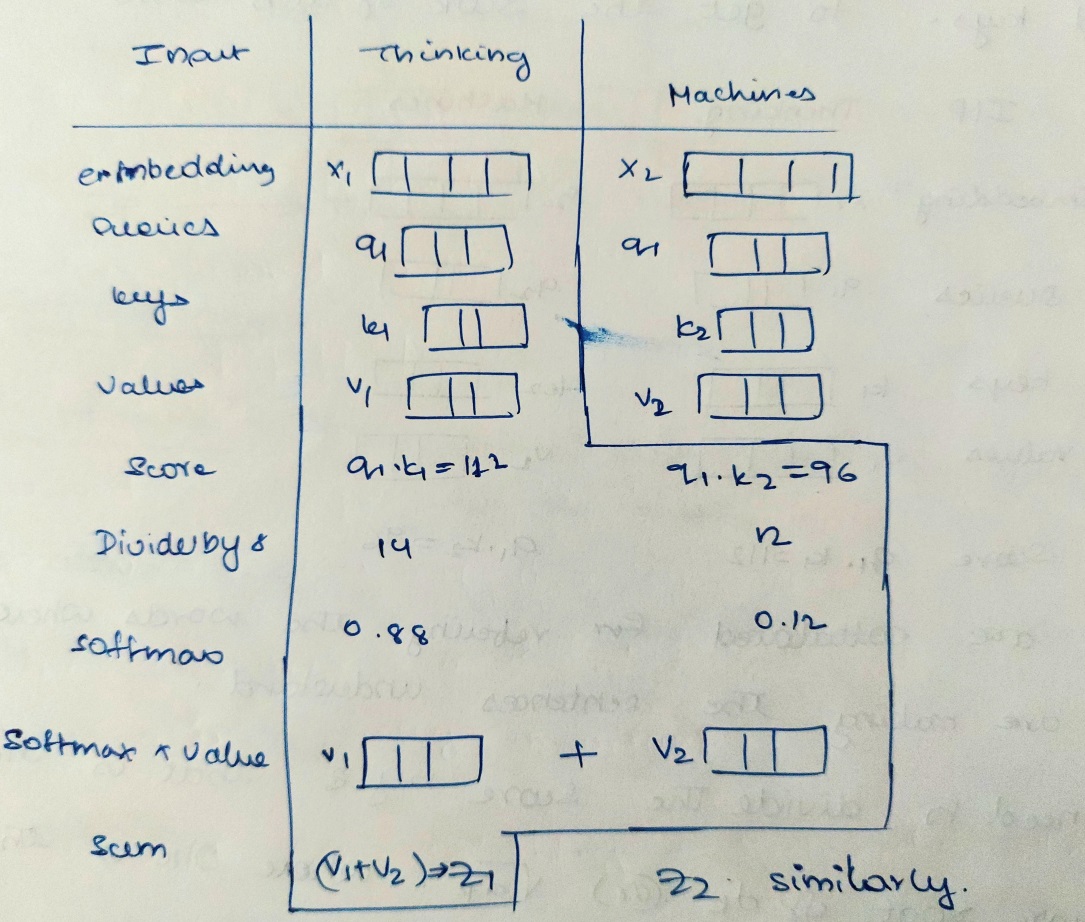
Score are calculated for referring the words where they are making the sentences understand. We need to divide the score by 8. The number 8 is derived from the dimensions of the queries, keys, values.



These parameters have the same dimensions which is 64 bytes. Denoted by dk. this is then passed to the activation function softmax where we are trying to find the most important word. All the softmax values when added to they give us the answer ‘1’.



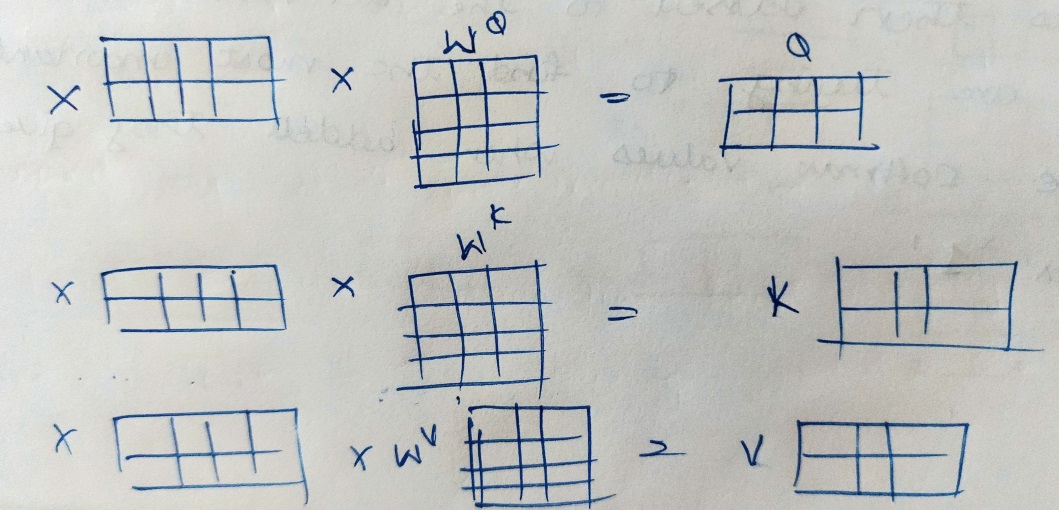
Then finally adding the calculated values for each of the words can give us the sum which is then sent to the feed forward network.



