**PEGASUS**

PEGASUS stands for Pre-training with Extracted Gap-sentences for Abstractive Summarization.

**Abstract from the research paper:**

Recent work pre-training Transformers with self-supervised objectives on large text corpora has shown great success when fine-tuned on downstream NLP tasks including text summarization.

However, pre-training objectives tailored for abstractive text summarization have not been explored. Furthermore, there is a lack of systematic evaluation across diverse domains.

In this work, the researchers proposed pre-training large Transformer-based encoder-decoder models on massive text corpora with a new self- supervised objective.

In PEGASUS, important sentences are removed/masked from an input document and are generated together as one output sequence from the remaining sentences, similar to an extractive summary.

Researchers has evaluated their best PEGASUS model on 12 downstream summarization tasks spanning news, science, stories, instructions, emails, patents, and legislative bills.

Experiments demonstrate it achieves state-of-the-art performance on all 12 downstream datasets measured by ROUGE scores.

Their model also shows surprising performance on low-resource summarization, surpassing previous state-of-the-art results on 6 datasets with only 1000 examples.

Finally, they validated the results using human evaluation and show that our model summaries achieve human performance on multiple datasets.

PEGASUS is a model for abstractive summarization, a type of summarization that creates summaries that are not just extractive (i.e. copying sentences from the original text) but that rephrase and condense the information using their own words. PEGASUS uses a special technique called gap-sentence generation (GSG) to pre-train a Transformer-based model.

**Here's how it works:**

* Important sentences are identified and removed from a document.
* The remaining sentences are used as input for the model.
* The model is tasked with generating the missing sentences, essentially creating an abstractive summary of the remaining text.

This training method helps the model learn how to identify key information and express it concisely, which is crucial for abstractive summarization.

Overall, PEGASUS is a significant advancement in the field of abstractive summarization, offering a powerful and versatile tool for automatically generating summaries of text documents.

**More on Pegasus pre-training:**

The PEGASUS pre-training process is centered around a clever technique called gap-sentence generation (GSG).

**Here's a deeper dive into how it works:**

* **Data Preparation:** A massive text corpus is used, like news articles or web documents.
* **Sentence Extraction:** Within this corpus, important sentences are identified. This can be done through various methods, like extractive summarization techniques or selecting sentences based on their position in the document (e.g., headlines or opening paragraphs).
* **Masking and Input Creation:** The identified important sentences are removed from the documents. The remaining text becomes the input for the model. Think of it like having a document with some key parts missing.
* **Gap-Sentence Generation (GSG) as the Objective:** The model's goal (objective) during training is to generate the missing sentences (the ones that were removed) based on the remaining context. This essentially forces the model to learn how to create summaries of the unseen parts based on the available information. It's similar to an extractive summary, but the model isn't simply copying sentences - it's learning to rewrite and condense the information using its own words.
* **Training:** The model is trained on a massive dataset using this GSG objective. It continuously refines its ability to predict the missing important sentences based on the context, essentially getting better at summarizing unseen text.

**Additional Points:**

* **Masking with [MASK1]:** The paper mentions using a special token, [MASK1], to indicate where the missing sentence(s) should be generated by the decoder part of the Transformer model.
* **Combination with MLM:** The original PEGASUS paper also explores combining GSG with another pre-training objective called Masked Language Modeling (MLM), which is a common technique used in other Transformer-based models.

**Benefits of GSG Pre-training:**

* **Focus on Summarization:** GSG specifically tailors the pre-training process towards the task of summarization, giving PEGASUS a head start compared to models trained only on generic objectives.
* **Strong Summarization Performance:** This focus on summarization during pre-training is a key reason why PEGASUS achieves state-of-the-art performance on various summarization tasks.

**Key characteristics of PEGASUS's transformer architecture:**

* **Number of Layers:** PEGASUS uses a deep architecture with 16 encoder and decoder layers, allowing for complex relationships to be captured within the text.
* **Positional Embeddings:** Unlike some transformers that use learned positional embeddings, PEGASUS utilizes static, sinusoidal positional embeddings to represent the order of words and sentences.
* **No LayerNorm after Embeddings:** Another deviation from some transformer architectures is the absence of layer normalization after the embeddings in PEGASUS.

The research paper doesn't explicitly mention a chronological order for the datasets used to train PEGASUS. It likely used a very large dataset and the order in which the data was fed into the model wouldn't be crucial.

However, the paper does mention that PEGASUS was pre-trained on a massive text corpus. This likely refers to a collection of multiple text datasets. While the specific datasets aren't named, the paper mentions the following categories the data came from:

News articles

Scientific papers

Short stories

Instructions (e.g., how-to guides)

Emails

Patents

Legislative bills

**PEGASUS Usage Note:**

There isn't direct support for loading Pegasus models in TensorFlow as of today. The original model was built using PyTorch.

However, there are a couple of ways you can potentially use Pegasus with TensorFlow:

* **Community Projects:** There are repositories on Github that offer Pegasus models converted to TensorFlow format. You can search for "PEGASUS tensorflow" to find these repositories (be sure to check the reputation and documentation before using any code).
* **ONNX Conversion:** You can explore converting a PyTorch Pegasus model to ONNX format first. ONNX is an open format for exchanging neural network models, and TensorFlow can run models in this format. Tools from the Hugging Face library can help with ONNX conversion [ONNX conversion with Hugging Face].

It's important to note that converting models might affect performance or accuracy, so it's advisable to test and compare the results carefully.