Member-only story

The Rise of Two-Tower Models in Recommender Systems

A deep-dive into the latest technology used to debias ranking models


Samuel Flender · Follow
Published in Towards Data Science
7 min read · Oct 29

Listen Share More

Photo by Evgeny Smirnov

Recommender systems are among the most ubiquitous Machine Learning applications in the world today. However, the underlying ranking models are plagued by numerous biases that can severely limit the quality of the resulting recommendations. The problem of building unbiased rankers — also known as unbiased learning to rank, ULTR — remains one of the most important research problems within ML and is still far from being solved.

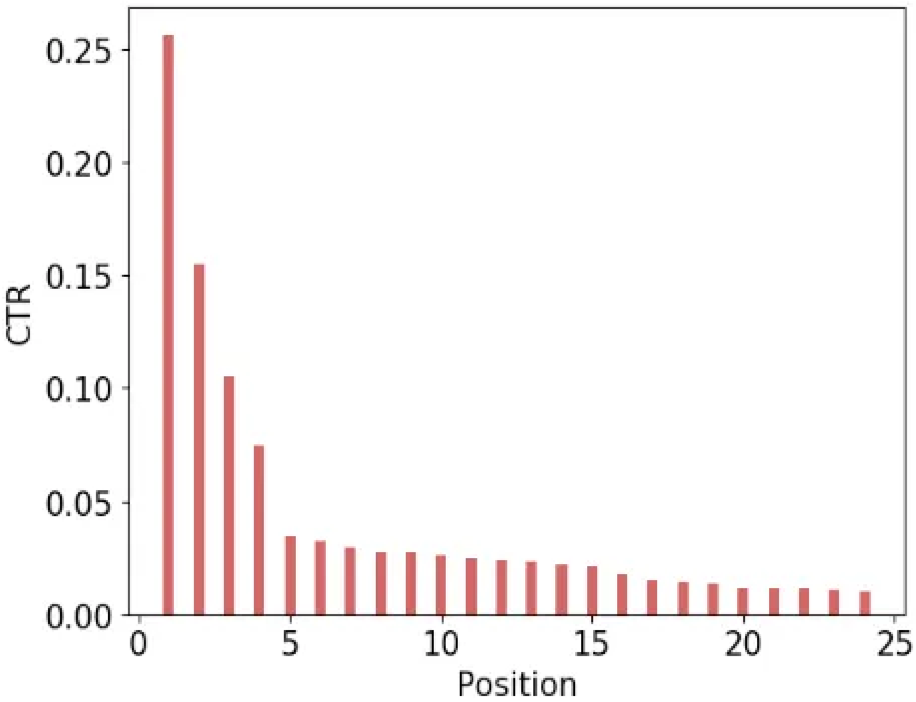
In this post, we'll take a deep-dive into one particular modeling approach that has relatively recently enabled the industry to control biases very effectively and thus build vastly superior recommender systems: the two-tower model, where one tower learns relevance and another (shallow) tower learns biases.

While two-tower models have probably been used in the industry for several years, the first paper to formally introduce them to the broader ML community was Huawei's 2019 PAL paper.

PAL (Huawei, 2019) — the OG two-tower model

Huawei's paper PAL ("position-aware learning to rank") considers the problem of position bias within the context of the Huawei app store.

Position bias has been observed over and over again in ranking models across the industry. It simply means that users are more likely to click on items that are shown first. This may be because they're in a hurry, because they blindly trust the ranking algorithm, or other reasons. Here's a plot demonstrating position bias in Huawei's data:



Position bias. Source: Huawei's paper PAL

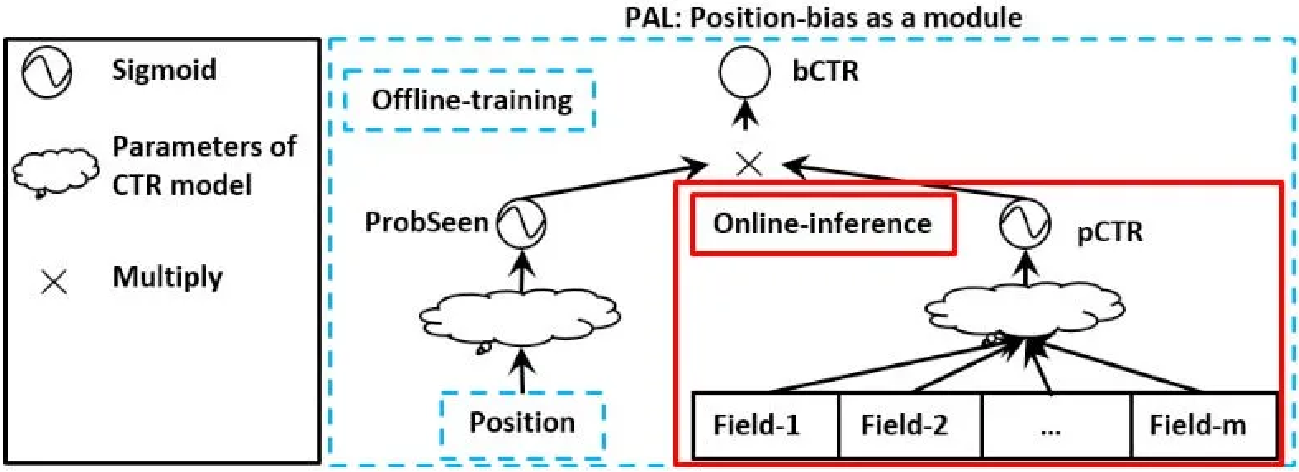
Position bias is a problem because we simply can't know whether users clicked on the first item because it was indeed the most relevant for them or because it was shown first — and in recommender systems we aim to solve the former learning objective, not the latter.

The solution proposed in the PAL paper is to factorize the learning problem as

```
p(click|x,position) = p(click|seen,x) x p(seen|position),
```

where x is the feature vector, and `seen` is a binary variable indicating whether the users has seen the impression or not. In PAL, `seen` depends only on the position of the item, but we can add other variables as well (as we'll see later).

Based on this framework, we can then build a model with two towers, each of which outputs one of the two probabilities on the right hand side, and then simply multiply the two probabilities:



Source: Huawei's paper PAL

The clouds in this image are simply neural networks: a shallow one for the position tower (because it only needs to process a single features), and a deep one for the CTR tower (because it needs to process a large number of features and create interactions between those features). We also call these two towers the bias tower and engagement tower, respectively.

Notably, at inference time, where positions aren't available, we run a forward pass only using the engagement tower, not the bias tower. Similar to Dropout, the model thus behaves differently at training time at inference time.

Does it work? Yes, indeed, PAL works remarkably well. The authors build 2 different version of the DeepFM ranking model (which I wrote about here), one version with PAL and baseline version with a naive way of treating item position: simply passing it as a feature into the engagement tower. Their online A/B test shows that PAL improves both click-through rates and conversion rates by around 25%, a huge lift!

PAL showed that positions themselves can be used as inputs to a ranking model, but they need to be passed through a dedicated tower, not the main model (a rule that has also been added as Rule 36 to Google's "Rules of ML"). The two-tower model was officially born — even though it has been likely used across the industry prior to the PAL paper.

"Watch Next" (YouTube, 2019) — the additive two-tower model

YouTube's paper "Recommending What Video to Watch Next" — commonly known simply as the "Watch Next" paper — came out around the same time as PAL and tried to solve the same problem: debiasing of recommender systems using a two-tower model. However, compared to PAL, YouTube used an *additive* two-tower model instead of a multiplicative one.