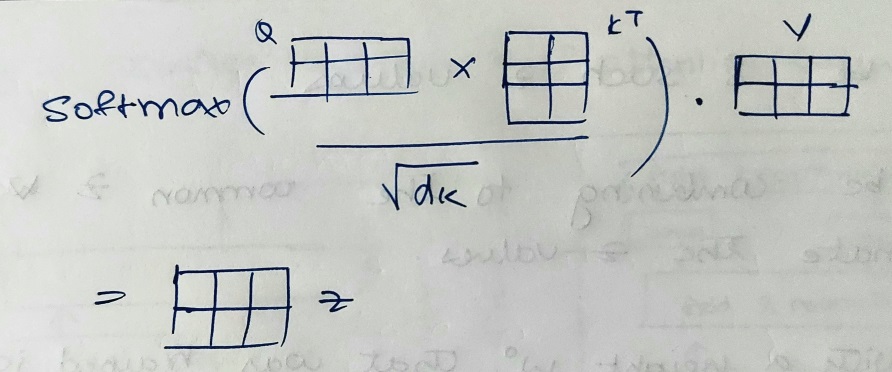
All the above steps can be summarized from the below diagram.

Here weights are randomly initialized.

X denotes the word to vec dimension which is 512 bytes. This X is then multiplied with the wQ, wK, wV to get the queries, keys, values matrix.



Then the softmax is calculated and then the softmax’s output is multiplied with the values to get the Z matrix.



**Multiheaded attention**

Is a layer with self attention layer that improves the performance of the attention layer by two ways.

* Expand the model’s ability to focus on different positions.
* It gives the attention layer multiple representation subspaces.

Using the multiheaded attention we can find the importance of each word in the given sentences. We will be using 8 such attention heads. These multiheaded attentions will then give us with 8 different z values. We will be combining these values to or concatenating them into on z vector. Then multiply with a weight w0 that was trained jointly with the model. The result would be the z-matrix that captures information from all the attention heads. We can send this forward to the feedforward neural network.

**Positional encoding**

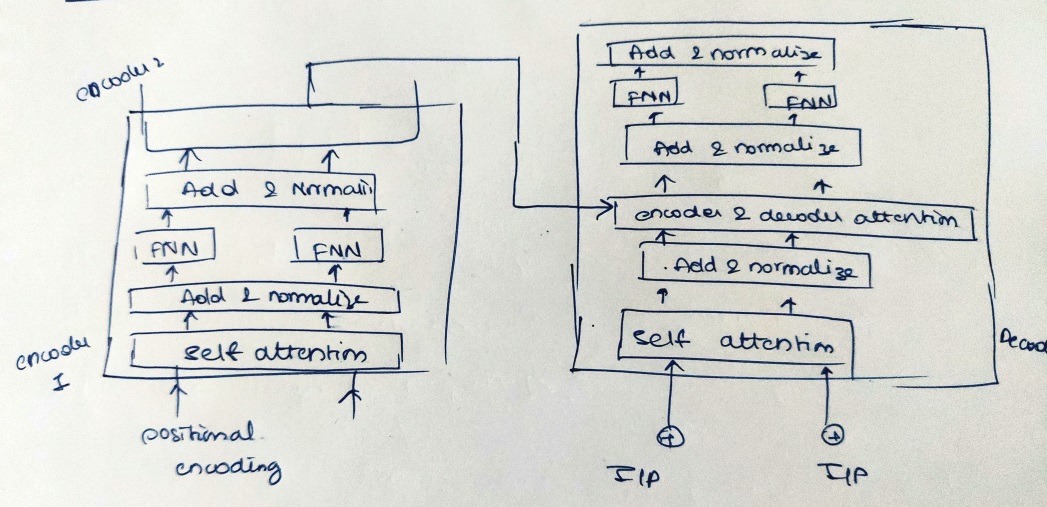
It tells us about the words that are closer to each other. It is a distance matrix that tells the distance between the words. Ordering of the words is important. We will be then adding the positional encodings with the embedding that gives us the time signaling.

