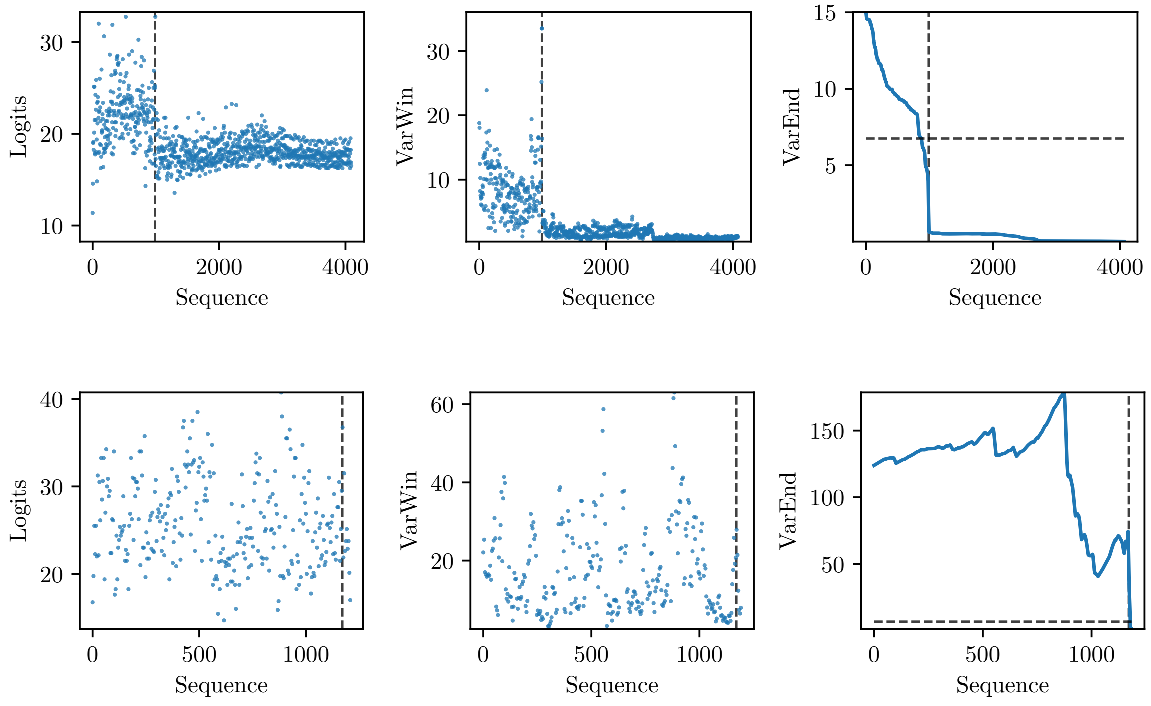

Figure 6: Examples for repetition detection on logits. Top: Sample with repetition, Bottom: Sample without repetition. Left: Highest logit score for each token in the sequence $\ell(x)$, Center: Sliding window variance of the logits $\mathrm{VarWin}_B[\ell](x)$, Right: Variance of variance from the position to the end $\mathrm{VarEnd}_B[\ell](x)$

5.4 Repetitions during inference

We notice that the model degenerates into repeating the same sentence over and over again. The model can not recover from this state by itself. In its simplest form, the last sentence or paragraph is repeated over and over again. We observed this behavior in 1.5% of pages in the test set, but the frequency increases for out-of-domain documents. Getting stuck in a repetitive loop is a known problem with Transformer-based models, when sampled with greedy decoding [44].
It can also happen that the model alternates between two sentences but sometimes changes some words, so a strict repetition detection will not suffice. Even harder to detect are predictions where the model counts its own repetitions, which sometimes happens in the references section.
In general we notice this kind behavior after a mistake by the model. The model is not able to recover from the collapse.

Anti-repetition augmentation Because of that we introduce a random perturbation during training. This helps the model to learn how to handle a wrongly predicted token. For each training example, there is a fixed probability that a random token will be replaced by any other randomly chosen token. This process continues until the newly sampled number is greater than a specified threshold (in this case, 10%). We did not observe a decrease in performance with this approach, but we did notice a significant reduction in repetitions. Particularly for out-of-domain documents, where we saw a 32% decline in failed page conversions.

