Indian Institute of Information Technology, Nagpur

Text Extraction and Summarization

Abstractive Text Summarization

- Transformer-based models: Transformer-based models like BERT, GPT-2, and T5 have achieved state-of-the-art results.These models use a self-attention mechanism to process the input sequence using encoder and generate a summary using decoders.
- Pre-trained Language Models(Transfer Learning): Pre-trained Language Models like GPT-3 and T5 have also shown promising results in abstractive summarization. These models are trained on massive amounts of text data and can generate high-quality summaries with minimal fine-tuning on a specific task.
- A Comprehensive Survey of Abstractive Text Summarization Based on Deep Learning

General Architecture

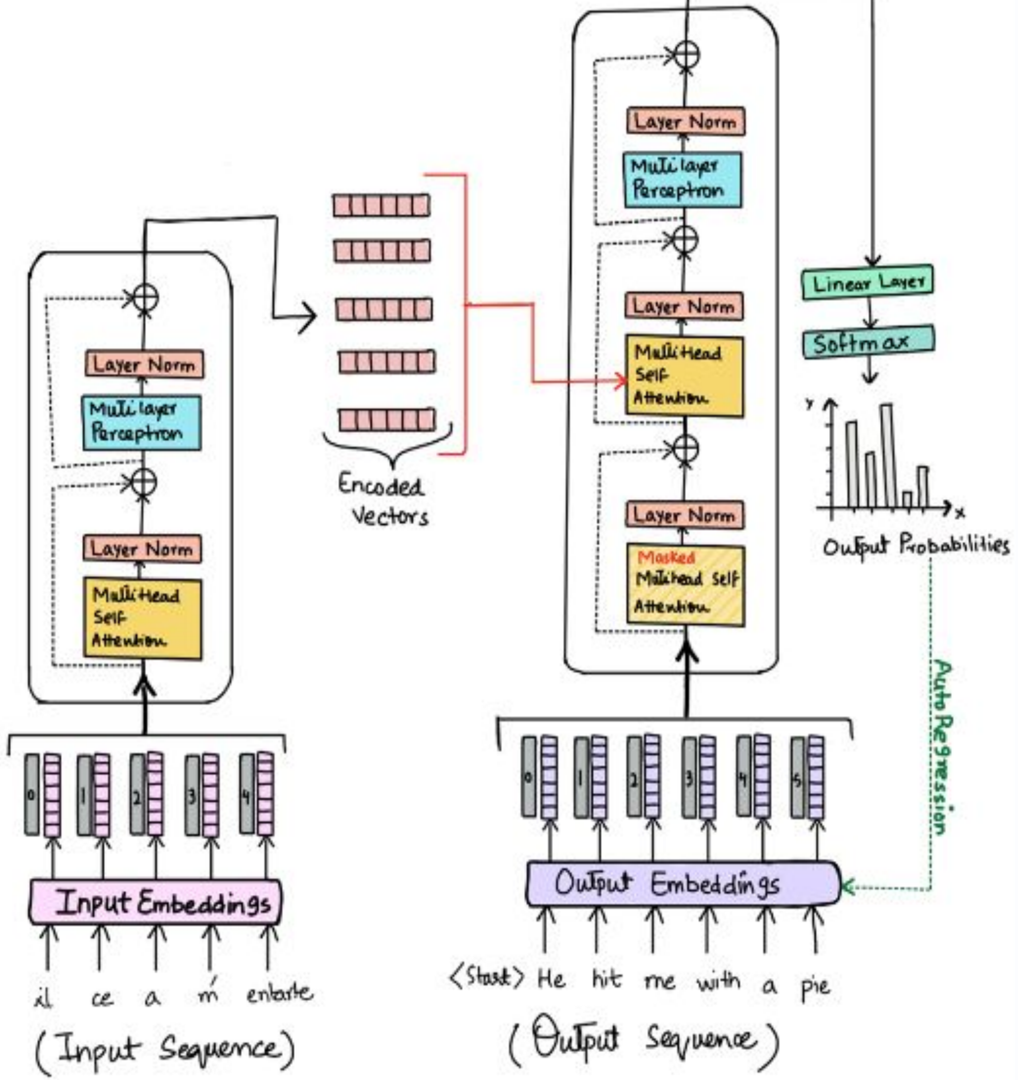
A general architecture of DL-based ABS, which is mainly composed of three steps:

- preprocessing
- semantic understanding
- summary generation

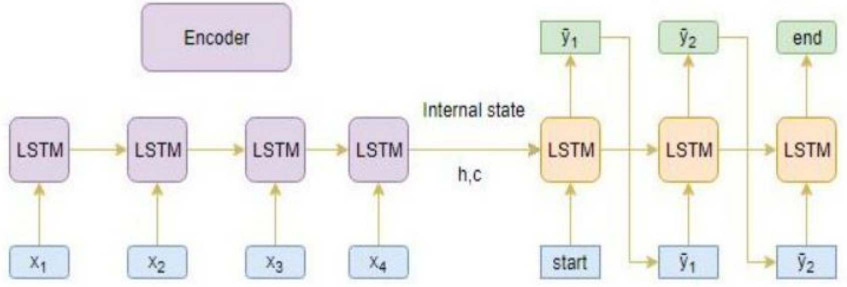
In the preprocessing step, some linguistic technologies are mainly used to structure the input text, such as sentence segmentation, word tokenization, and stop-word removal, etc. In the semantic understanding step, a neural network is constructed to recognize and represent the deep semantics of the input text.is step occurs in the vector space, and finally generates a fusion vector for the next step. In the summary generation step, the generator makes appropriate adjustments to the fusion vector provided in the previous step, and then maps the vector space representation to the vocabulary to generate summary words.