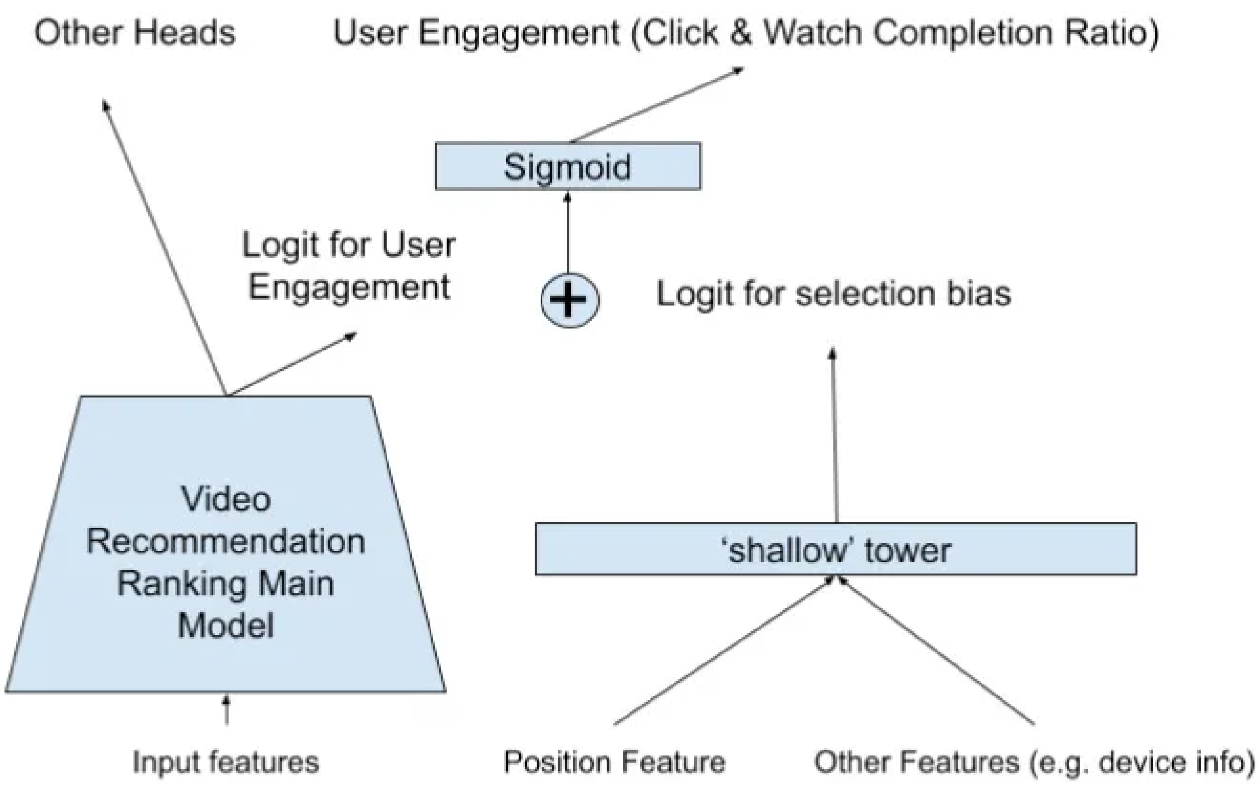
To see why this works, consider again the factorization of the learning objective:

```
p(click|x,position) = p(click|seen,x) x p(seen|position)
```

Because probabilities are sigmoids of logits, sigmoids are simply weighted exponential functions, and the exponential function follows the product rule for exponents, we can also write:

```
logit(click|x,position) = logit(click|seen,x) + logit(seen|position),
```

which is, alas, the two-tower additive model.



The two-tower additive model. Source: "Recommending What Video to Watch Next"

Another notable innovation compared to PAL is the use of other features besides position in the shallow tower. One example is the user device type. This makes sense because different devices are expected to exhibit different forms of position bias: for example, position bias may be more severe on phones with smaller screens where the user can see fewer positions at the same time and has to swipe more. As such, the shallow tower in YouTube's model doesn't just learn position bias, it learns all sorts of selection biases, depending on which features are being passed into the shallow tower.

In order to ensure the shallow tower actually leverages all of these features, the authors add a Dropout layer with a dropout probability of 10% on top of the position feature alone. Without it, the model may over-rely on the position feature and not learn any other selection biases.

