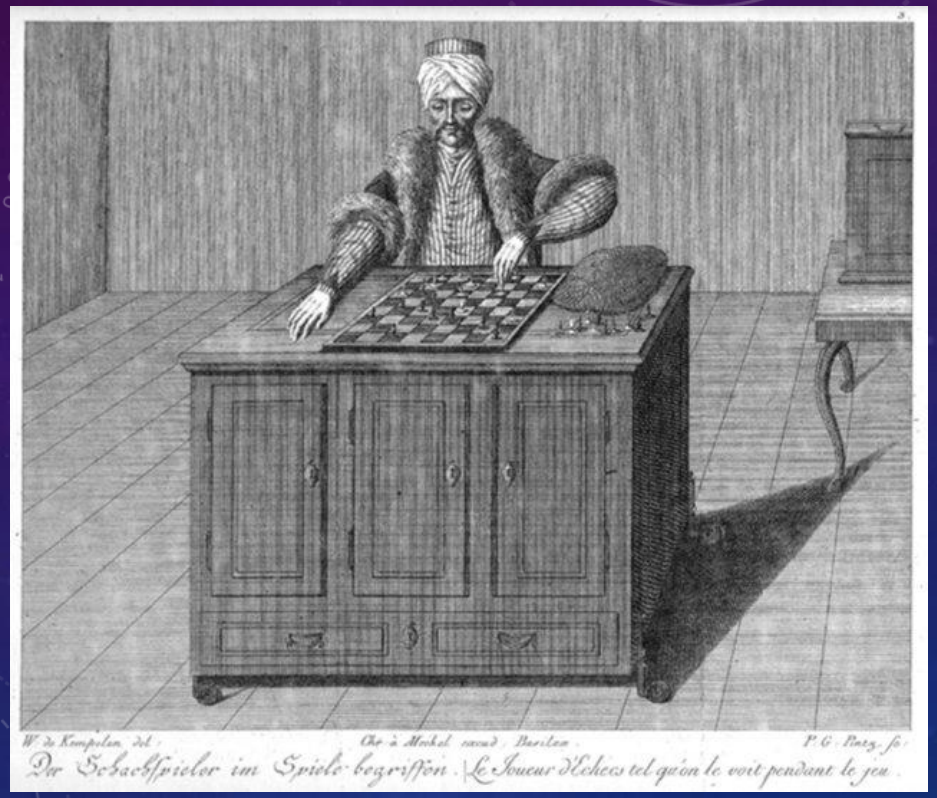
AI ENGINEERING

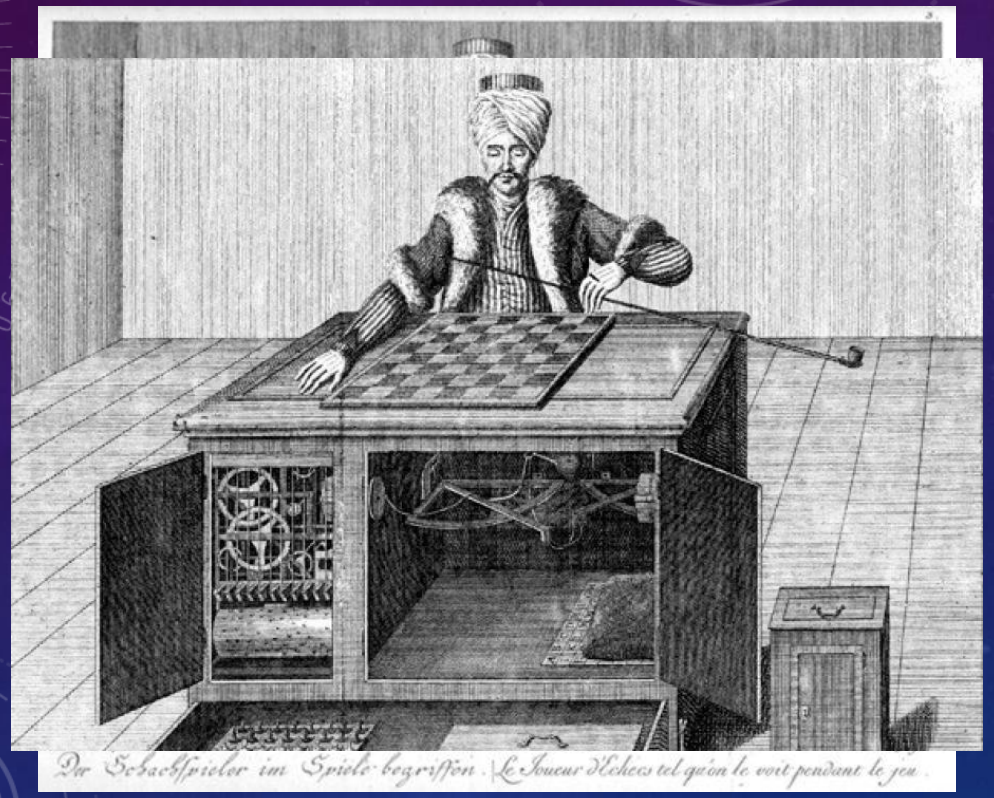


PROF. LEV MUCHNIK

2025

העברית הכשרת מנהלים
בית הספר למנהל עסקים
האוניברסיטה העברית בירושלים
האקדמיה להייטק