Step by Step Architecture of Model:

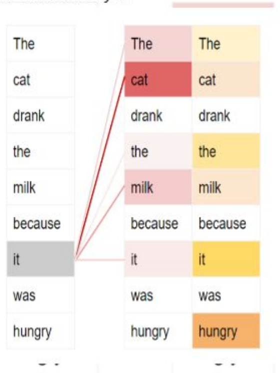
- **Preprocessing**: The input text is preprocessed by tokenizing it into words or subwords and converting it into a sequence of numerical vectors. The vocabulary used for tokenization is typically learned during pre-training.
- **Encoder**: The preprocessed input text is then passed through an encoder component of the transformer model, which consists of multiple self-attention layers. Each self-attention layer attends to all the previous layers to generate a contextualized representation of each input token.
- **Decoder**: The decoder component of the transformer model generates the summary. Like the encoder, the decoder consists of multiple self-attention layers. However, in addition to self-attention, the decoder also attends to the encoder output to incorporate information from the input text.
- **Masking:** During training, some of the input tokens may be masked out, meaning that they are replaced with a special "mask" token. This forces the model to predict the missing tokens based on the surrounding context.
- **Prediction**: The decoder generates a probability distribution over the possible output tokens for each position in the summary. The token with the highest probability is selected as the predicted output token.
- **Beam search**: To generate a full summary, the model typically uses a beam search algorithm to select a sequence of output tokens with the highest overall probability. The beam search algorithm maintains a list of the k most likely sequences of output tokens, where k is a hyperparameter that determines the width of the search beam.
- **Post Processing**: The final summary is then post processed by converting the numerical vectors back into words or subwords, removing any special tokens (such as the mask token), and formatting the summary for readability.



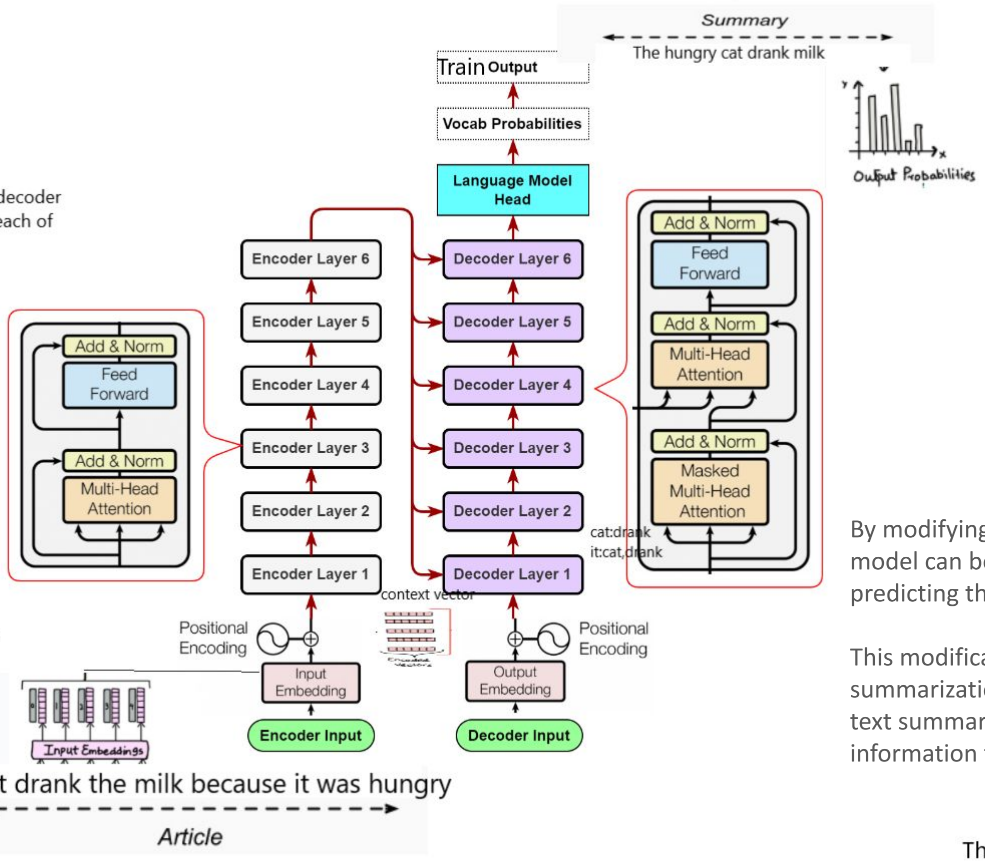
- By modifying the output layer of the transformer, the model can be trained to generate summaries instead of predicting the next word in a sequence.This modification allows the transformer to be used for text summarization, which requires the generation of a shorter text summary that captures the most important information from the input tex

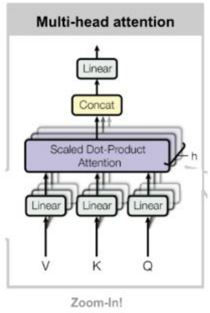


context vector probability so that each decoder layer can extractimportant words from each of the encoder layer.



Attention model finds relation of each words with each of the others words in the sentences.