After the self attention outputs are calculated they are passed to the Add & normalize layer. Each encoder has a residual connection around it and followed by a layer normalization step.

Residual connection – is a connection that is directly connected to the output without having wasted on unnecessary calculations on the layers between.

**Decoder**

Decoder takes in the inputs and these inputs are words that are referring to the inputs that were given to the encoder. Decoder’s architecture is very much similar to the encoder but there is one additional layer that is called the encoder-decoder attention that takes in the input from the encoder and combines the inputs of the decoder and then normalizes it and passed to the feed forward neural network. Here the inputs are passes one by one so that the mapping is correct.



Then the final two layers they are the linear and the softmax layers that takes the inputs from the decoder which are basically the scores, these scores are turned into probabilities. The cell with the highest probabilities is chosen, and the word associated with it is produced as the output for this time step. All the steps are continued till it encounters the ‘end to sentence’.

**BERT** is the implementation of the transformers. That uses bidirectional encoders and decoders that create representations from the transformers. BERT is given the sentences, but some words are taken out, those words are to be predicted by the BERT that is best suited. This is repeated; eventually the algorithm gets better at predicting those sentences.

**Conclusion**

Text summarization is helpful in providing useful information from the text documents given, spending less time in reading and getting useful information. News articles, fact sheets and mails fall under these categories. Sentences builds upon the previous, text summarization may not be helpful. Research journals, medical text are a good example where text summarizations are not very accurate. The summarized texts might help us giving the summarized results but they might lack the style and tone of the text that the author tries to convey.

Hence these are all various extractive methods that can be used for the text summarization. The mentioned methods are easy as compared to the abstractive text summarization, for better conclusion on extractive text summarization. Extractive text summarization is one of the methods for text summarization which uses two types of methods based on the topic representation and indicator representation.

ML Algorithms like SVM, Naïve Bayes, logistic regression can be used but apart from these machine learning models Hidden markov model and conditional random fields have outperformed the above mentioned models. These models have higher intuition levels and are best at performing the NLP tasks as they use probabilistic calculation based on the current scenarios to predict the upcoming scenarios, hence these models are best suited for the text summarization in indicator representation methods

Abstractive text summarization uses various such implementations for preparing the summary for the documents that are given this technique is difficult and uses little bit of understanding in the deep learning concepts. It is an unsupervised approach that uses words that are not in the documents plays a guessing game that has accurate meaning with the documents. Its uses RNN (recurrent neural network) which is similar to a normal neural network it is used in the prediction of the next states data based on the previous states. But it has an issue of exploding/ vanishing gradient problem that is overcome by the LSTM. This uses two different lines for remembering the long and short term memory. BERT uses the concept of transformers that uses encoder and decoder to convert the inputs. It is a model that has uses attention to speed with which these models can be trained. BERT is given the sentences, but some words are taken out, those words are to be predicted by the BERT that is best suited. This is repeated; eventually the algorithm gets better at predicting those sentences.

**References**

[Abstractive Text Summarization with Natural Language Processing | by Mars Xiang | The Startup | Medium](https://medium.com/swlh/abstractive-text-summarization-with-nlp-ec3924c0b1d5)

[Understanding Automatic Text Summarization-2: Abstractive Methods | by Abhijit Roy | Towards Data Science](https://towardsdatascience.com/understanding-automatic-text-summarization-2-abstractive-methods-7099fa8656fe)

[Types of Text Summarization: Extractive and Abstractive Summarization Basics - Turbolab Technologies](https://turbolab.in/types-of-text-summarization-extractive-and-abstractive-summarization-basics/)

[Approaches to Text Summarization: An Overview - KDnuggets](https://www.kdnuggets.com/2019/01/approaches-text-summarization-overview.html)

[Abstractive Summarization Using Pegasus - Turbolab Technologies](https://turbolab.in/abstractive-summarization-using-pegasus/)

[Abstractive Summarization Using Google's T5 - Turbolab Technologies](https://turbolab.in/abstractive-summarization-using-googles-t5/)

<http://jalammar.github.io/illustrated-transformer/>

<https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>

<https://analyticsindiamag.com/a-guide-to-hidden-markov-model-and-its-applications-in-nlp/#:~:text=A%20Hidden%20Markov%20Model%20(HMM,which%20are%20not%20directly%20observable.&text=By%20Yugesh%20Verma-,A%20Hidden%20Markov%20Model%20(HMM)%20is%20a%20statistical%20model%20which,also%20used%20in%20machine%20learning.>

<https://towardsdatascience.com/conditional-random-fields-explained-e5b8256da776>

<https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>

<https://en.wikipedia.org/wiki/Latent_semantic_analysis>

**google colab link for latent semantic analysis -**

<https://colab.research.google.com/drive/19iEmYXesgfxJfwwghzGWxQNDh1WwWLI6?usp=sharing>