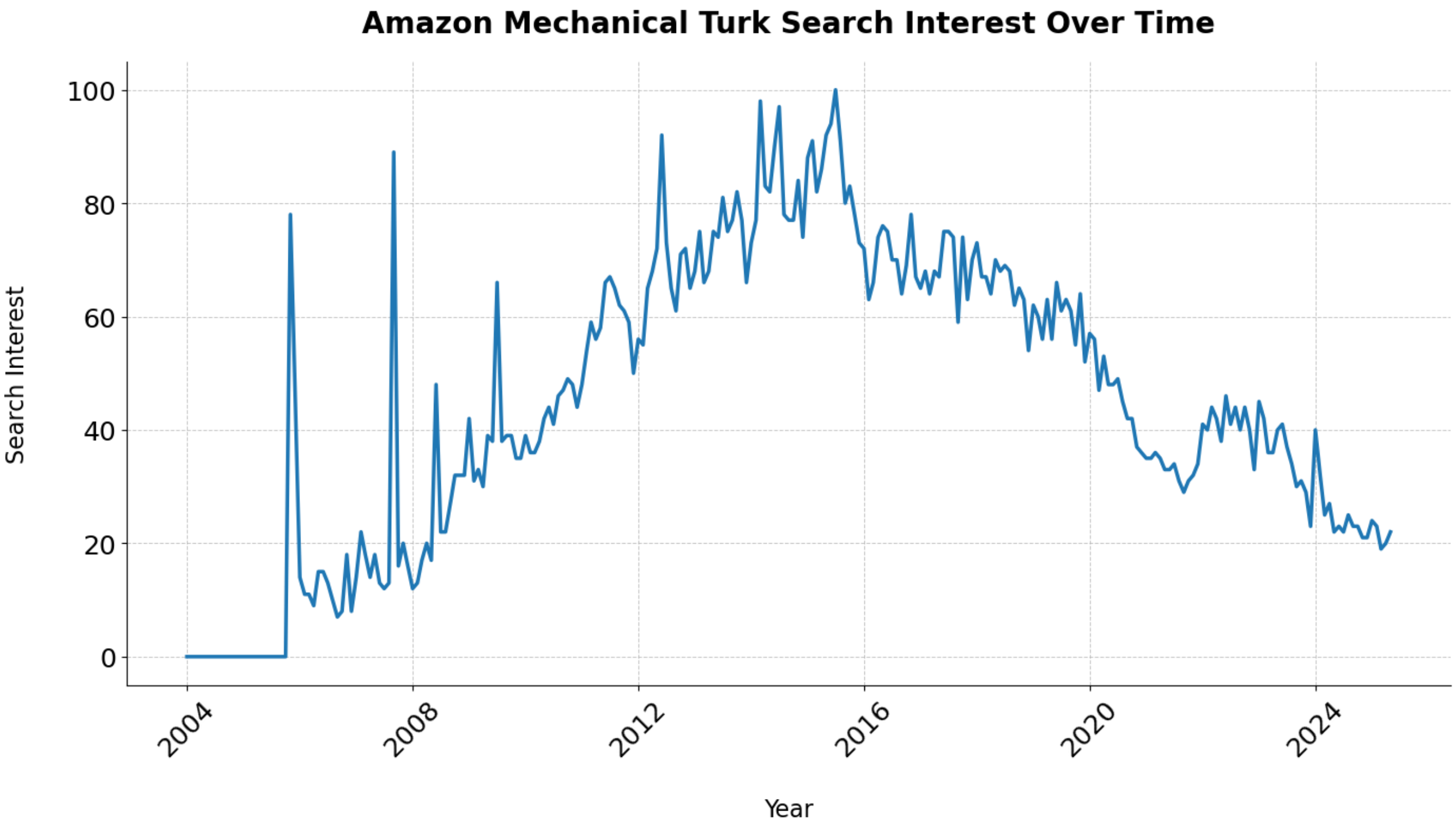
AI ENGINEERING

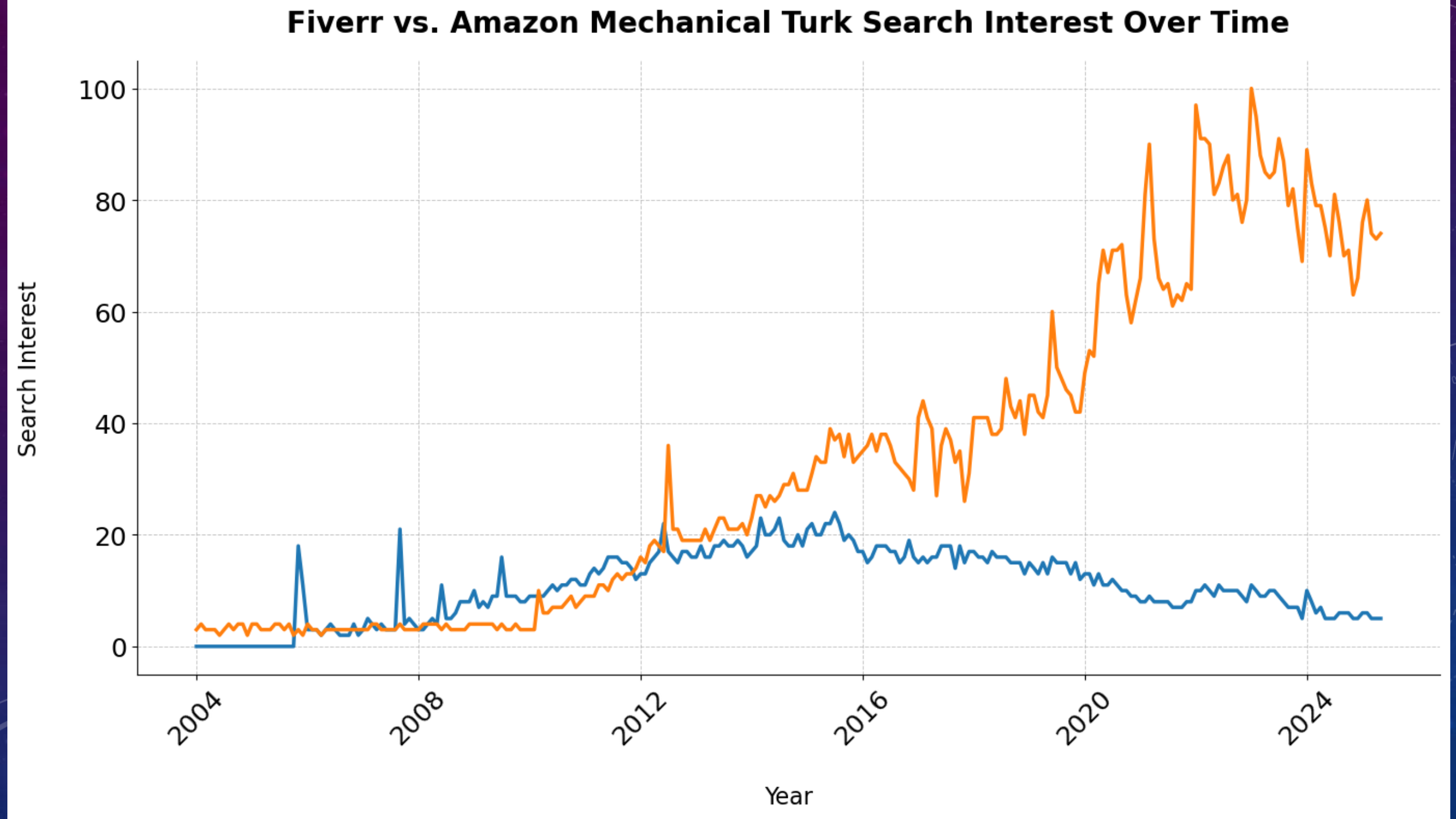


