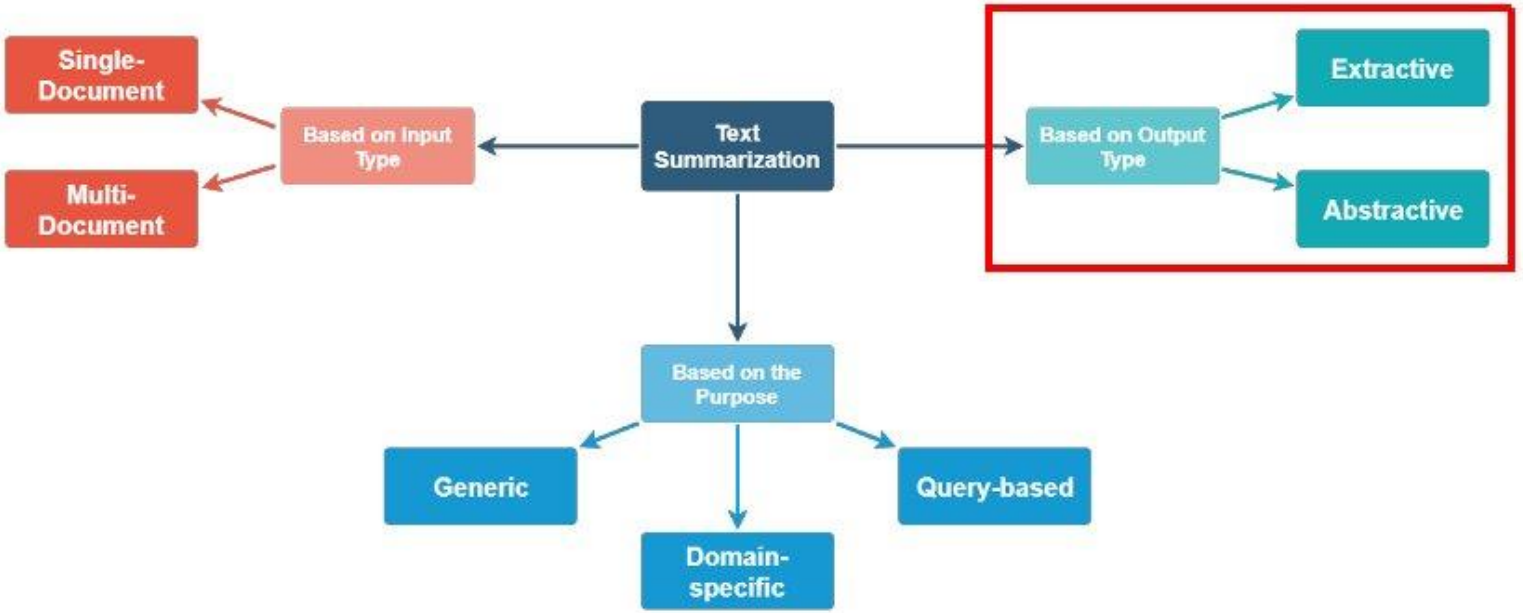
Document Summarization using Neural Networks

Manu Hegde, VIII ‘A’, CSE
1DA15CS067

1. Literature Survey and Existing Work

1.1 Paradigms in Document/Text summarization



1.2 Major types of Document Summarization

- **Extractive document summarization** is generating summaries by selecting phrases, sentences which can effectively represent the gist of the document.

- **Abstractive document summarization** is generating summaries that may contain words other than those present in the original document, this approach can yield sentences not written by the author of the document.

Abstractive summarization requires that the summarizer be able to represent a given set of sentences in an intermediate representation using which a summary can be derived. Hence until 2013, efforts for summarization were mostly Extractive.

The first major success in abstractive summarization was enabled by the creation of high dimensional word embeddings*[1]* such that words were represented as vectors and whose dot product showed their similarity.

This led to development of recurrent neural network based encoder decoder (Sequence to Sequence) models *[2]*.