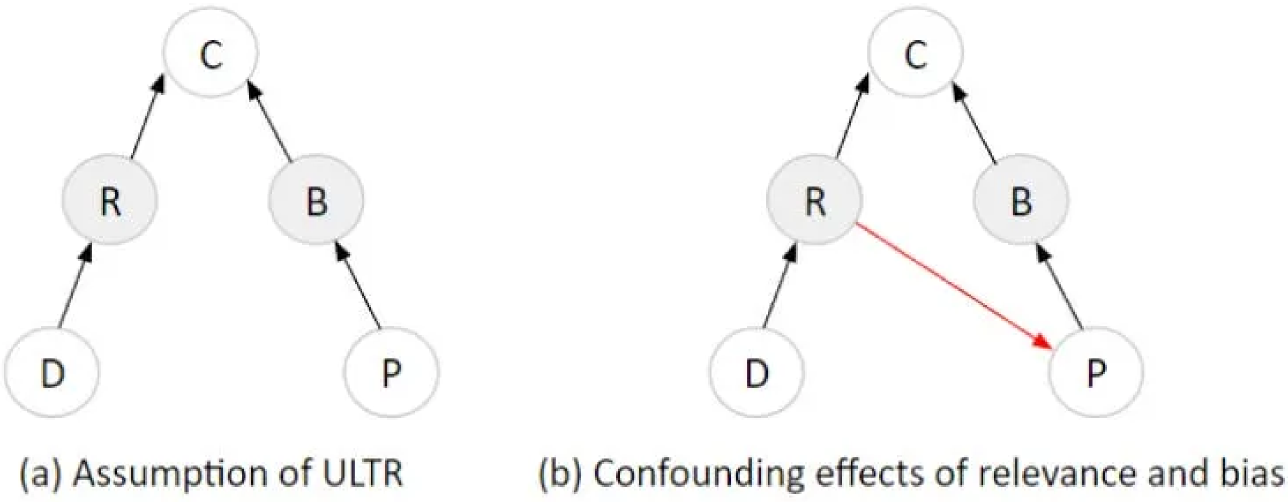
Using A/B testing, the authors show that adding the shallow tower improves their "engagement metric" (unfortunately no details were given on what that metric is) by 0.24%.

Disentangling relevance and bias in ULTR (Google, 2023)

Up until this point, we've made the assumption that the two towers in ULTR can learn independently during model training. Alas, this assumption is not true, show the authors of recent Google paper, "Towards Disentangling Relevance and Bias in Unbiased Learning to Rank".

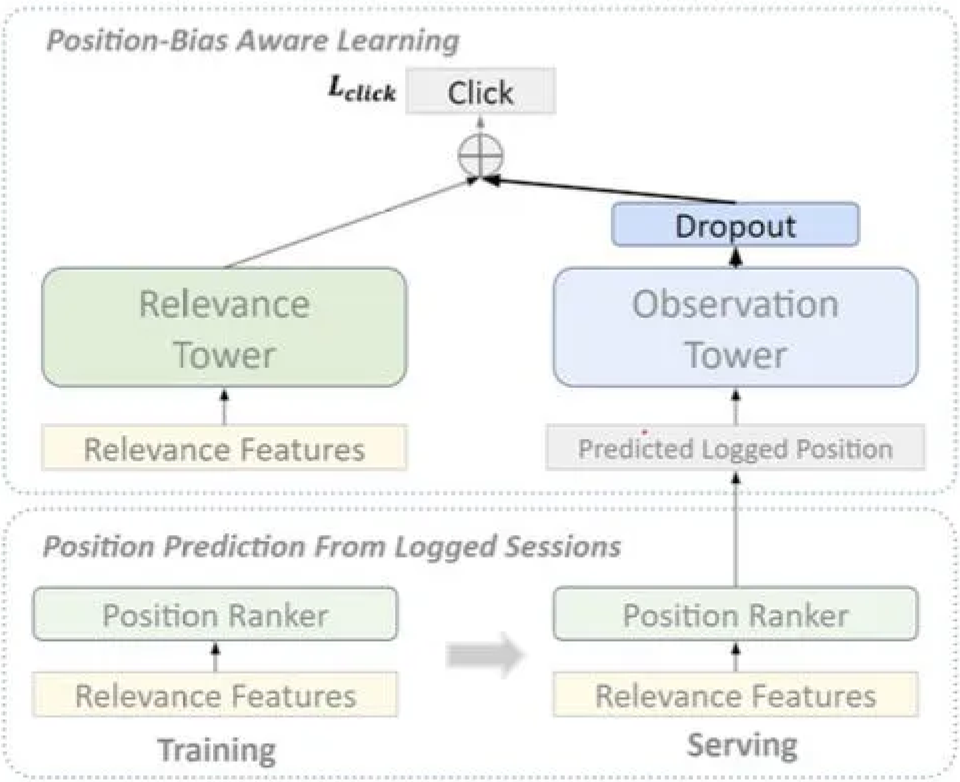
Here's a Gedankenexperiment to see why: suppose there's a perfect relevance model, that is, a model that can perfectly explain clicks. In such a case, once we re-train the model on new data, all the bias tower needs to do is learn how to map positions to clicks. The relevance tower wouldn't have to learn anything — it would be entirely useless!

