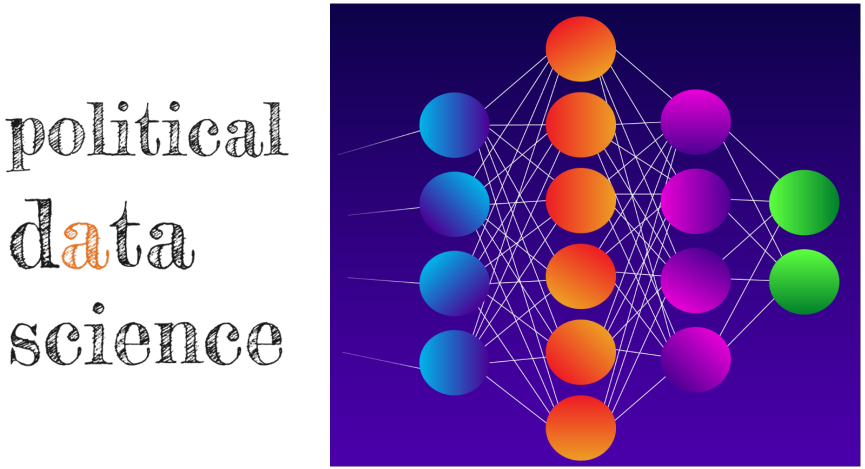
# Applied Deep Learning for NLP

## Week 11 - Text Summarization

Juan Carlos Medina Serrano

Technische Universität München
Hochschule für Politik
Political Data Science

Munich, 14. January 2021



## Conditional Language Model

Task of predicting the next word, given the words so far, and **some other input x**

$$P(y_t|y_1, ..., y_{t-1}, x)$$

For Machine translation: x=source sentence, y=target sentence

## Summarization

Task: Given input x, write summary y which is shorter and contains the main information of x

Summarization can be:

- Single-document: summary from a single x document
- Multi-document: summary from several documents $x_1, ..., x_n$. (With overlapping content)

## Different Datasets Different Tasks

Summarization is a broad task. Existing datasets differ in the input and target:

- Gigaword: x=first one or two sentences, y=headline
- NYT, CNN/DailyMail: x=news article y=multisentence summary
- Sample Wikipedia: x=Wikipedia sentence y= simple sentence (**Sentence simplification**)

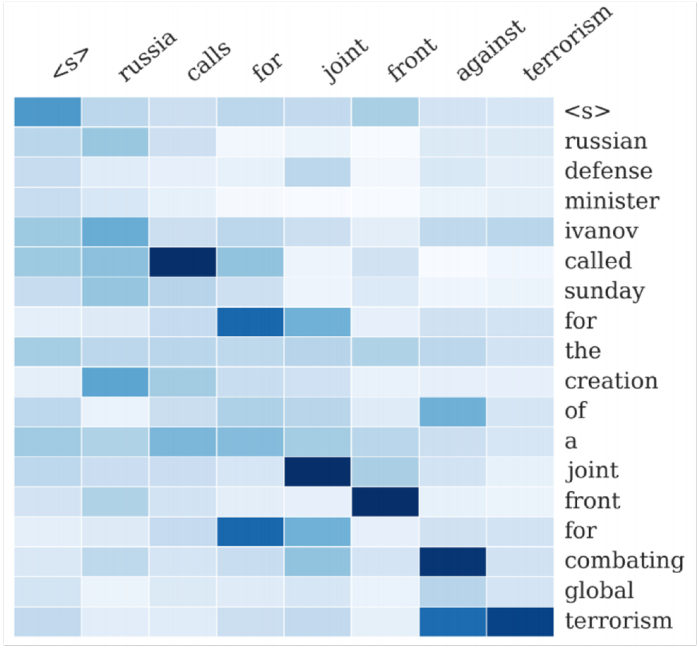
## Summarization Strategies

- **Extractive** Summarization: Select parts (sentences) of the original text to form a summary. Similar task to keyword extraction, but the inputs are sentences and not ngrams.

- **Abstractive** Summarization: Generate new fluent text that summarizes the text.

## Seq2Seq

Basic Seq2Seq + Attention architectures (pre-transformers) work great for sentence simplification:

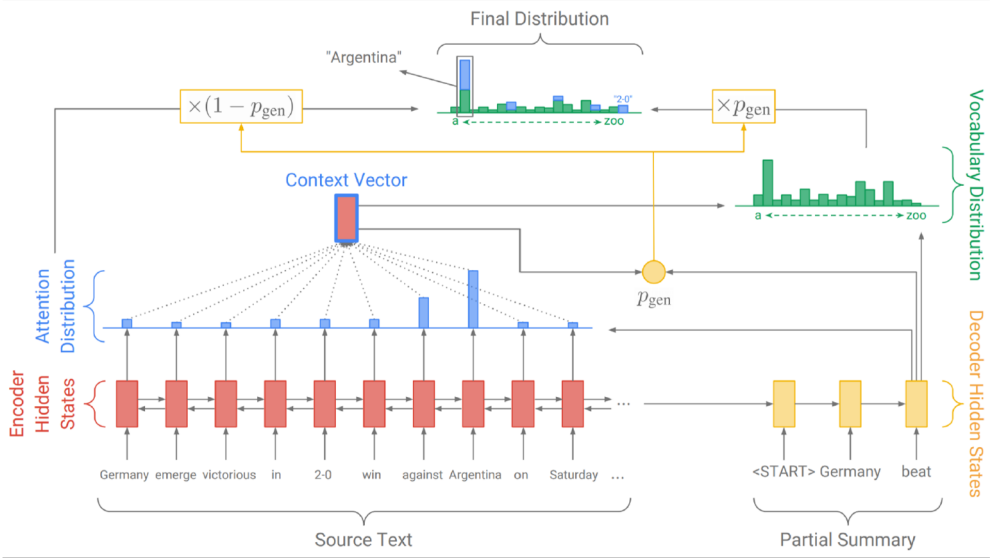



Not so good for long paragraphs. How to make it better?