[1] - Word2Vec https://arxiv.org/abs/1301.3781,
[2] - Sequence to Sequence Learning with Neural Networks https://arxiv.org/abs/1409.3215

1.3 Approaches to Extractive Document Summarization

Extractive document summarization has been mostly done by complex systems carefully engineered with manual feature selection and pre processing based on language specific patterns.
Mostly they belonged to following categories:

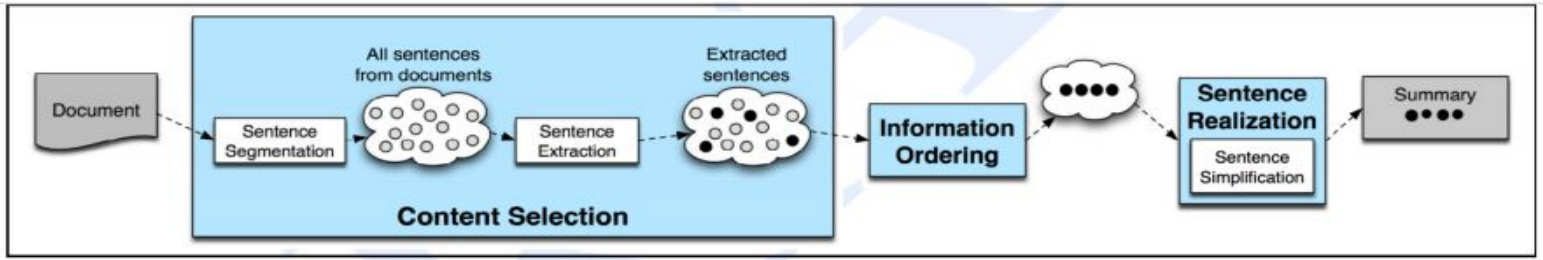
Sentence scoring functions:

- Based on presence of topic keywords
- Features such as where the sentence appears in the document

Graph-based algorithms

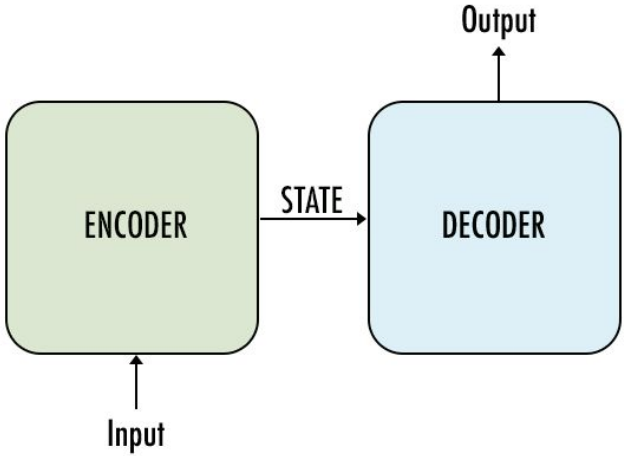
- view the document as a set of sentences (nodes), with edges between each sentence pair
- Edge weight is proportional to sentence similarity
- Use graph algorithms to identify sentences which are central in the graph

A general architecture of a pre neural era extractive summarizer



1.4 Approaches to Abstractive Document Summarization

Abstractive document summarization has been achieved using various types of models.
However, all major models are based on Encoder-Decoder architecture. The Encoder-Decoder architecture is a common base technology for various other tasks like Translation, Text classification, etc.



Encoder: Takes text as input, in the form of word embeddings (like Word2Vec), optionally adds position encoding for words and outputs a latent state/representation for given input.

Latent State/ Representation is an intermediate, fixed length value (usually), which is capable of meaningfully representing the contents of the input.

Decoder: Takes the latent representation as input and generates the final output i.e the summary.

Recurrent Bi-LSTM/GRU based Seq2Seq, Seq2Seq with Attention Mechanism and Transformers with Self Attention are three major and widely used architectures for abstractive summarization.