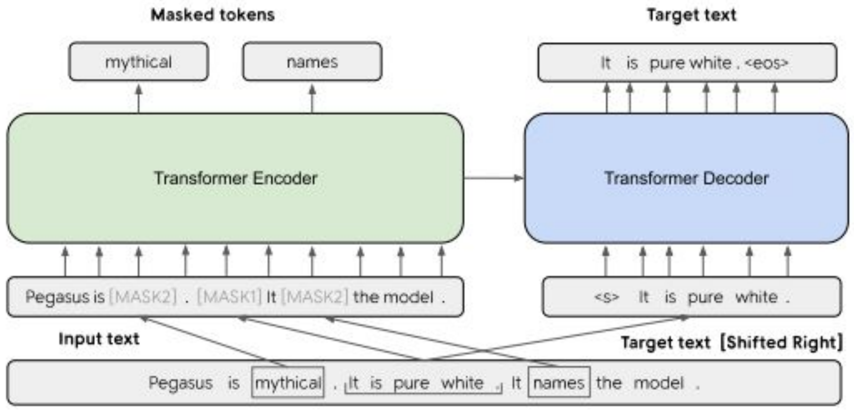
By modifying the output layer of the transformer, the model can be trained to generate summaries instead of predicting the next word in a sequence.

This modification allows the transformer to be used for text summarization, which requires the generation of a shorter text summary that captures the most important information from the input text.

The trained output is matched using ROUGE accuracy metric with test output.

PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization

- PEGASUS is a transformer-based model pre-training with extracted gap sentences for abstractive summarization to generate summaries that preserve the most important information from the input text.The base architecture of PEGASUS is a standard Transformer encoder-decoder. Both GSG and MLM are applied simultaneously.
- They train both input-output pairs and partially masked input-output pairs, which allows it to better handle long sequences and generate more coherent summaries.
- After pre-training, the PEGASUS model can be fine-tuned on a variety of summarization tasks, such as news article summarization, scientific paper summarization, and even when low resource summarization is considered.(dataset C4 same as that of T5).
- **PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization**


Masked tokens
mythical
names
Target text
It is pure white . <eos>
Transformer Encoder
Transformer Decoder
Pegasus is [MASK2] . [MASK1] It [MASK2] the model .
<s> It is pure white .
Input text
Target text [Shifted Right]
Pegasus is mythical . It is pure white . It names the model .

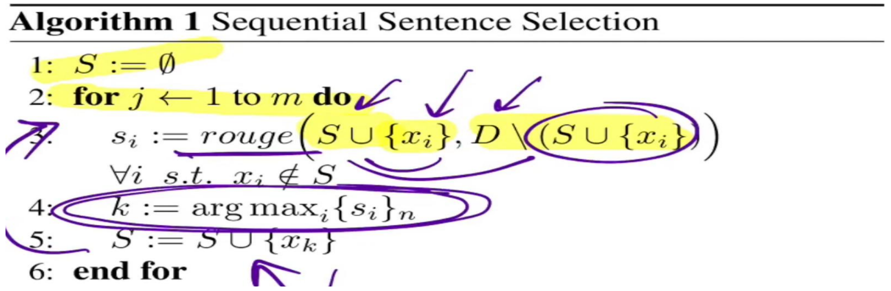
The base architecture of PEGASUS is a standard Transformer encoder-decoder. Both GSG and MLM are applied simultaneously

How output masked helps?

The model to learn a strong understanding of the relationships between the sentences in the document, as well as the key concepts and entities mentioned in the text.By training on a large corpus of documents, the model is exposed to a wide range of writing styles and content domains, which helps it to learn to generalize to new documents and generate high-quality summaries.

How does PEGASUS decide which sentences to mask?

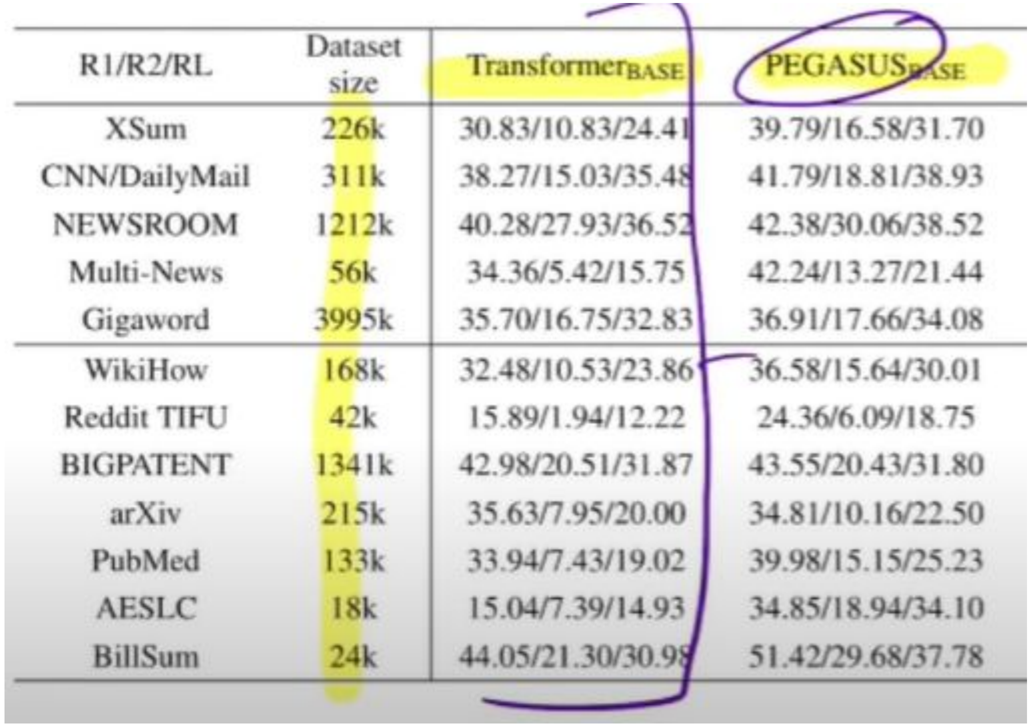
By,extractive summarization,model first generates a set of candidate sentences that are likely to be important for summarizing the document then scored based on their importance, and the top-scoring sentences are selected to be included in the final summary.



