The main innovation that has moved methods in natural language processing from largely *not working* to largely *working* is pretraining. At a high level, this idea returns exactly to the start of the course. Pretraining means trying to learn from a huge amount of text (and usually now also images, video, audio, etc. But we'll focus on text.) All of the topics we've covered so far—expressivity of neural networks, optimization, tokenization, and parallelizable architectures—are to some extent in service of better pretraining.

The method we introduced in lecture 1 is similar to word2vec [Mikolov et al., 2013], and we saw that our simple word prediction algorithm led to very interesting learned structure. By scaling up the expressivity of the architecture and the scale, interesting things keep happening!

Recall language modeling

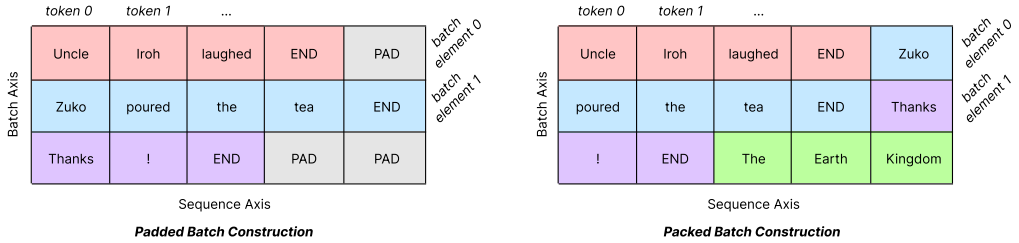

Figure 1: Comparison of padding and packing for batch construction.

What predictive problem are we going to be training our network on? You guessed it, language modeling. Recall that we have some learnable parameters θ in our distribution p_θ, we have a data distribution $\mathcal{D}$, and we're going to optimize:

$$\min_\theta \mathbb{E}_{x\sim\mathcal{D}} \left[-\log p_\theta(x)\right] \tag{1}$$

But at this point let's get a bit more specific. This math suggests that we're optimizing specifically over entire documents. In practice, something slightly messier happens.

Let's say I have a batch size B and a maximum sequence length n. I've probably set B and n such that I have as long sequences as I can, and as large a batch as I can, such that it'll fit on my GPU cluster. So, that's Bn tokens I can potentially learn from. However, if I try to optimize the math above, suggesting that I optimize for the likelihood of whole documents, this means I need to filter out documents longer than n tokens. Furthermore, any documents shorter than n tokens I need to pad with useless blanks that won't be trained on, in order to fill out the batch. That's wasted compute!

Instead, batches are *packed* with tokens. That is, we string together a bunch of documents and just pick the first Bn tokens, even if they cross document boundaries. If they do cross document boundaries, we include a document separator token.

We then optimize a similar-looking objective over this new distribution over tokens, call it $\tilde{\mathcal{D}}$.

$$\min_\theta \mathbb{E}_{x_{1:t}\sim\tilde{\mathcal{D}}} \left[-\log p_\theta(x_t \mid x_{<t})\right] \tag{2}$$

This is a bit of a technical detail—much of the time we're still optimizing document likelihoods, but not always due to the packing (Figure 1), but I think it helps build intuition—we're looking for useful token sequences to learn from, and a lot, as fast as possible.

Pretraining Data Size and GPT Models

First, consider the size of datasets that are used in practice for pretraining models such as the GPT series of models. In June of 2018, the initial idea of a *generative pre-trained transformer* (GPT) was introduced by Alec Radford and others at OpenAI [Radford and Narasimhan, 2018]. The GPT-1 model has 117 million learnable parameters, and was trained on the BooksCorpus, containing roughly 985 million words or 1.3 billion tokens. With this size and training, the model was already able to do interesting things by primarily learning to predict the next token. Later, in 2019, Radford and others released GPT-2 [Radford et al., 2019] which has 1.5 billion learnable parameters and was pretrained on a large corpus of web text containing 8 million documents and roughly 21 billion tokens. In 2020, OpenAI then announced GPT-3 [Brown et al., 2020] with 175 billion parameters trained on an estimate ∼500 billion tokens of CommonCrawl data. Finally, GPT-4 [OpenAI et al., 2024] and GPT-5 were announced in March, 2023 and August 2025 respectively, but we don't really know details about the size of the model or training corpus.

Some other common, recent models and their associated parameter counts are listed in Table 1. Many current leading models have hundreds of billions of tokens and are trained on tens of trillions of tokens.