PROF. LEV MUCHNIK

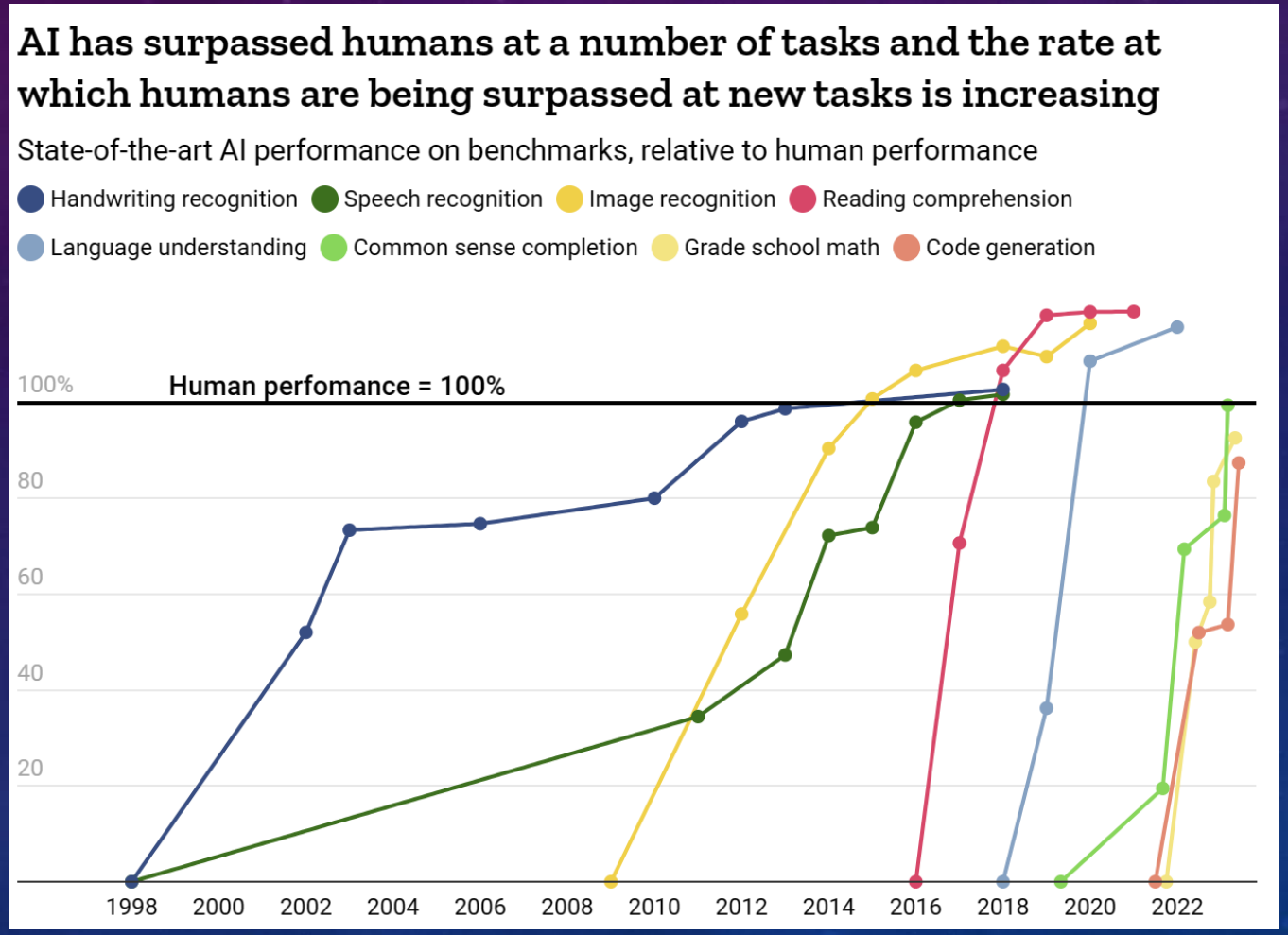
2025

העברית הכשרת מנהלים
בית הספר למנהל עסקים