Algorithm 1 Sequential Sentence Selection

```
1: $S := \emptyset$
2: for $j \leftarrow 1$ to $m$ do
3:     $s_i := rouge(S \cup \{x_i\}, D \setminus (S \cup \{x_i\}))$
       $\forall i \ s.t. \ x_i \notin S$
4:     $k := \arg\max_i \{s_i\}_n$
5:     $S := S \cup \{x_k\}$
6: end for
```

Following BERT, we select 15% tokens in the input text, and the selected tokens are (1) 80% of time replaced by a mask token [MASK2], or (2) 10% of time replaced by a random token, or (3) 10% of time unchanged.



R1/R2/RL	Dataset size	Transformer$_{BASE}$	PEGASUS$_{BASE}$
XSum	226k	30.83/10.83/24.41	39.79/16.58/31.70
CNN/DailyMail	311k	38.27/15.03/35.48	41.79/18.81/38.93
NEWSROOM	1212k	40.28/27.93/36.52	42.38/30.06/38.52
Multi-News	56k	34.36/5.42/15.75	42.24/13.27/21.44
Gigaword	3995k	35.70/16.75/32.83	36.91/17.66/34.08
WikiHow	168k	32.48/10.53/23.86	36.58/15.64/30.01
Reddit TIFU	42k	15.89/1.94/12.22	24.36/6.09/18.75
BIGPATENT	1341k	42.98/20.51/31.87	43.55/20.43/31.80
arXiv	215k	35.63/7.95/20.00	34.81/10.16/22.50
PubMed	133k	33.94/7.43/19.02	39.98/15.15/25.23
AESLC	18k	15.04/7.39/14.93	34.85/18.94/34.10
BillSum	24k	44.05/21.30/30.98	51.42/29.68/37.78

A comparison of PEGASUSbase and TRANSFORMERbase on XSum, CNN/DailyMail and Gigaword(etc), shows that Pegasus does much better than our original model framework.

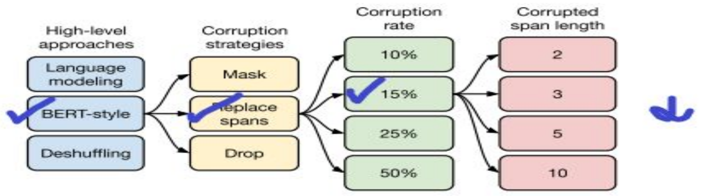
T5 Text-to-Text Transfer Transformer

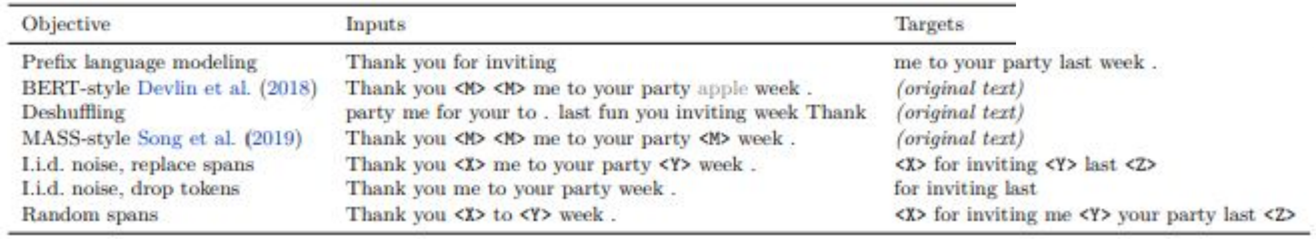
- T5 is a popular language model developed by Google, based on the same transformer architecture as BERT(Bidirectional Encoder Representations from Transformers) and GPT.
- It is a transformer-based model that uses a text-to-text framework for pretraining ie.a variant of the BERT pre-training objective called span corruption, which involves randomly masking contiguous spans of text in the input and training the model to predict the original text.
- The pre-trained model can then be fine-tuned on specific downstream tasks,summarization,here it uses both extractive and abstractive summarization.
- The decoder component fine-tuned using a single approach to GPT, is typically trained to generate the summary one token at a time, using an autoregressive approach. During inference, the decoder generates the summary one token at a time, conditioning each token on the previously generated tokens, until the entire summary has been generated it accurately captures the most important information in the document, while also being grammatically correct and readable.
- The C4 dataset, which is used to pre-train T5, is a large corpus of text that was derived from the Common Crawl web corpus. By filtering, the T5 model is trained on a high-quality corpus of text.
- **Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer**

The model was experimented with different approaches:

- the Language Modeling (predict the afterword in a sentence by considering all preceding words),
- Deshuffling (shuffle all the words in a sentence followed by training the model to predict the original text),
- Denoising objectives (masking a sequence of words from the sentence followed by training the model to predict the masked words).

