

# RMGPT: Transformers as Revenue Managers

Kalyan Talluri, Hanzhao Wang  
Imperial College Business School

# Background



# Motivation: Generative AI to assist the Revenue Manager

Generative AI is all the rage

1. ChatGPT/OpenAI
2. Gemini
3. Co-pilot
4. ...

What can it do to the Revenue Manager

# I ask AI a simple question: How should I price my hotel room on 2 July 2024

Ask AI



depend on various factors, including your costs, your target market, your competition, and the value that your product or service provides to your customers.

To determine an appropriate price, you could consider conducting market research to understand your target customers' willingness to pay and how much your competitors are charging for similar products or services. You could also calculate your costs, including materials, labor, overhead, and any other expenses related to producing or delivering your product or service. Additionally, you could consider the unique value proposition of your product or service and how much customers would be willing to pay for that value.

Ultimately, finding the right price for your product or service can be an iterative process that requires experimentation and adjustment over time.

# Large Language Models (LLM)

- Very impressive Natural Language Processing (NLP)
- Quiet effective Chatbots
- Very effective in Image Recognition and Classification
- Require tremendous computing power (\$1 billion and upwards for each foundational model)
- Require tremendous amounts of training data
  - Image banks, Wikipedia, Library of Congress, News ...
- Big Research question
  - What can it do with small dynamic changeable data to help the Revenue Manager?

# Challenges of AI in the RM context

- Limited history
- Unobservables
  - Size of the market for a particular day in the future
  - Customer behaviour
  - Alternates
- Competition (their sales, their future prices)



# Our Research Project (Hanzhao, Li)

- Small available data typical of RM and Operations
- Develop LLM style architectures
- Methodologies for predicting and setting prices
- Goal: Superior to hand-coded models
  - Off-the shelf transformer models
  - Off-the shelf optimization engines



# Pricing and Learning

Pricing → Learning → Deep Learning



# Pricing with Known Demand Model

- Task: price a product in a well-known market
- Assume the true demand model of the market is known:



- Object: maximize the revenue (assume no cost) by choosing the price:  
Revenue=Demand × Price
- Demand Model:  $D=20-P$  → Revenue:  $R=(20-P) \times P$  → Optimal Price: £10

# Pricing with History Data

- When the true demand model is unknown
  - Assume there is history data collected from the same market, i.e., from last year
  - Estimate a demand model through the data
  - Choose the price to maximize the **estimated** revenue from the **estimated demand model**

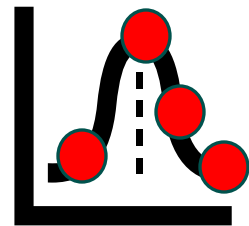
History Data -> Est. Demand Model:  $D=20-P$  -> Est. Revenue Model:  $R=(20-P) \times P$  -> Est. Optimal Price: £10

Month	Jan	Feb	Mar	Apr	May	June	July
Price	£5	£15	£10	£16	£10	£10	£10
Demand	15	5	10	4	10	10	10
Revenue	75	75	100	64	100	100	100

# “Rich” Data vs. “Poor” (uninformative) Data

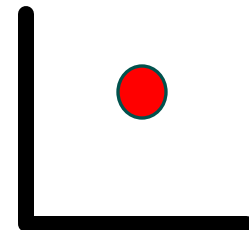
“Rich”

Month	Jan	Feb	Mar	Apr	May	June	July
Price	£5	£15	£10	£16	£10	£10	£10
Demand	15	5	10	4	10	10	10
Revenue	75	75	100	64	100	100	100



“Poor”

Month	Jan	Feb	Mar	Apr	May	June	July
Price	£5	£5	£5	£5	£5	£5	£5
Demand	15	15	15	15	15	15	15
Revenue	75	75	75	75	75	75	75



# Pricing without “Rich” History Data

- When only “poor” history data available or no data at all (new market)
  - May need some time to learn this new market (e.g., conduct experimental prices)
  - Task: **dynamic pricing** in **a time interval**
    - Time interval: need some time to learn
    - Dynamic pricing: update the price after collecting more data
  - Object: maximize the **cumulative** revenue over the time interval
    - Count the opportunity cost from experimental prices

# One Strategy: Greedy Pricing

In each month:

1. Estimate demand model from previously collected data (from previous months)
2. Estimate the optimal price from the estimated demand model, and apply it in this month