A Neural Attention Model for Abstractive Sentence Summarization - https://arxiv.org/abs/1509.00685
LONG SHORT-TERM MEMORY - https://www.bioinf.jku.at/publications/older/2604.pdf
Gated Recurrent Units - https://arxiv.org/abs/1412.3555

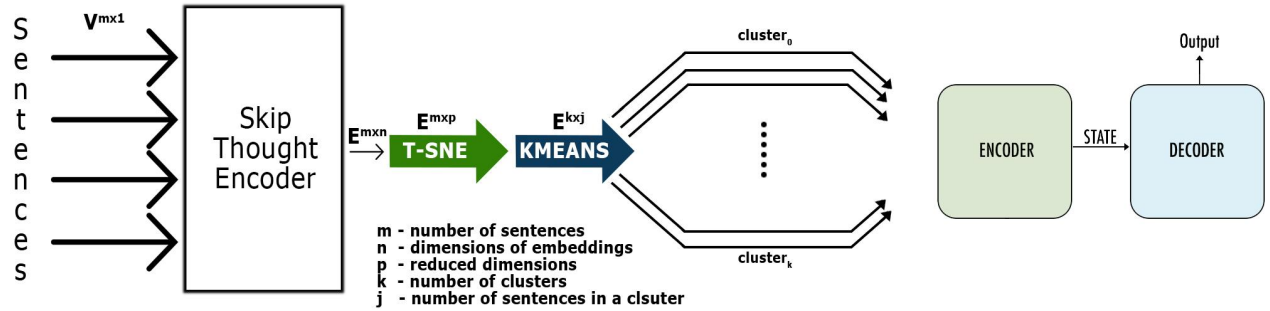
2. Document Summarization using Skip Thought vectors and content based clustering

2.1 Introduction

The model attempts to segregate sentences based on their content as well as based on the desired length of summary. This segregation leads to formation of sentence clusters. One summary per cluster is obtained. When these individual summaries are combined, they form the document summary.

The mode is as follows

Figure 1.0 - Model Architecture



2.2 Algorithm

Algorithm for the model in the figure 1.0

```
Algorithm Summarize
Input : Document
Output : Summary

S <- summarize_document(D)
  p <- number of sentences in document
  q <- output size of skipthought encoder
r <- reduced dimensions after TSNE
E_pXq <- Array(p, q)
Summary <- Array()
for s_i ϵ document
    e_i <- skipthoughts(s_i)
    E_pXq.append(e_i)
E_pXr <- tsne(E_pXq, dimensions = r,
perplexity = 30, iters = 1000)
num_of_clusters <- cluster_strategy(p, E_pXr)
clusters <- KMeans(E_pXr, num_of_clusters)
for c_i ϵ clusters
    summ_i <- summarize_cluster(c_i)
    Summary.append(summ_i)
return Summary

cluster_strategy(p, E_pXr)
    strategy <- square_root
    if strategy = square_root
        return sqrt(p)
    else
        return elbow_method(E_pXr)

summarize_cluster(cluster)
  mini_doc <- str_concat(cluster)
  hidden_states <- encoder(mini_doc)
  decoder_output <- attention_decoder(
  hidden_states)
  Summary <- Array()
  for h_j ϵ hidden_states
  summ_i <- greedy_decoder(h_j)
  Summary.append(summ_i)
```

2.3 Algorithm Explanation

The Algorithm operates as follows.

- Obtain embeddings for every sentence.
- Reduce dimensionality of embeddings using T-SNE.
- Cluster the embeddings using K-Means.
- Obtain one summary for each cluster separately.

A. Obtain Embeddings for every Sentence

Skip-Thought encoder to obtain fixed length sentence embeddings, for sentences of varying length. This can step is necessary to obtain meaningful representation of a sentence in form of a vector i.e an embedding in a higher dimension, such that the sentences with similar meaning are mapped close to each other and vice versa. Also this space is such that a dot product between two sentences can be used to obtain a measure of their similarity or dissimilarity. The length of these vectors is 4800. Using Skipthought sentence embeddings can solve the misunderstanding of sentences due to UNKs, because of its vocabulary expansion capability to handle previously unknown words.

B. Reduce Dimensionality of Embeddings using T-SNE

T-SNE (t-Distributed Stochastic Neighbour Embedding)[9] is widely known algorithm for dimensionality reduction. The T-SNE algorithm is designed to project a set of points in higher dimension to a lower dimension, yet trying to minimize the loss in pairwise distances between all set of input points.

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

Where is the probability density under a Gaussian centered at .
Since the sentence embeddings each are of length 4800. Effectively projecting them to lower dimension without loss of information is necessary and would lead to better performance. This would result in much better result from K-Means Clustering. Hence T-SNE was chosen over other algorithms like SVD or PCA.