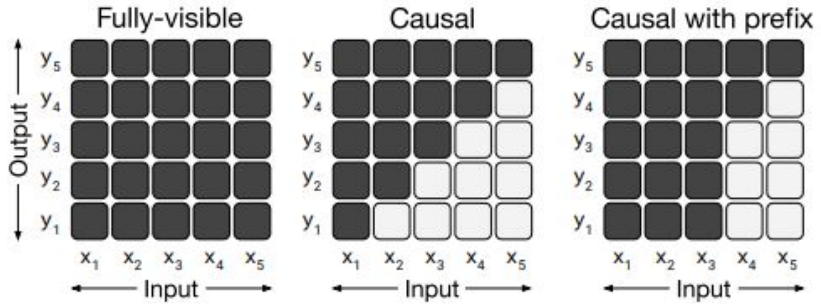
Denoising objectives was selected



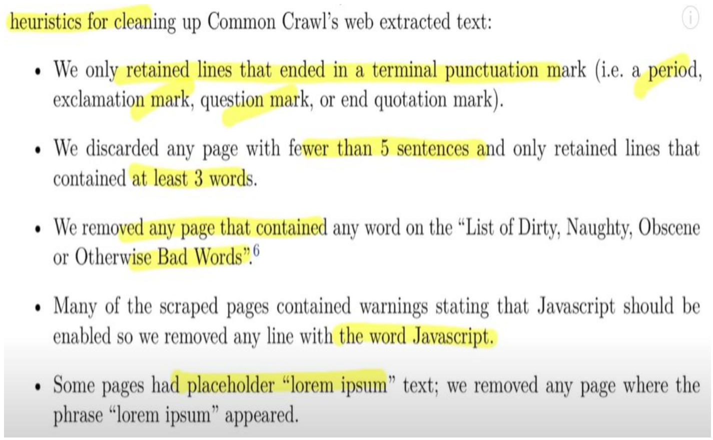


Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

T5 uses a combination of these attention mask patterns, including the causal attention mask to generate the summary one token at a time in a left-to-right fashion, as well as the encoder-decoder attention mask to encode the input text and generate the summary.

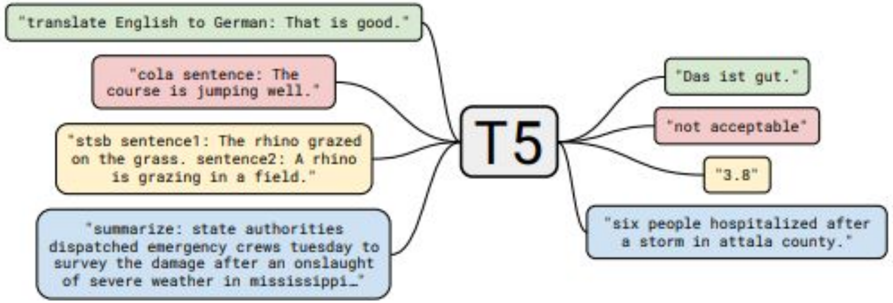


C4 datasets removed:



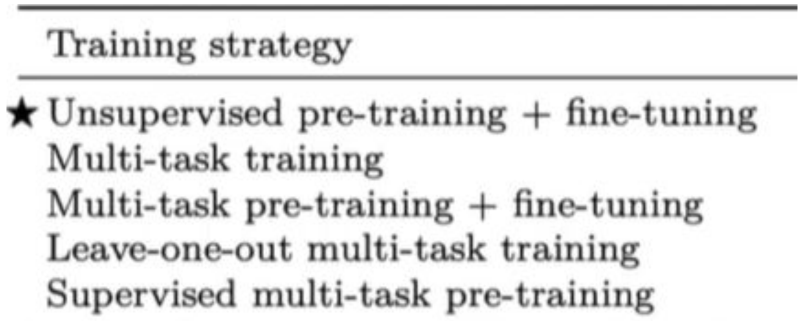
heuristics for cleaning up Common Crawl's web extracted text:

- We only retained lines that ended in a terminal punctuation mark (i.e. a period, exclamation mark, question mark, or end quotation mark).
- We discarded any page with fewer than 5 sentences and only retained lines that contained at least 3 words.
- We removed any page that contained any word on the "List of Dirty, Naughty, Obscene or Otherwise Bad Words".[6]
- Many of the scraped pages contained warnings stating that Javascript should be enabled so we removed any line with the word Javascript.
- Some pages had placeholder "lorem ipsum" text; we removed any page where the phrase "lorem ipsum" appeared.


"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

https://youtube.com/shorts/HrGSh_XcsaE?feature=share

Task prefix:summarization you add the keyword summarize then colon and then you put in the text that you want to summarize on the other end you get it summarization so this way they have tamed the model in terms of picking the hint through the prefix of the input that they give to help the model be in that specific zone of weights that would.