Example: true demand model is

Demand=20-Price, where the optimal price=£10

Month	Jan	Feb	Mar	April	...
Est. Demand Model	-				
Est. Optimal Price	£5				
Est. Demand	-				
True Demand	15				

# Failure of Greedy Pricing

In each month:

1. Estimate demand model from previously collected data (from previous months)
2. Estimate the optimal price from the estimated demand model, and apply it in this month

Example: true demand model is

$$\text{Demand} = 60 - 10 \times \text{Price}, \text{ where the optimal price} = \text{£}3$$

Month	Jan	Feb	Mar	April	...
Est. Demand Model	-				
Est. Optimal Price	£5				
Est. Demand	-				
True Demand	10				

# Another Strategy: Pricing with Explorations

## Greedy Pricing

1. Estimate demand model from collected data
2. Estimate the **optimal price** from the estimated demand model, and apply it

## Pricing with Explorations

1. Estimate demand model from collected data
2. Estimate the **optimal price** from the estimated demand model, and apply it **with some perturbation when uncertain**

Month	Jan	Feb	Mar	April	...
Est. Demand Model	-				
Applied Price	£5				
Est. Demand	-				
True Demand	10				

True demand model: Demand=60-10×Price, where the optimal price=£3



# Third Strategy: Pricing with Data from Similar Markets



- History data is collected from London
- But need to price in New York

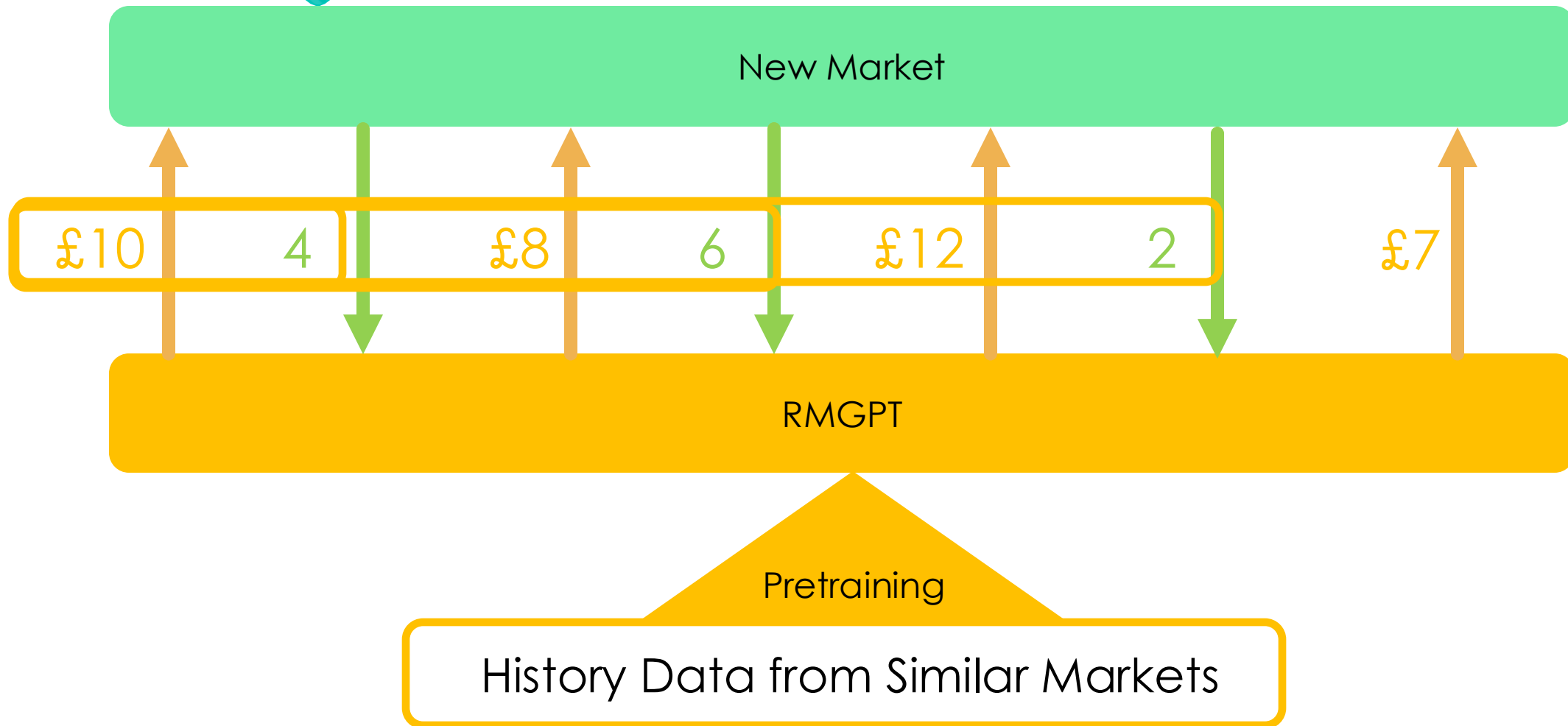
History data can be collected from more markets: London, Beijing, Tokyo,...