In other words, relevance has a confounding effect on position, as shown in the red arrow in the causal graph below.



The confounding effect of the relevance tower on the bias tower. Source: Zhang et al 2023

But that's not what we want: we want the relevance tower to keep learning as new data comes in. One simple method to solve this problem — eliminate the red edge in the causal graph — is by adding a Dropout layer on top of the bias logit, as shown in the plot below:



Adding dropout on top of the bias tower. Source: Zhang et al 2023.

Why Dropout? The key idea is that by randomly dropping out the bias logit, we "nudge" the model to rely a little bit more on the relevance tower instead of simply learning the mapping between historic positions and relevance. Note that this idea is very similar to YouTube's Watch Next paper, however here we drop out the entire bias logit instead of only the position feature.

In order for this to work, the logit dropout probability needs to be high: with a dropout of 0.5 (that is, randomly zeroing out the bias logit in 50% of the training examples) the authors beat PAL by 1% click NDGC on production data from the Chrome App store — a strong demonstration of this simple technique.

Summary

Let's recap:

- The two-tower model powerful approach to build unbiased ranking models: one tower learns relevance while the other tower learns position bias (and, optionally, other biases).
- We can combine the output from the two towers either by multiplying their probabilities (as done in Huawei's PAL), or by adding their logits (as done in YouYube's Watch Next). We call the latter the additive two-tower model.
- The input to the bias tower is usually the position of the item, but we can also add other features that introduce bias such as the user device type.
- Dropout (either on the positions or on the entire bias logit) prevents the model from over-relying on historic positions and has been shown to improve generalization.

And that's just the tip of the iceberg. Given the importance of recommender systems in today's tech industry, and the discovery that two-tower modeling can substantially improve predictive performance, this research domain is probably still far from its full potential: we're really just getting started!

If you like this content and want to stay on top of the latest technologies in ML, make sure to subscribe.

Machine Learning | Artificial Intelligence | Data Science | Technology | Editors Pick

Follow

Written by Samuel Flender

3K Followers · Writer for Towards Data Science

Cutting-edge ML research, simplified. For the latest, subscribe to my Newsletter: mlfrontiers.substack.com