האוניברסיטה העברית בירושלים
האקדמיה להייטק

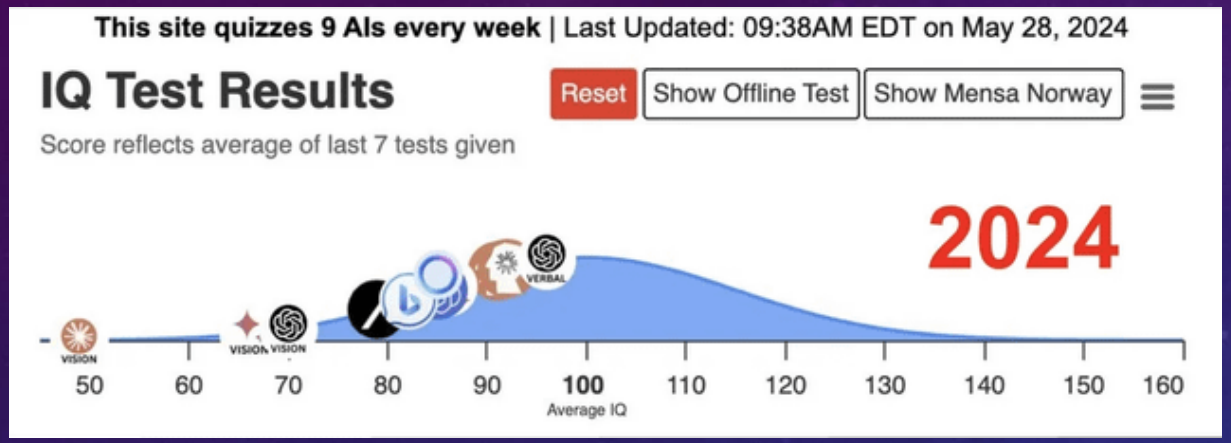

Amazon Mechanical Turk Search Interest Over Time
Search Interest
100
80
60
40
20
0
2004
2008
2012
2016
2020
2024
Year