## Copy Mechanisms

Abstractive summarization can get better using the ideas of extractive summarization: Add a mechanism to copy important parts of the original sentence. Copying rarely occurs in seq2seq architectures.

Idea: Use a probability of generating the next word, instead of copying it



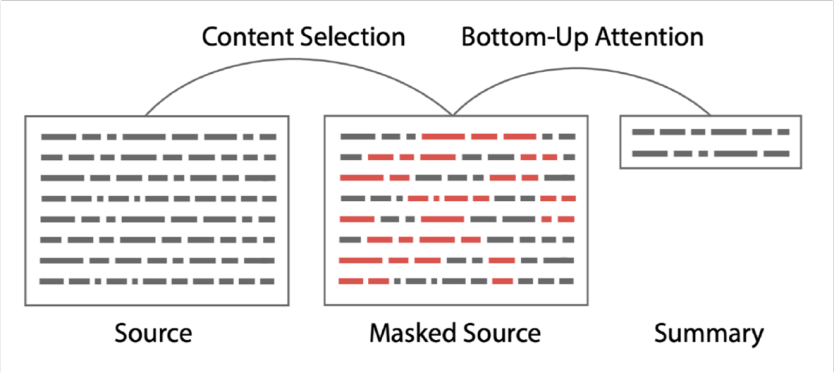

https://arxiv.org/pdf/1704.04368.pdf

## Content Selection

Problem: On each step of the decoder, there is a word-level attention to select next word. However, this does not include any **global** strategy.

Step 1: Use a neural sequence-tagging model to tag words as include or don't-include

Step 2: The attention in the Seq2seq can't attend to words tagged as don't include.



## ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the equivalent of BLUE for summarization.

Also based on **overlapping n-grams**. BUT, no brevity penalty. Normally with a maximum length constraint.

BLUE is based on *precision*.
ROUGE is based on *recall*.

BLUE is reported as a single number that combines 1,2,3,4 n-grams
ROUGE is reportedly separately: ROUGE-1, ROUGE-2, ROUGE-L (longest common subsequence).

## PEGASUS

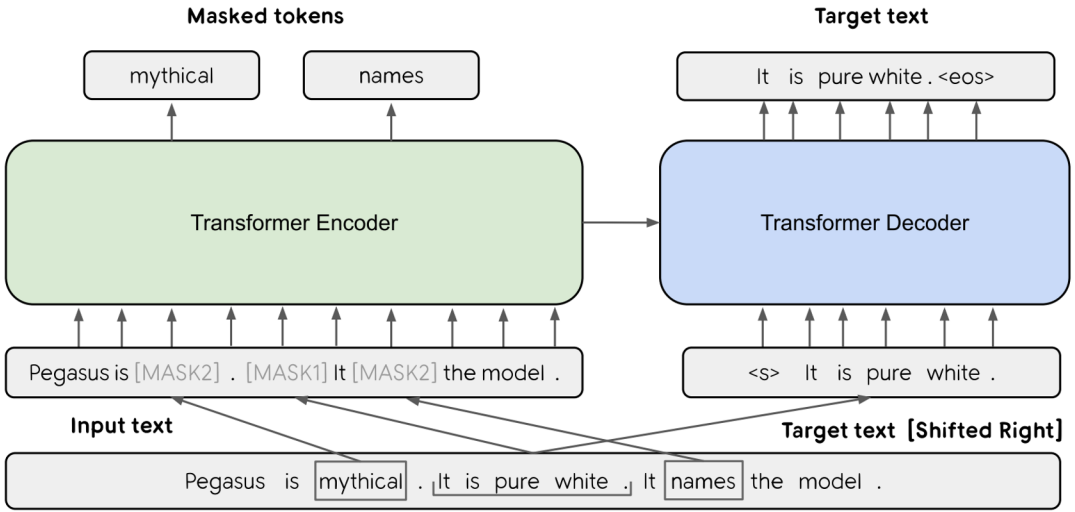
Transformer Encoder- Decoder architecture. It uses the MLM task (Encoder side) and adds a new objective (Decoder side):

**Gap Sentences Generation:** Mask whole sentences from documents, and concatenate the gap-sentences as a pseudo-summary.
Similar to T5, it reconstructs only the masked sentences.

## PEGASUS

Transformer Encoder- Decoder architecture. It uses the MLM task (Encoder side) and adds a new objective (Decoder side):

**Gap Sentences Generation:** Mask whole sentences from documents, and concatenate the gap-sentences as a pseudo-summary.
Similar to T5, it reconstructs only the masked sentences.



## PEGASUS

Masks 15% of the sentences. Selecting the right sentences to mask is important! How:

- Random m sentences
- Lead: First m sentences
- Principal: Select top-m scored sentences according to importance. Use ROUGE1 between sentence and the rest of the document.

Beats all previous SOTA. (Except for the Gigaword dataset)