In this step the algorithm uses TSNE to project the high dimensional sentence embeddings to a lower dimension while trying to maintain pairwise distances between the sentence embeddings. This allows for better performance due to lesser but equivalent information being used.

Since embeddings are basically points in higher dimension, the topography of the higher dimension, if can be represented in a lower dimension, we can say the pairwise distances between points are preserved.

B. Reduce Dimensionality of Embeddings using T-SNE

As in figure 2.0, T-SNE is very well known for effectively reducing the dimensionality of data without notable loss of information.

Figure 2.0 is a 3-d plot of images of digits 0-9 that are fed to T-SNE which reduces them to 3 dimensions, resulting in (x,y,z) coordinates for each image.

It is not to be mistaken that data is segregated using T-SNE. It only shows the distribution of data in a very effective way.

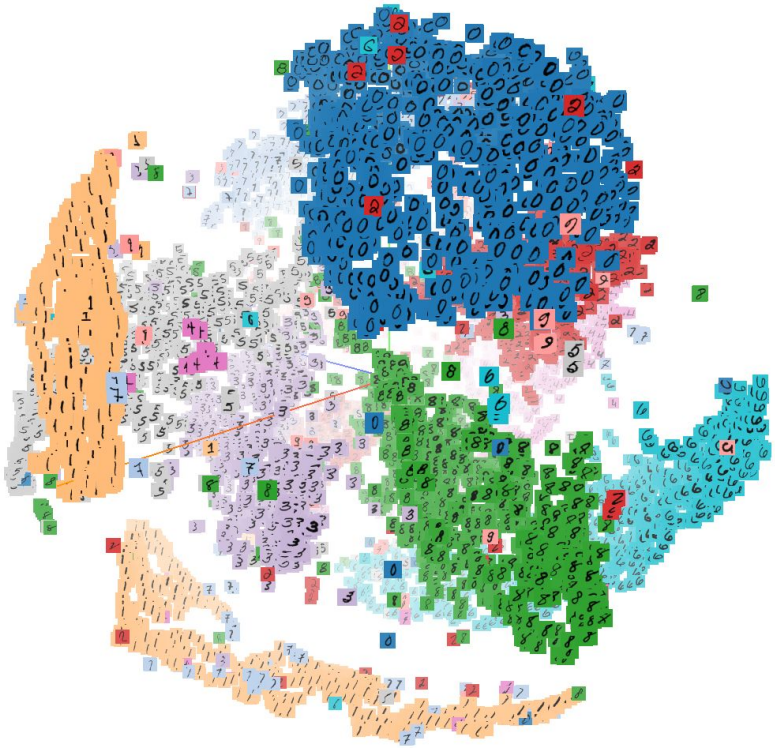
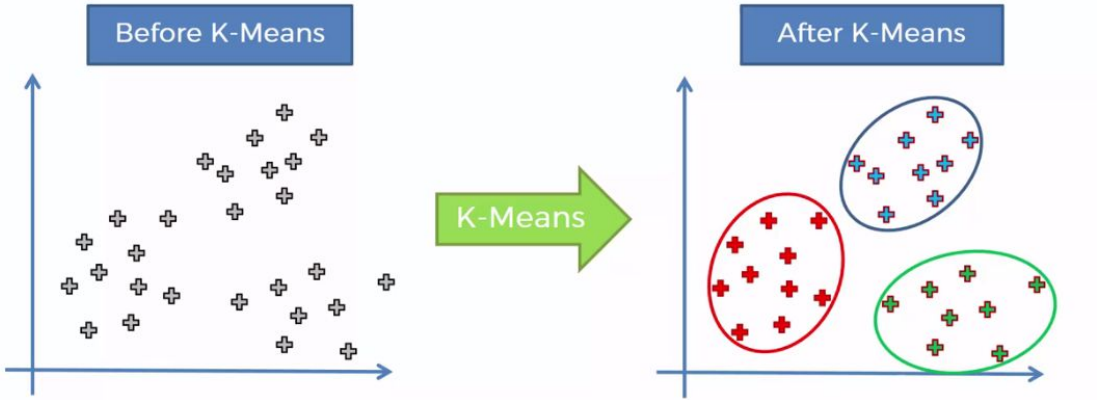


Figure 2.0

C.Cluster the embeddings using K-Means

K Means is used for clustering the sentence embeddings. The number of clusters is directly related to the desired length of summary. Usually this number is chosen as square root of the number of sentences in the document. This can solve the problem of loss of context as well as requirement of large computational resource for large documents. Here, each cluster is formed such that all sentences in a cluster have similar meaning or are very closely related to each other. Hence the document is effectively divided based on content similarity.