Question: how to leverage the similar history data when pricing in a new market?

# Our Solution: RMGPT

- Revenue Management Generative Pretrained Transformer (RMGPT) is a deep learning model designed to solve revenue management tasks (e.g., dynamic pricing in a new market)
- It can leverage the history data from similar markets (with the new collected data from the new market) when making decisions

# RMGPT as Revenue Manager



# ChatGPT vs. RMGPT

## ChatGPT

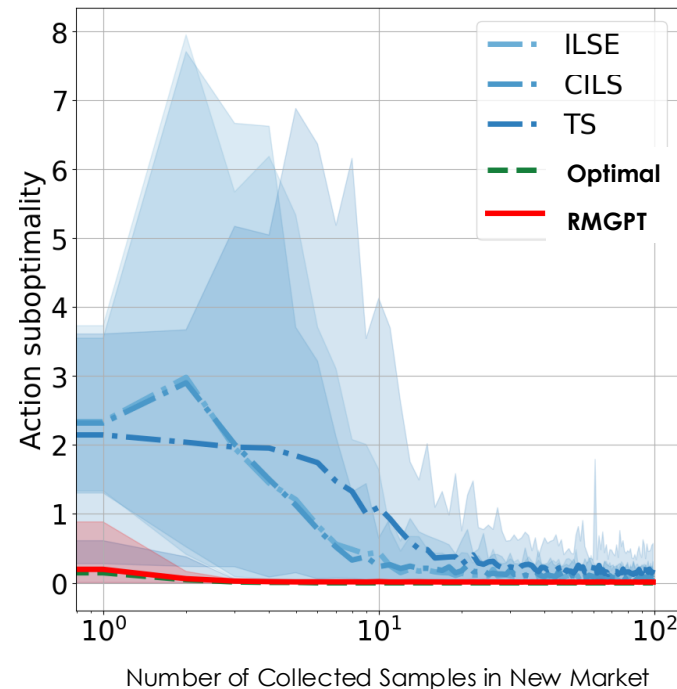
- Generative: text based on prompt text
  - Prompt: Tell me a story
  - Generate: Here is a story...
- Pretrained: trained by texts
  - Websites
  - News
  - Books

## RMGPT

- Generative: price based on the data collected from the **new market**
  - Prompt: (£15, 5), (£5, 15),... (from the new market)
  - Generate: £10 (as the price to apply)
- Pretrained: trained by history data (and good decisions) from similar markets
  - Market 1:
    - Data: (£10, 5), (£5, 20),..., Optimal price: £5
  - Market 2:
    - Data: (£8, 15), (£20, 5),..., Optimal price: £8
  - ...

# Why RMGPT? Power of Pretraining

- Through pretraining, RMGPT can utilize the knowledge of making good decisions from similar markets:
- When a new market is close to one similar market in pretraining data: less cost of exploration -> more revenues