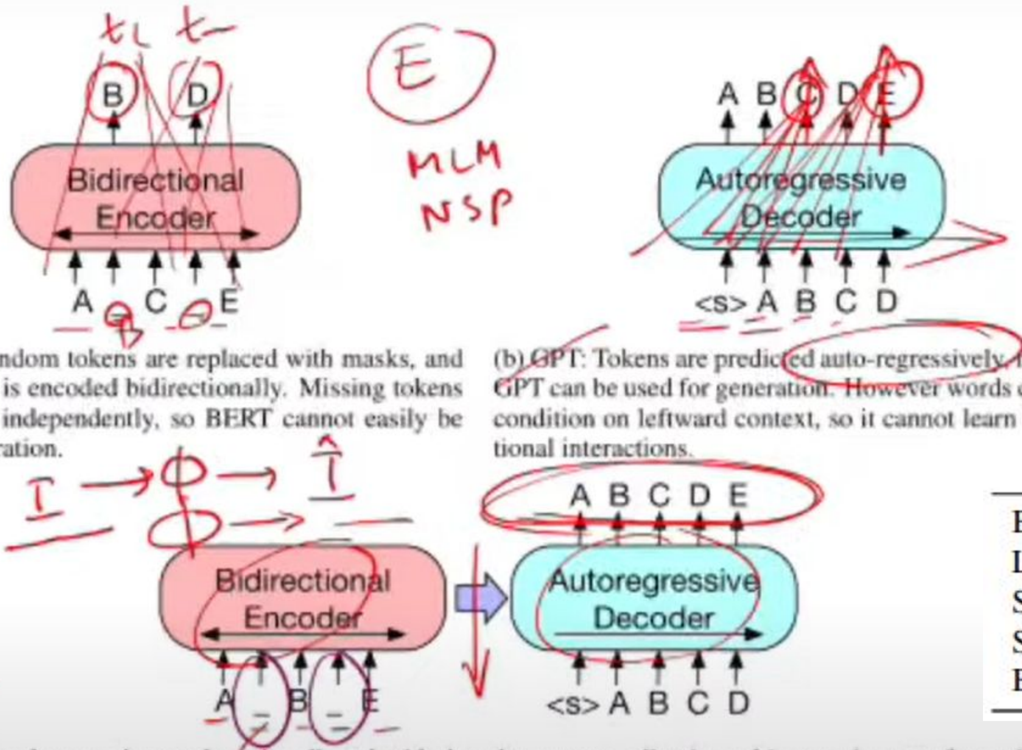
Training strategy
★ Unsupervised pre-training + fine-tuning
Multi-task training
Multi-task pre-training + fine-tuning
Leave-one-out multi-task training
Supervised multi-task pre-training

The choice of which training strategy to use depends on various factors, including the size of the dataset, the quality of the input data, and the available computational resources. Talking specifically about text summarization

However, generally, unsupervised pre-training + fine-tuning is one of the most widely used. This is because unsupervised pre-training allows the model to learn general language representations from large amounts of unlabeled text data, which can be fine-tuned on a smaller labeled dataset for the specific task of text summarization.(task prefix)

BART*Bidirectional AutoRegressive Transformers*

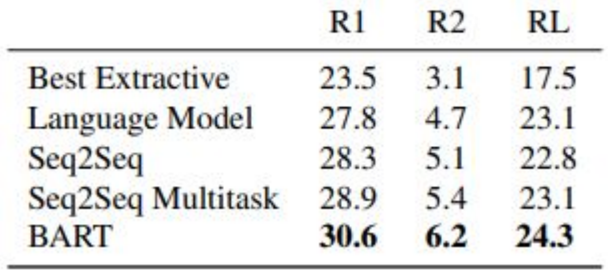
- BART is a sequence-to-sequence model that is based on the transformer architecture and is trained on a combination of denoising autoencoder and sequence-to-sequence objectives.
- It uses a standard Tranformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder),
- It uses a pre-training objective that involves corrupting the input by randomly permuting the words and training the model to reconstruct the original sequence.
- BART is a denoising autoencoder that maps a corrupted document to the original document it was derived from. It is implemented as a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder.
- By incorporating both noising and denoising into the training process, BART can learn to generate high-quality summaries that are more robust to variations in the input text. This can be especially helpful for text summarization tasks, where the input text can vary in length, structure, and content.
- https://arxiv.org/pdf/1910.13461.pdf



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.



	R1	R2	RL
Best Extractive	23.5	3.1	17.5
Language Model	27.8	4.7	23.1
Seq2Seq	28.3	5.1	22.8
Seq2Seq Multitask	28.9	5.4	23.1
BART	**30.6**	**6.2**	**24.3**

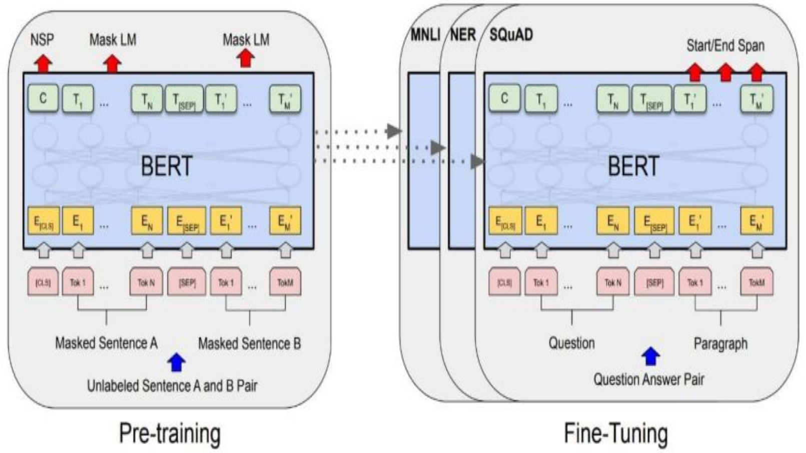
BART is a denoising autoencoder that maps a corrupted document to the original document it was derived from. It is implemented as a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder

Pretraining BART:Unlike existing denoising autoencoders, which are tailored to specific noising schemes, BART allows us to apply any type of document corruption.

- Token Masking Following BERT,random tokens are sampled and replaced with [MASK] elements.
- Token Deletion Random tokens are deleted from the input, it must decide which positions are missing inputs.
- Text Infilling A number of text spans are sampled, with span lengths drawn from a Poisson distribution (λ = 3). Each span is replaced with a single [MASK] token.Text infilling teaches the model to predict how many tokens are missing from a span.
- Sentence Permutation A document is divided into sentences based on full stops,are shuffled in a random order.
- Document Rotation A token is chosen uniformly at random, and the document is rotated to identify the start of the document.