Repetition detection Since we are generating a maximum of 4096 tokens the model will stop at some point, however it is very inefficient and resource intensive to wait for a “end of sentence” token, when none will come. To detect the repetition during inference time we look at the largest logit value $\ell_i = \max \boldsymbol{\ell}_i$ of the ith token. We found that the logits after a collapse can be separated using the following heuristic. First calculate the variance of the logits for a sliding window of size $B = 15$

$$\mathrm{VarWin}_B[\boldsymbol{\ell}](x) = \frac{1}{B}\sum_{i=x}^{x+B}\left(\ell_i - \frac{1}{B}\sum_{j=x}^{x+B}\ell_j\right)^2.$$

Here ℓ is the signal of logits and x the index. Using this new signal we compute variances again but this time from the point x to the end of the sequence

$$\mathrm{VarEnd}_B[\boldsymbol{\ell}](x) = \frac{1}{S-x}\sum_{i=x}^{S}\left(\mathrm{VarWin}_B[\boldsymbol{\ell}](i) - \frac{1}{S-x}\sum_{j=x}^{S}\mathrm{VarWin}_B[\boldsymbol{\ell}](i)\right)^2.$$

If this signal drops below a certain threshold (we choose 6.75) and stays below for the remainder of the sequence, we classify the sequence to have repetitions.

During inference time, it is obviously not possible to compute the to the end of the sequence if our goal is to stop generation at an earlier point in time. So here we work with a subset of the last 200 tokens and a half the threshold. After the generation is finished, the procedure as described above is repeated for the full sequence.

5.5 Limitations & Future work

Utility The utility of the model is limited by a number of factors. First, the problem with repetitions outlined in section 5.4. The model is trained on research papers, which means it works particularly well on documents with a similar structure. However, it can still accurately convert other types of documents.
Nearly every dataset sample is in English. Initial tests on a small sample suggest that the model's performance with other Latin-based languages is satisfactory, although any special characters from these languages will be replaced with the closest equivalent from the Latin alphabet. Non-Latin script languages result in instant repetitions.
Generation Speed On a machine with a NVIDIA A10G graphics card with 24GB VRAM we can process 6 pages in parallel. The generation speed depends heavily on the amount of text on any given page. With an average number of tokens of $\approx$ 1400 we get an mean generation time of 19.5s per batch for the base model without any inference optimization. Compared to classical approaches (GROBID 10.6 PDF/s [4]) this is very slow, but it is not limited to digital-born PDFs and can correctly parse mathematical expressions.
Future work The model is trained on one page at a time without knowledge about other pages in the document. This results in inconsistencies across the document. Most notably in the bibliography where the model was trained on different styles or section titles where sometimes numbers are skipped or hallucinated. Though handling each page separately significantly improves parallelization and scalability, it may diminish the quality of the merged document text.
The primary challenge to solve is the tendency for the model to collapse into a repeating loop, which is left for future work.

6 Conclusion

In this work, we present Nougat, an end-to-end trainable encoder-decoder transformer based model for converting document pages to markup. We apply recent advances in visual document understanding to a novel OCR task. Distinct from related approaches, our method does not rely on OCR or embedded text representations, instead relying solely on the rasterized document page. Moreover, we have illustrated an automatic and unsupervised dataset generation process that we used to successfully train the model for scientific document to markup conversion. Overall, our approach has shown great potential for not only extracting text from digital-born PDFs but also for converting scanned papers and textbooks. We hope this work can be a starting point for future research in related domains.
All the code for model evaluation, training and dataset generation can be accessed at https://github.com/facebookresearch/nougat.

7 Acknowledgments

Thanks to Ross Taylor, Marcin Kardas, Iliyan Zarov, Kevin Stone, Jian Xiang Kuan, Andrew Poulton and Hugo Touvron for their valuable discussions and feedback.
Thanks to Faisal Azhar for the support throughout the project.

References

[1] Sebastian Spiegler. Statistics of the Common Crawl Corpus 2012, June 2013. URL https://docs.google.com/file/d/1_9698uglerxB9nAglvaHkEgU-iZNm1TvVGuCW7245-WGvZq47teNpb_uL5N9.