K Means tries to group data into clusters. However it does this iteratively and tries to find the appropriate grouping of data, i.e it assumes a given number of clusters to be present in the data and then iteratively shifts the centroid of each cluster until convergence.

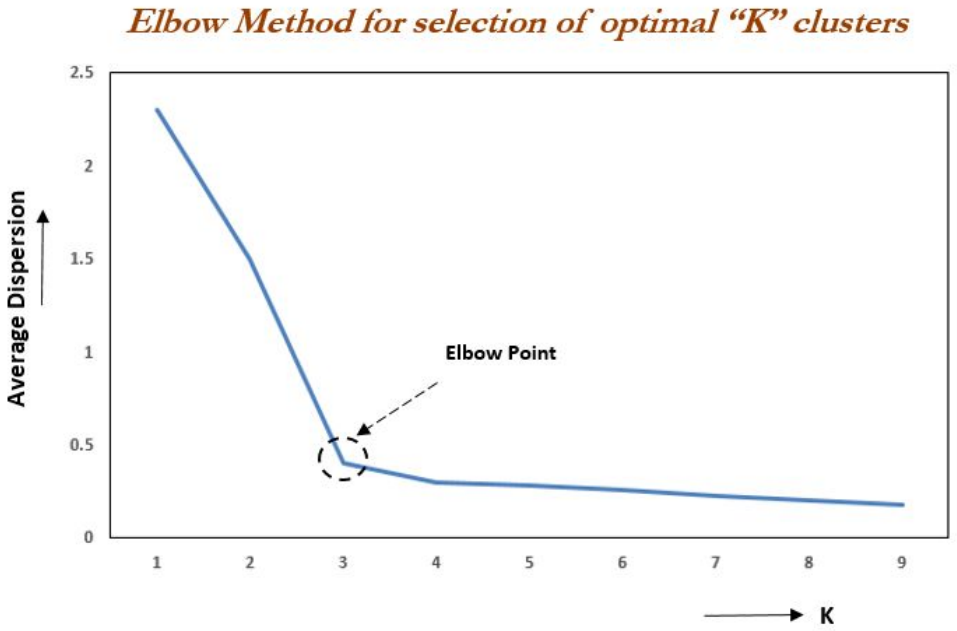


C.Cluster the embeddings using K-Means

However, in cases when the data is not clearly separable, elbow method needs to be used to determine the appropriate number of clusters the data should be divided into.

This is done by iteratively running KMeans with increasing the number of clusters to be formed i.e KMeans will divide data into more clusters. At one given point, the average dispersion (inverse of density) of points in a cluster stops decreasing drastically.

That determines the number of clusters in the data.



D.Obtain one summary for each cluster separately

Finally, set of sentences in each cluster are fed to an encoder decoder model. This helps in solving the problem of coverage tracking. Also since all sentences in a cluster have similar meaning or are closely related to each other, the encoder decoder model would process sentences of same or
similar topic. This would allow for a model to be used that is computationally less intensive due to smaller input size and smaller context. There are different variants of attention. Two main variants are Additive and Multiplicative attention.
The following equation describes Multiplicative attention.

$$e_i = v^T tanh(W_1 h_i + W_2 s) \epsilon R$$

Where e_i is the attention vector, W_1, W_2, are weight matrices and v is a weight vector.

Segregation of context or topic is of great significance since assumptions can be made in case of unknown words. An encoder decoder model consisting of Bidirectional LSTM and attention decoder would generate the final summary for each cluster independently. Additionally Beam search could be incorporated for better results as opposed to Greedy decoding that is in use currently.

Applications

- Summarizers can be used for quick understanding of lengthy academic papers.
- It can also be used for obtaining simplified explanations of complex scientific literature.
- For generation of news feed of news articles.
- For summarizing tenders and legal documents including end user licence agreements which are known to be very long.
- For quick comparison of documents.
- For Question-Answering on large amount of text.