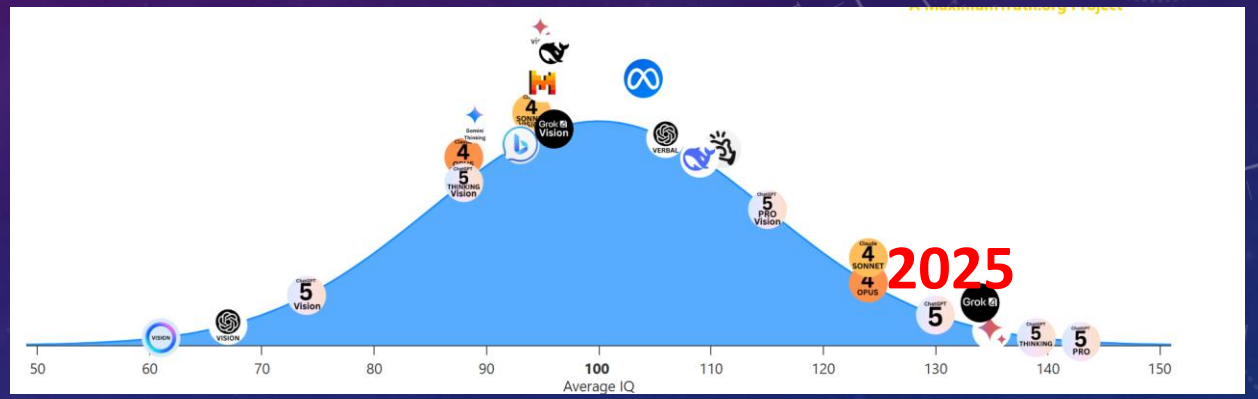

Fiverr vs. Amazon Mechanical Turk Search Interest Over Time
Search Interest
100
80
60
40
20
0
2004
2008
2012
2016
2020
2024
Year

THE RATE OF AI PROGRESS



THE “~~IPHONE~~ AI MOMENT”