---

More from Samuel Flender and Towards Data Science

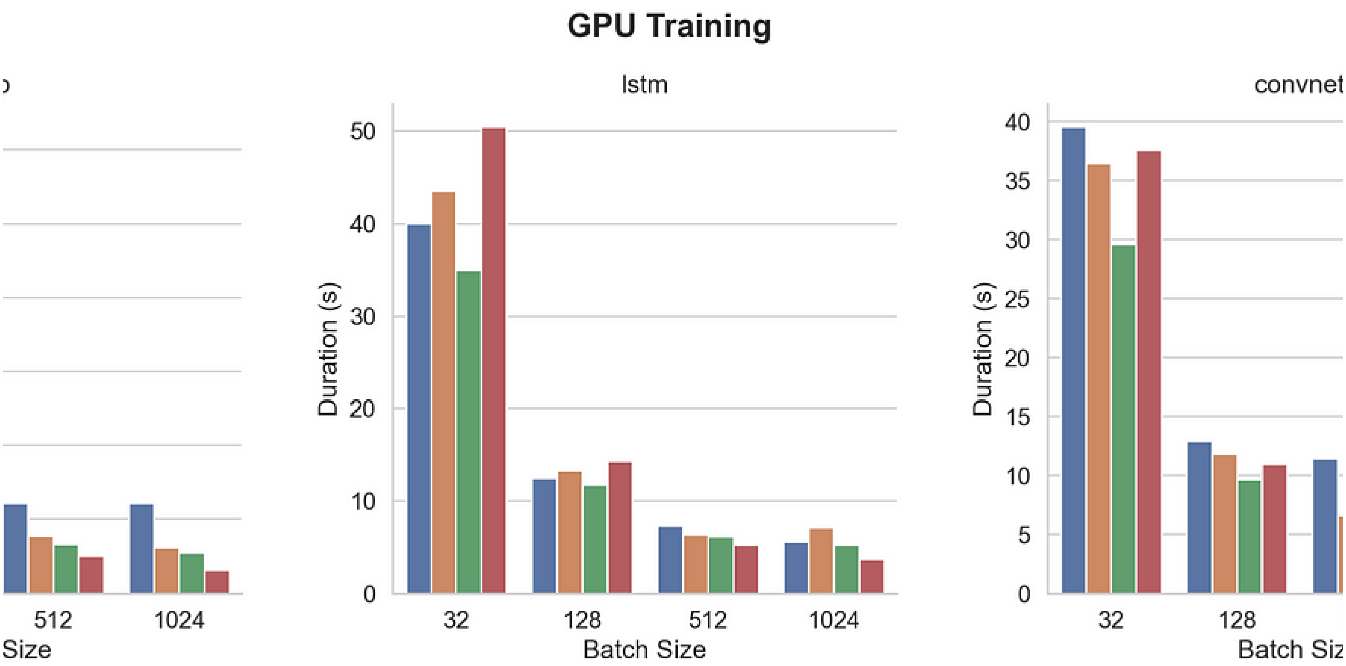
Samuel Flender in Towards Data Science

How I Cracked the Meta Machine Learning Engineering Interview

Practical tips for the coding, design, and behavior rounds

· 7 min read · Oct 25, 2022

1.5K 8



Fabrice Daniel in Towards Data Science

Apple M2 Max GPU vs Nvidia V100, P100 and T4

Compare Apple Silicon M2 Max GPU performances to Nvidia V100, P100, and T4 for training MLP, CNN, and LSTM models with TensorFlow.

· 6 min read · Nov 2

881 14

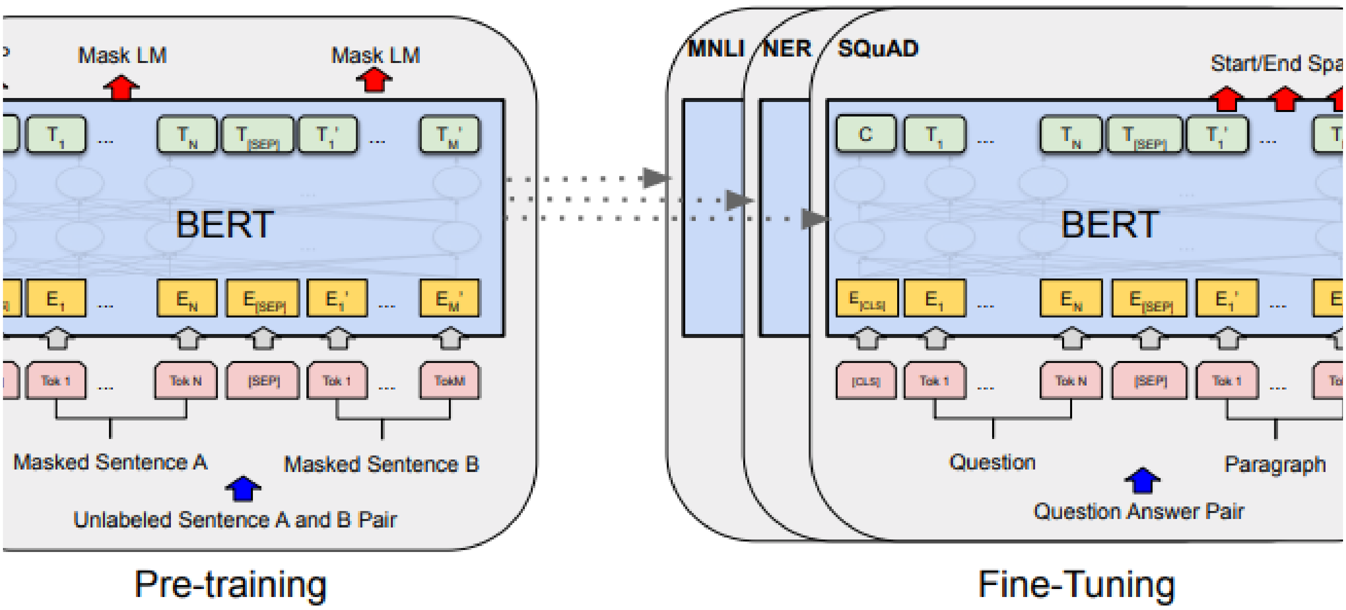
Mike Shakhomirov in Towards Data Science

Python for Data Engineers

Advanced ETL techniques for beginners

· 17 min read · Oct 21

487 4



Samuel Flender in Towards Data Science

What exactly happens when we fine-tune BERT?