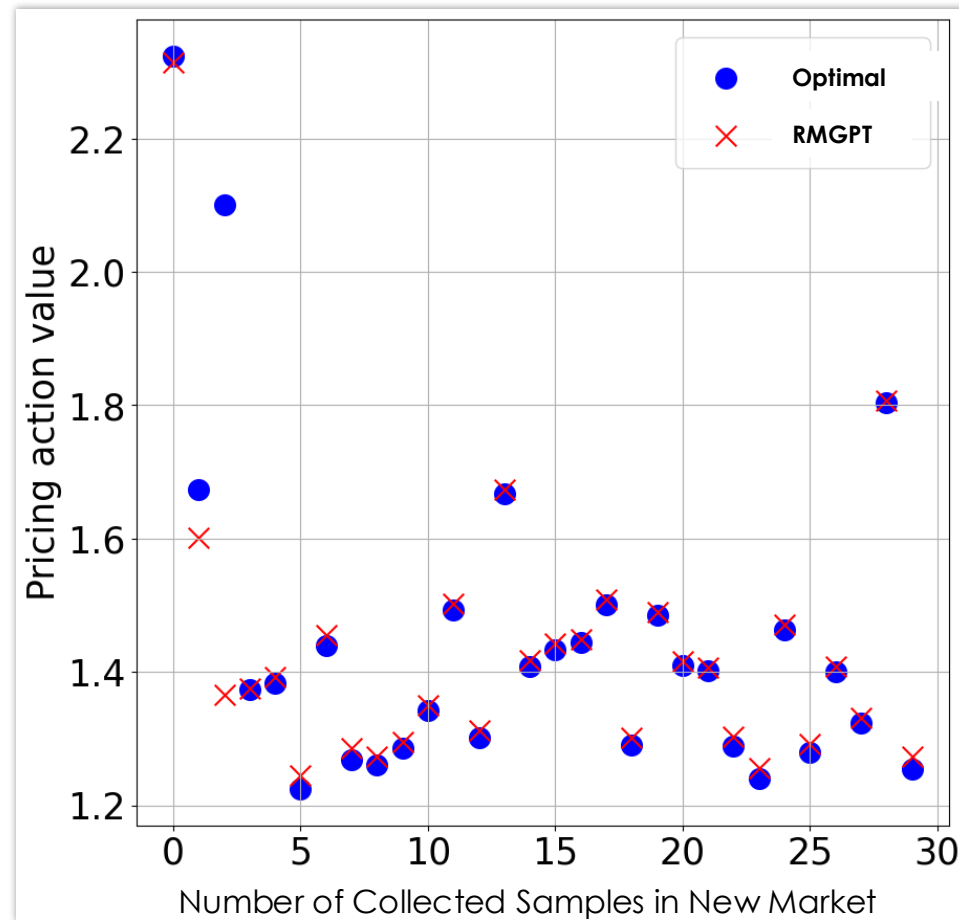


# Decisions from RMGPT

- When the history data is **infinite** in each similar market
  - RMGPT's decision is the **optimal decision** based on the information **from both the history data (from similar markets) and the new collected data (from the new market)**
- Exploration?
  - When the history data is **finite** in each similar market (always in practice), RMGPT's decisions have explorations