This masked inputs goes to my model at the encoder and at the decoder end it is trained against the actual input

Fine Tuning:For sequence classification tasks, the same input is fed into the encoder and decoder, and the final hidden state of the final decoder token is fed into new multi-class linear classifier.
Sequence Generation Tasks Because BART has an autoregressive decoder, it can be directly fine tuned for sequence generation tasks such as abstractive question answering and summarization. In both of these tasks, information is copied from the input but manipulated, which is closely related to the denoising pre-training objective. Here, the encoder input is the input sequence, and the decoder generates outputs autoregressive.



Methods for pre-training and fine-tuning BERT in general. Both pre-training and fine-tuning employ the identical architectures, with the exception of output layers. For numerous downstream operations, the same pre-trained model parameters are utilised to initialise models. All settings are fine-tuned during fine-tuning.A special symbol [CLS] appears before each input example, and a special separator token [SEP] appears after each input example (for example, separating questions and answers).But just assume inputs and outputs to be the long text and summary respectively.

GPT-3

- GPT-3 is a large-scale transformer-based language model that has been trained on a massive amount of text data using a diverse range of pre-training objectives.
- It has shown impressive results on various NLP tasks, including abstractive summarization.
- GPT-3 uses a variant of the GPT transformer architecture, which is based on the transformer decoder architecture and uses a left-to-right decoding strategy.
- It also uses a novel scaling technique that enables it to handle longer sequences of text than previous transformer models.
- https://www.width.ai/post/gpt3-summarizer

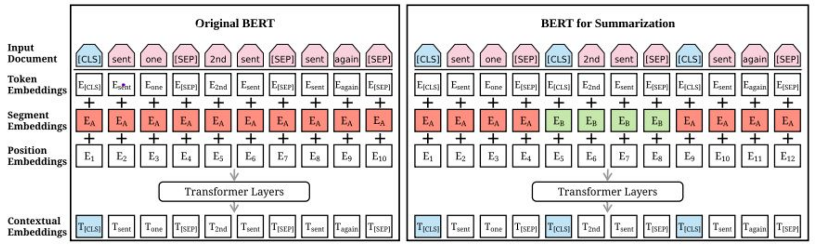
ProphetNet

- ProphetNet is a transformer-based model that uses a variant of the self-attention mechanism and introduces a new pre-training objective called masked sequence to sequence pre-training.
- This pre-training objective involves randomly masking both the input and output sequences and training the model to predict the masked tokens.
- The ProphetNet model uses a multi-layer bidirectional transformer encoder and a unidirectional transformer decoder. The encoder processes the input sequence in both forward and backward directions, allowing it to capture contextual information from both past and future tokens.
- ProphetNet is designed to generate long sequences of text and has shown promising results in abstractive summarization tasks. ProphetNet's pretraining model is designed to capture long-range dependencies in the input sequence and generate high-quality output sequences

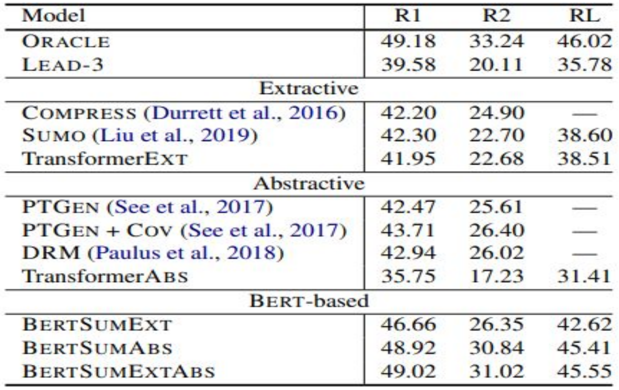
Papers with modifications

Text Summarization with Pretrained Encoders BERTSum(https://arxiv.org/pdf/1908.08345.pdf):

- The authors propose a new summarization model called BERTSum, which consists of two components: a transformer-based encoder that is pretrained on a large corpus of text using the BERT model, and a decoder that generates the summary based on the encoded representation of the input document.
- Our extractive model is built on top of this encoder by stacking several inter sentence Transformer layers. For abstractive summarization, we propose a new fine-tuning schedule which adopts different optimizers for the encoder and the decoder as a means of alleviating the mismatch between the two.
- The use of BPE(byte-pair encoding) and subword tokenization in BERTSum allows the model to handle OOV words,they handle by representing them as a combination of subword units that are present in the vocabulary.



✓ Figure 1: Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences.



Model	R1	R2	RL
ORACLE	49.18	33.24	46.02
LEAD-3	39.58	20.11	35.78
Extractive			
COMPRESS (Durrett et al., 2016)	42.20	24.90	—
SUMO (Liu et al., 2019)	42.30	22.70	38.60
TransformerEXT	41.95	22.68	38.51
Abstractive			
PTGEN (See et al., 2017)	42.47	25.61	—
PTGEN + COV (See et al., 2017)	43.71	26.40	—
DRM (Paulus et al., 2018)	42.94	26.02	—
TransformerABS	35.75	17.23	31.41
BERT-based			
BERTSUMEXT	46.66	26.35	42.62
BERTSUMABS	48.92	30.84	45.41
BERTSUMEXTABS	49.02	31.02	45.55

Papers with modifications

"ExSum: Enhancing Extractive Summarization with Pre-trained Sequence-to-Sequence Models"https://arxiv.org/pdf/2111.10952.pdf :

- This paper proposes a modified version of the T5 model called "ExT5: Towards Extreme Multi-Task Scaling for Transfer Learning" for enhancing extractive summarization. The model is pre-trained on a large corpus of text and fine-tuned on summarization tasks using a novel training objective that encourages the model to generate more informative summaries.
- ExT5 uses an additional input embedding called "summary embedding" that is concatenated to the input sequence, allowing the model to better capture the relationship between the input document and summary. It also includes an additional output head for predicting the importance scores of each sentence in the input document.
- This proposed approach outperforms several strong baselines on the benchmark datasets, achieving state-of-the-art performance on the CNN/Daily Mail and XSum datasets.