Model	Provider	Model Size (# Parameters)	Corpus Size (# Tokens)
OLMo-2	AllenAI	7-13 billion	5 trillion
Gemma 3	Google	270 million - 27 billion	6-12 trillion
DeepSeek-V3	DeepSeek AI	685 billion	14.8 trillion
Llama 3	Meta	7-405 billion	15 trillion
Qwen3	Alibaba	235 billion - 1 trillion	36 trillion

Table 1: Common large language models with their size and number of training tokens. Ranges of model sizes indicate that multiple models of different sizes were part of the same named release (e.g., Gemma-3-8B and Gemma-3-27B).

Data distributions

Take a look at Table 2. Here are two "real" pretraining documents from FineWeb [Penedo et al., 2024].

What token sequences should we learn from? Sometimes we approximate the training distribution of language models as *the whole internet*, but this is wrong for various reasons. Still, it's a useful intuition, and a large part of pretraining datasets is often Common Crawl dumps, which are large public crawls of the internet. But cleaning, filtering, formatting the text in raw crawls is critical.

Here are some types of filtering or processing found in the FineWeb dataset [Penedo et al., 2024]:

- Text extraction from HTML! Actually not easy to get right.
- Language filtering for mostly-English data
- URL filtering to avoid adult content
- Deduplication
- Personally identifiable information heuristic removal

To put some of these pre-processing steps in perspective, consider the texts removed

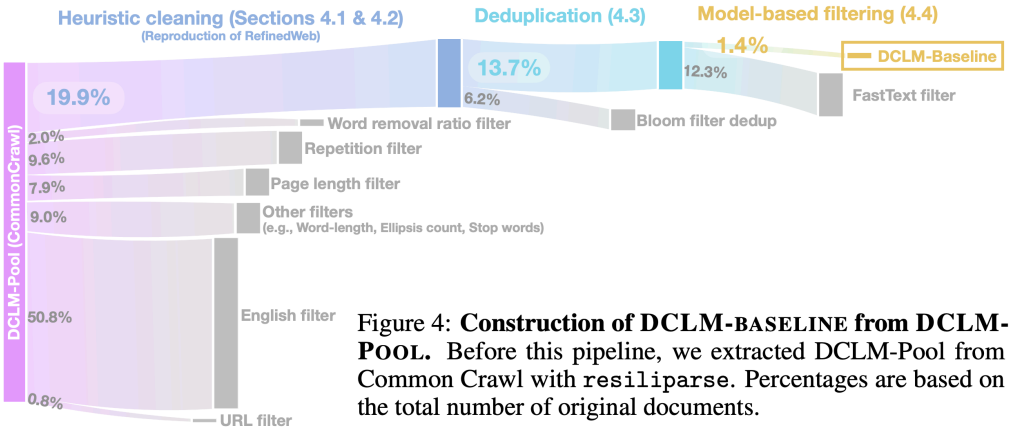

Figure 4: **Construction of DCLM-BASELINE from DCLM-POOL.** Before this pipeline, we extracted DCLM-Pool from Common Crawl with `resiliparse`. Percentages are based on the total number of original documents.

Figure 2: Proportion of texts removed by pretraining data filtering and pre-processing steps from [Li et al., 2024]

by each of several steps in Figure 2. Beginning with a pool of CommonCrawl texts, DataCompLM removes nearly 99% of the original documents to curate a corpus for pretraining, and the final corpus still contains roughly 2-3 trillion tokens. While most documents are removed by filters similar to those listed above (e.g., language filters, deduplication), the final step, *model-based filtering*, primarily concerns selecting the kinds of documents we think would be best for pretraining.

It's **very expensive** to test hypotheses about what kind of web data is best for pretraining. Vaguely, some notion of quality is usually used—Wikipedia is high quality, random web pages might be low-quality. One nice concrete intuition is as follows, however.

In DataCompLM, they trained models to score each web document with how alike it is to a combination of Reddit Explain-Like-I'm-Five data and a synthetically generated question-and-long-response chatbot dataset called OpenHermes2.5 [Teknium, 2023]. That is, when filtering down from a raw web dump, documents are kept when the model predicts that they're *more* like these sources. And this worked very well!

Model-Based Filtering

As mentioned, if Wikipedia is primarily high quality, why can't we just train a model only on Wikipedia texts? All of English Wikipedia contains roughly 5-7 billion tokens [1]. In contrast, however, the original GPT-2 model training used more than 21 billion tokens of text. Therefore, we need a method to estimate the quality of documents outside of known sources like Wikipedia.