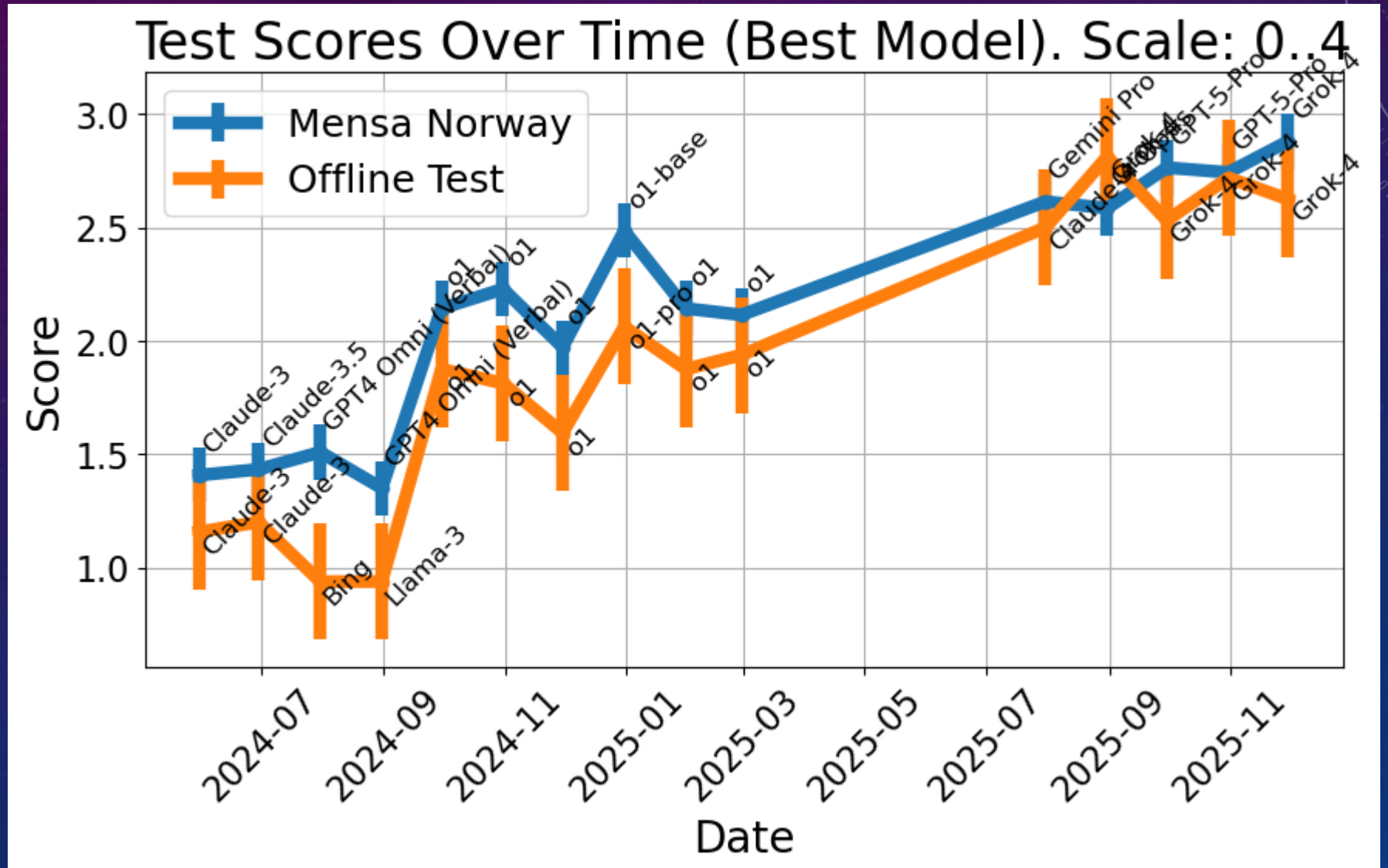


https://trackingai.org/

All data:
https://trackingai.org/app/database/all_ai_replies.csv.gz

tracking_ai_analysis.ipynb

THE "~~IPHONE~~ AI MOMENT"



HUMANITY'S LAST EXAM

Here is a representation of a Roman inscription, originally found on a tombstone. Provide a translation for the Palmyrene script.
A transliteration of the text is provided: RGYNᵒ BT ḤRY BR ᶜTᵒ ḤBL

https://scale.com/leaderboard/humanitys_last_exam

Performance Comparison

1	gpt-5-2025-08-07 NEW	25.32 ±1.70	Calib Err: 50
2	gemini-2.5-pro-preview-06-05	21.64 ±1.61	Calib Err: 72
2	o3 (high) (April 2025)	20.32 ±1.58	Calib Err: 34
2	gpt-5-mini-2025-08-07 NEW	19.44 ±1.55	Calib Err: 65
2	o3 (medium) (April 2025)	19.20 ±1.54	Calib Err: 39
3	Gemini 2.5 Pro Experimental (March 2025)	18.16 ±1.51	Calib Err: 71
3	o4-mini (high) (April 2025)	18.08 ±1.51	Calib Err: 57
3	Gemini 2.5 Pro Preview (May 06 2025)	17.80 ±1.50	Calib Err: 70
9	o4-mini (medium) (April 2025)	14.28 ±1.37	Calib Err: 59
9	Gemini 2.5 Flash (April 2025)	12.08 ±1.28	Calib Err: 80
10	claude-opus-4-1-20250805-thinking NEW	11.52 ±1.25	Calib Err: 71
10	Gemini 2.5 Flash Preview (May 2025)	10.96 ±1.22	Calib Err: 82
10	Claude Opus 4 (Thinking)	10.72 ±1.21	Calib Err: 73