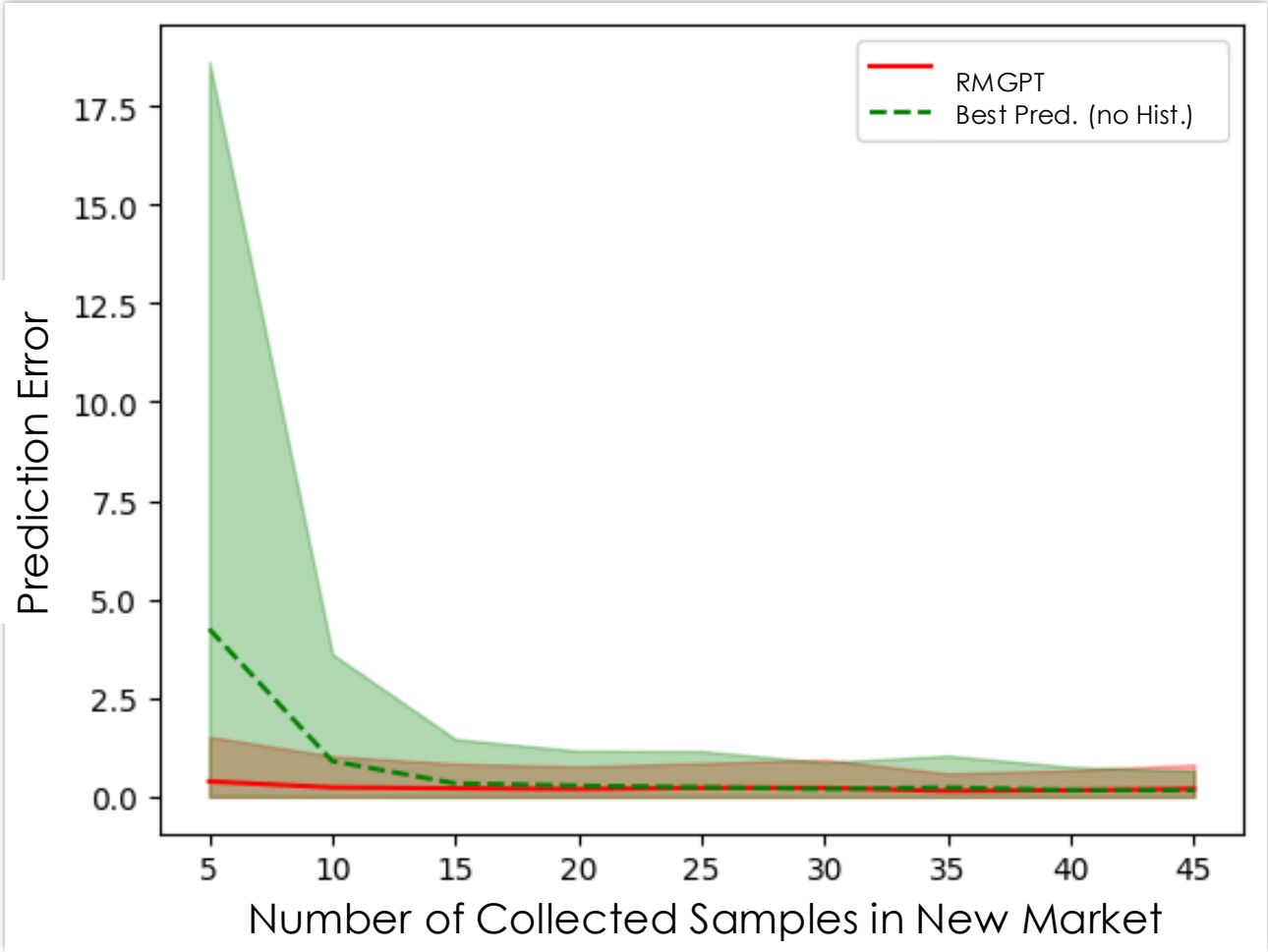
Optimal pricing with explorations

# Decisions from RMGPT: Visualization



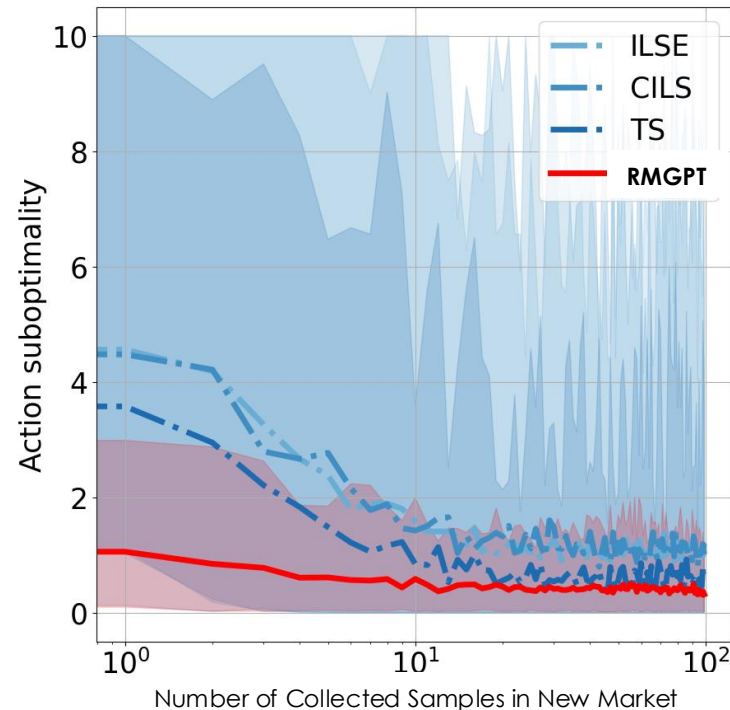


# By-product: Demand Prediction



# Solution of Model Mis-specification

- Pricing strategies are usually designed for only one demand type (e.g., economic class), and will fail if the true demand is different (e.g., business class)
- For RMGPT, we can add all possible demand types in the pre-training



# Takeaways

- Pricing with history data: “rich”
- Pricing without “rich” history data: explorations
- Pricing with history data from similar markets: RMGPT
  - It leverages both the information from similar markets and the new market
  - The information from similar markets can reduce the cost of explorations in the new market
  - It makes near optimal decisions (with explorations)
- RMGPT can be trained to solve other tasks: inventory control, queueing control, ...

**Thank you!**



**Code and Paper**