"Better Document-Level Generation with Explicit Span Masking and Insertion"https://arxiv.org/abs/1907.10529 :

- This paper proposes a modified version of the BERT model called "SpanBERT" that incorporates explicit span masking and insertion operations to improve the model's performance on document-level generation tasks, including summarization. The authors show that SpanBERT outperforms standard BERT models on several benchmark datasets.
- SpanBERT is trained to predict spans of contiguous tokens in text, which allows it to better capture the relationships between phrases and sentences in text.

Comparison of Models

PEGASUS is particularly effective at summarizing long documents and scientific papers, while T5 is known for its versatility and has been used for a wide range of NLP tasks. Similarly, GPT-3 is known for its ability to generate fluent and coherent text, making it a good choice for summarizing news articles and other types of content. ProphetNet is designed to generate long sequences of text and has shown promising results in summarization tasks. BART is a powerful sequence-to-sequence model that can handle various NLP tasks and has achieved state-of-the-art results in abstractive summarization.https://arxiv.org/ftp/arxiv/papers/2105/2105.00824.pdf

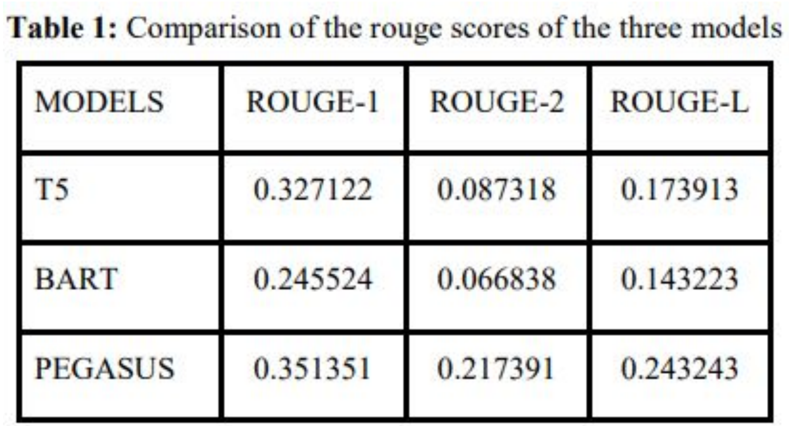
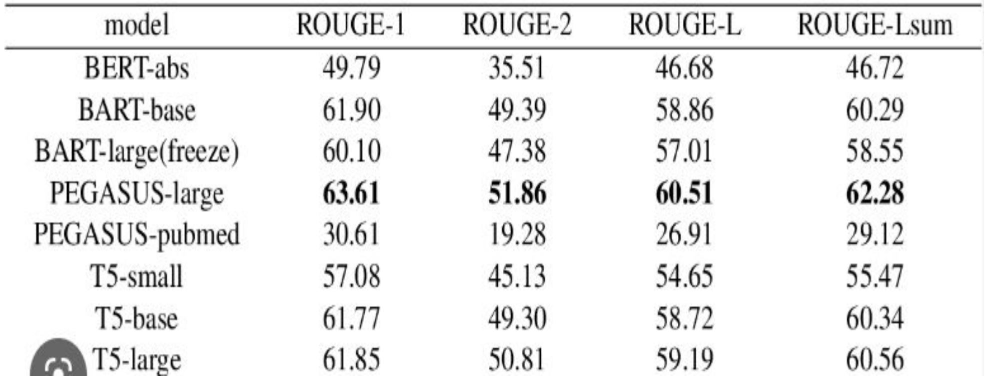


Table 1: Comparison of the rouge scores of the three models

MODELS	ROUGE-1	ROUGE-2	ROUGE-L
T5	0.327122	0.087318	0.173913
BART	0.245524	0.066838	0.143223
PEGASUS	0.351351	0.217391	0.243243



model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
BERT-abs	49.79	35.51	46.68	46.72
BART-base	61.90	49.39	58.86	60.29
BART-large(freeze)	60.10	47.38	57.01	58.55
PEGASUS-large	**63.61**	**51.86**	**60.51**	**62.28**
PEGASUS-pubmed	30.61	19.28	26.91	29.12
T5-small	57.08	45.13	54.65	55.47
T5-base	61.77	49.30	58.72	60.34
T5-large	61.85	50.81	59.19	60.56

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics commonly used in text summarization. It measures the overlap between the words or n-grams (contiguous sequences of n words) in the generated summary and the reference summary.

- The metrics are named based on the size of the n-grams used for comparison. For example: ROUGE-1: measures the overlap of unigrams (single words) between the generated summary and reference summary.
- ROUGE-2: measures the overlap of bigrams (sequences of two adjacent words) between the generated summary and reference summa
- ROUGE-3: measures the overlap of trigrams (sequences of three adjacent words) between the generated summary and reference summary.

Website related

News Article-

Blog-

Video-