While some other approaches exist [2], the approach taken in DataCompLM to score the quality of documents based on known sources is similar to many of the model-based filtering approaches often used to filter pretraining corpora. Assume you have a pool of documents, $\mathcal{D} = \{D_1, D_2, ..., D_N\}$ that you may want to use to pretrain your model where each document is a string over a finite vocabulary, $D_i = w_1, ..., w_T$. Given that quality is difficult to define precisely, it is unlikely we will come up with a good approach

[1] As of October 2025, there are around 5 billion words on English Wikipedia.

[2] For example, one heuristic of quality that was used for curating data for GPT-2 was to collect webpages linked to and highly upvoted in Reddit comments. In this case, the general idea is that posts with many upvotes link to higher quality web pages while those with fewer upvotes likely link to lower quality web pages.

FineWeb Examples
Previous abstract Next abstract Session 40 - The Interstellar Medium. Display session, Tuesday, June 09 Gamma Ray Burst (GRB) explosions can make kpc-size shells and holes in the interstellar media (ISM) of spiral galaxies if much of the energy heats the local gas to above 10^7 K. Disk blowout is probably the major cause for energy loss in this case, but the momentum acquired during the pressurized expansion phase can be large enough that the bubble still snowplows to a kpc diameter. This differs from the standard model for the origin of such shells by multiple supernovae, which may have problems with radiative cooling, evaporative losses, and disk blow-out. Evidence for giant shells with energies of $\sim 10^53$ ergs are summarized. Some contain no obvious central star clusters and may be GRB remnants, although sufficiently old clusters would be hard to detect. The expected frequency of GRBs in normal galaxies can account for the number of such shells. Program listing for Tuesday
Wikipedia sobre física de partículas Rapidinho. Me falaram que a definição de física de partículas da Wikipedia era muito ruim. E de fato, era assim: Particle physics is a branch of physics that studies the elementary particle|elementary subatomic constituents of matter and radiation, and their interactions. The field is also called high energy physics, because many elementary particles do not occur under ambient conditions on Earth. They can only be created artificially during high energy collisions with other particles in particle accelerators. Particle physics has evolved out of its parent field of nuclear physics and is typically still taught in close association with it. Scientific research in this area has produced a long list of particles. Mas hein? Partículas que só podem ser criadas em aceleradores? Física de partículas é ensinada junto com física nuclear? A pesquisa produz partículas (essa é ótima!)? Em que mundo essa pessoa vive? Reescrevi: Particle Physics is a branch of physics that studies the existence and interactions of particles, which are the constituents of what is usually referred as matter or radiation. In our current understanding, particles are excitations of quantum fields and interact following their dynamics. Most of the interest in this area is in fundamental fields, those that cannot be described as a bound state of other fields. The set of fundamental fields and their dynamics are summarized in a model called the Standard Model and, therefore, Particle Physics is largely the study of the Standard Model particle content and its possible extensions. Eu acho que ficou bem melhor. Vamos ver em quanto tempo algum editor esquentado da Wikipedia vai demorar para reverter. Atualmente está um saco participar da Wikipedia por causa dessas pessoas.

Table 2: Some example documents from FineWeb. Note that they're still pretty messy.

to label each document as definitively "bad" or "good". So, the aim of model-based quality filtering is to instead sort this set of documents by there quality.

Taking sources of documents that we assume are high quality (i.e., Wikipedia), we can train a classifier defined by $f(D) \mapsto c$, where the output, c, is a value $[0, 1]$ represents the probability that D could be from our high quality sources. If Wikipedia is our high quality source for example, we train this classifier to prioritize (i.e., give a high score to) documents that look like or contain Wikipedia text and de-prioritize (i.e. give a low score to) documents that don't look like Wikipedia text. We can then label each document with this classifier, sort our training pool by the associated score, and take the highest scoring documents as our pretraining data.

To train these model-based filters, we can use much of what we have learned in the course so far. Instead of predicting the next token, however, we instead want to perform binary classification that will output a single quality score. If we remove the final linear layer and softmax of a transformer, we instead get a vector corresponding to the last token of the input, $h = \text{Transformer}_\theta(D)$ where $h \in \mathbb{R}^d$ and θ represents all learnable parameters of the transformer. Intuitively, this h is the best representation of the document we can extract from the model given it incorporates some information about all prior tokens in the sequence through all layers of the model.