A closer look into some of the recent BERTology research

· 6 min read · Feb 21, 2022

279 3

See all from Samuel Flender

See all from Towards Data Science

Recommended from Medium

Cassie Kozyrkov

Why I quit my job as Google's Chief Decision Scientist

What's it like to go from being Chief Decision Scientist at Google to being, well… just me?

· 7 min read · Oct 28

7.6K 99

 The PyCoach in Artificial Corner

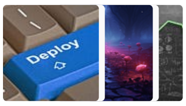
OpenAI Just Released GPTs: Create Your Own ChatGPT And Make Money From It (No Coding Required)

We're about to see the App Store for AI.

· 4 min read · 5 days ago

2.5K 50

Lists

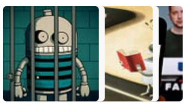


Predictive Modeling w/ Python

20 stories · 587 saves

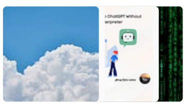
ChatGPT prompts

30 stories · 644 saves



AI Regulation

6 stories · 182 saves



ChatGPT

22 stories · 252 saves

Vaishnav Manoj in DataX Journal

JSON is incredibly slow: Here's What's Faster!

Unlocking the Need for Speed: Optimizing JSON Performance for Lightning-Fast Apps and Finding Alternatives to it!

16 min read · Sep 28

8.4K 111

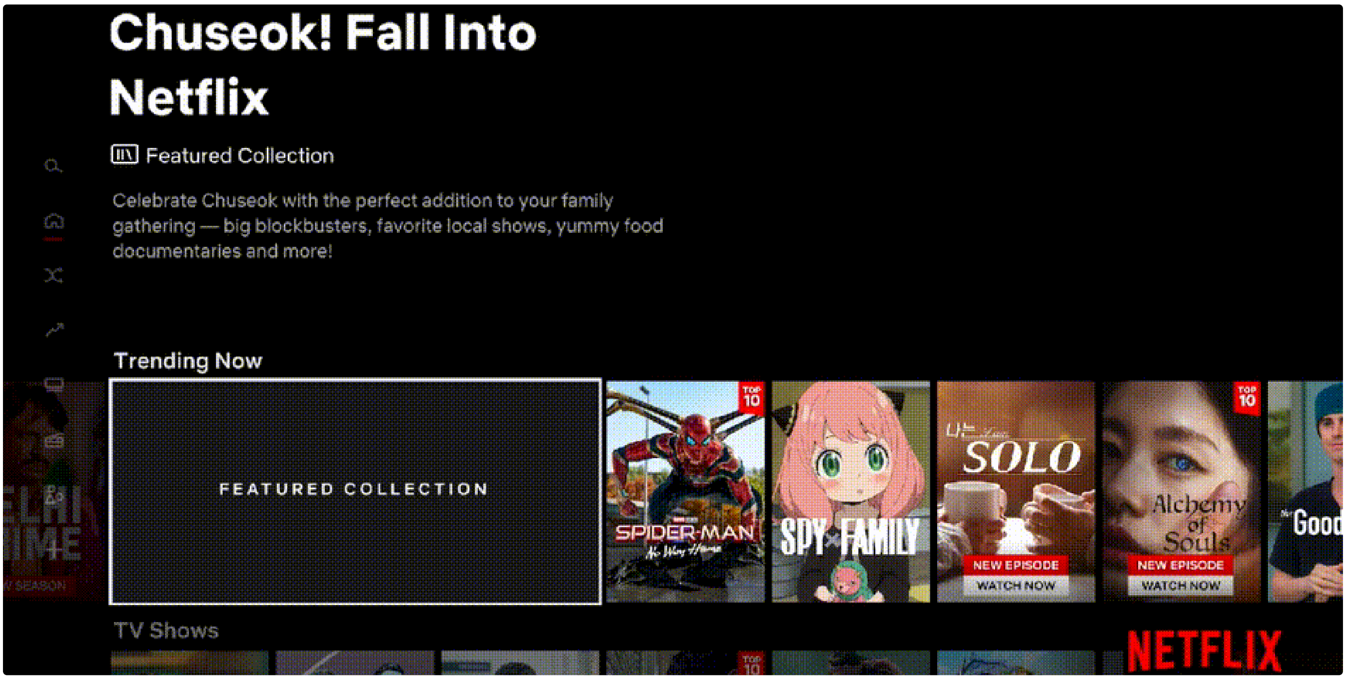
Barack Obama

Thoughts on Israel and Gaza

It's been 17 days since Hamas launched its horrific attack against Israel, killing over 1,400 Israeli citizens, including defenseless...

5 min read · Oct 23

39K 989



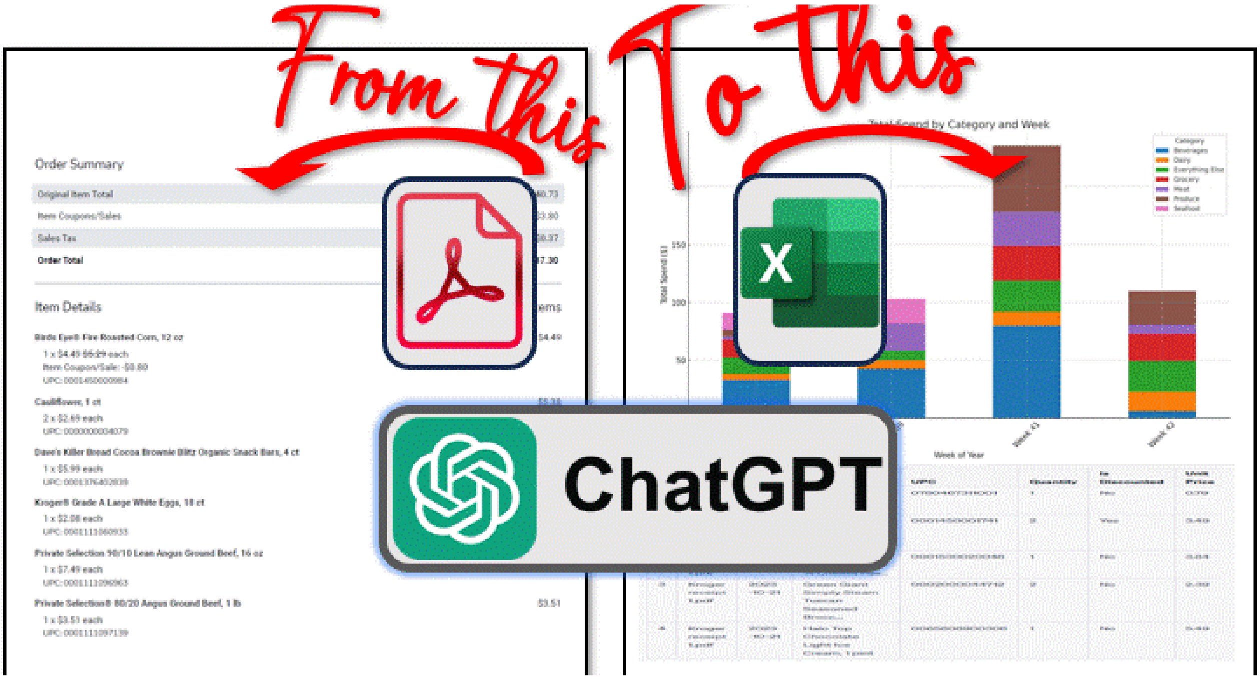
Netflix Technology Blog in Netflix TechBlog

The Next Step in Personalization: Dynamic Sizzles

Authors:Bruce Wobbe, Leticia Kwok

7 min read · 4 days ago

882 21



David Leibowitz in Towards AI

ChatGPT As an OCR For PDFs: Your ETL Engine for Data Analysis

Coding in English at the speed of thought

· 14 min read · Nov 3

750 8

See more recommendations