Using this representation from the model, we define a new model as follows:

$$f(D) = \sigma(h^\intercal g) \tag{3}$$

where g is a small set of learnable parameters in $\mathbb{R}^d$ defining the classifier. Then, we can define a loss function for this classifier and use gradient descent to train all parameters. Formally, we could try to optimize the score given to documents from Wikipedia (or another high quality source):

$$\min_{\theta,g} \mathbb{E}_{D \in \text{Wiki}} \left[-\log(f(D)) \right] \tag{4}$$

This formulation, however, can be trivially optimized by always predicting $f(D) = 1$ because it lacks *negative examples*. So, we modify our training by assuming that generally, any other documents that we sample besides Wikipedia documents are, on average, worse than Wikipedia. While this may result in a poor classifier if we relied on binary labels, remember that we only need to *sort* our pool of training data by their similarity to Wikipedia rather than discretely classify documents as high or low quality. So, we instead optimize:

$$\min_{\theta,g} \mathbb{E}_{D \in \text{Wiki}} \left[-\log(f(D)) \right] - \mathbb{E}_{D \in \mathcal{D}} \left[-\log(f(D)) \right] \tag{5}$$

Scaling laws (estimates)

Part of the promise of pretraining is that it's been shown to work (and thus is to some extent predicted to work) across many orders of magnitude. This is especially important because of the immense costs of scaling to each additional order of magnitude. See Figure 3, in which scaling the amount of computation, the amount of data, and the number of parameters leads to a decay in loss that scales linearly with the logarithm of the increase in cost. While sometimes stated as natural laws, I think of them as useful estimates.

Given this predictable relationship, we can attempt to determine the "optimal" tradeoff between scaling the number of parameters and the number of tokens a model is trained

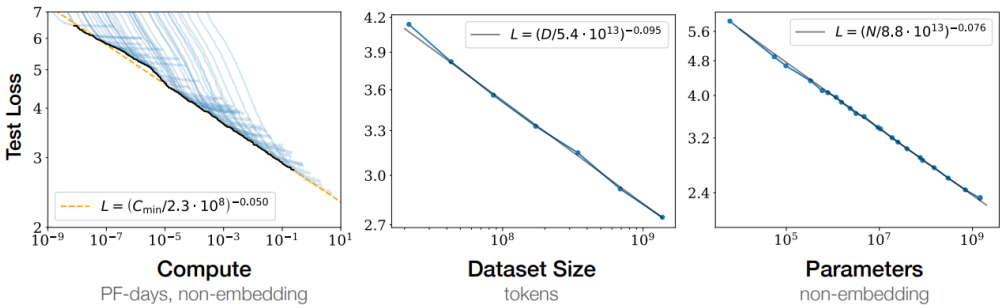

Figure 3: The scaling law plots from Figure 1 of [Kaplan et al., 2020].

on. Specifically, work has investigated how we can empirically determine this tradeoff. To do so, they fit a function that predicts the loss a transformer model will achieve at the end of pretraining [Hoffmann et al., 2022], which is of the form

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} \tag{6}$$

In this function, $\hat{L}(N, D)$ represents the predicted loss (similar to what we have seen earlier in the course) based on the number of parameters (N) and the number of training tokens (D). As there is inherit uncertainty in predicting the next word of real language, the first term, E, represents the minimum achievable or ideal loss of the model. The second term, $\frac{A}{N^{\alpha}}$, represents the added loss due to having too few parameters, while the third term, $\frac{B}{D^{\beta}}$, represents the added loss due to not training on enough data. Among the constants in this function, α and β represent the *rates* of improvement gained when scaling the number of parameters and the number of training tokens respectively. Therefore, knowing these values of these rates provides general guidelines for how to scale the number of parameters and training tokens relative to each other in order to achieve the optimal loss.

After fitting this function to real model training runs and losses, the authors estimated that the relevant scalar values are roughly $\alpha \approx \beta \approx 0.5$. Because α and β are found to be similar, this means that optimally, the number of parameters and the number of training tokens should be scaled similarly to each other.

This optimal tradeoff, however, is generally not followed in practice; while this function gives insights into how to optimally use a computational budget to minimize the resulting training loss, it does not take into account the costs of model inference. Smaller models are less costly to run, so model developers generally choose instead to tradeoff more time spent pretraining and more tokens to keep the costs of model inference to users low.

References

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

[Hoffmann et al., 2022] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. (2022). Training compute-optimal large language models.

[Kaplan et al., 2020] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[Li et al., 2024] Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S. Y., Bansal, H., Guha, E., Keh, S. S., Arora, K., et al. (2024). Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[OpenAI, 2025] OpenAI (2025). Gpt-5 system card.

[OpenAI et al., 2024] OpenAI et al. (2024). Gpt-4 technical report.

[Penedo et al., 2024] Penedo, G., Kydlíček, H., Lozhkov, A., Mitchell, M., Raffel, C. A., Von Werra, L., Wolf, T., et al. (2024). The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

[Radford and Narasimhan, 2018] Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[Teknium, 2023] Teknium (2023). Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants. https://huggingface.co/datasets/teknium/OpenHermes-2.5.