COURSE PHILOSOPHY: LASTING PRINCIPALS

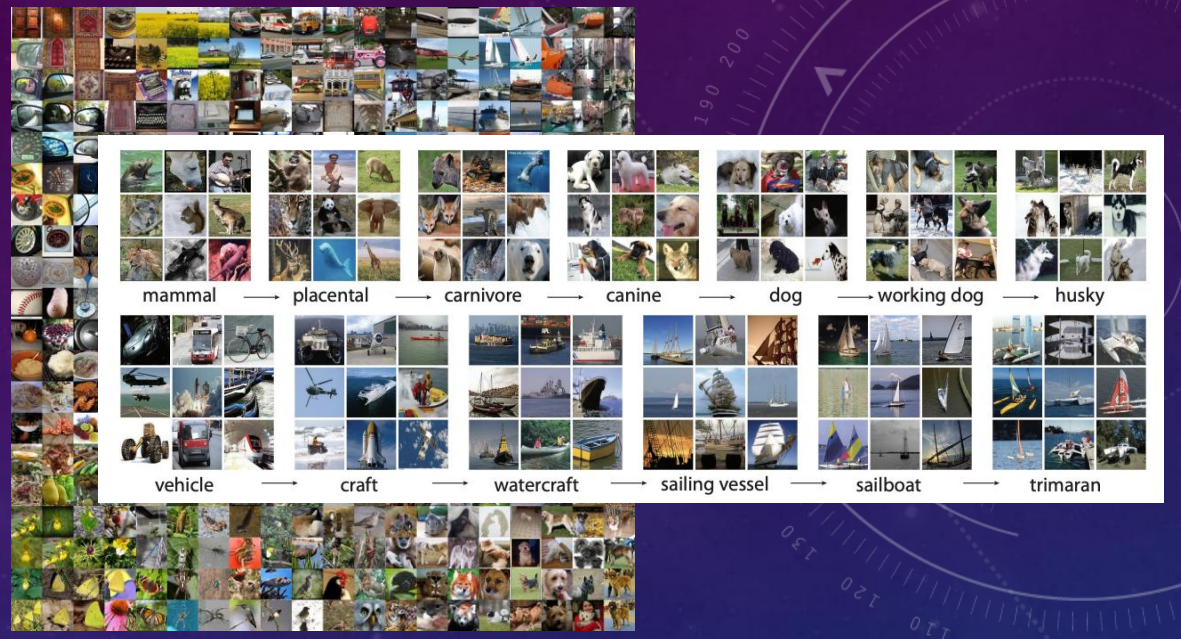
- **Python is the tool**

- **Data is the Fuel:** Understanding, collecting, cleaning, and preparing (Module 1)

- **Building the Toolbox:**
  - **Specialized, classical ML models – basic principals – model evaluation** (Module 1)
  - **NLP, PyTorch & Deep Learning (Module 2)**
  - **Transformers and LLMs (Module 3)**
  - **AI-based systems (Module 4)**

Year (July 1)	Population	Yearly % Change	Yearly Change	Median Age	Fertility Rate	Density (P/Km²)	Urban Pop %	Urban Population
2020	**7,794,798,739**	1.05 %	81,330,639	30.9	2.47	52	56.2 %	4,378,993,944
2019	**7,713,468,100**	1.08 %	82,377,060	29.8	2.51	52	55.7 %	4,299,438,618
2018	**7,631,091,040**	1.10 %	83,232,115	29.8	2.51	51	55.3 %	4,219,817,318
2017	**7,547,858,925**	1.12 %	83,836,876	29.8	2.51	51	54.9 %	4,140,188,594
2016	**7,464,022,049**	1.14 %	84,224,910	29.8	2.51	50	54.4 %	4,060,652,683
2015	**7,379,797,139**	1.19 %	84,594,707	30	2.52	50	54.0 %	3,981,497,663
2010	**6,956,823,603**	1.24 %	82,983,315	28	2.58	47	51.7 %	3,594,868,146
2005	**6,541,907,027**	1.26 %	79,682,641	27	2.65	44	49.2 %	3,215,905,863
2000	**6,143,493,823**	1.35 %	79,856,169	26	2.78	41	46.7 %	2,868,307,513
1995	**5,744,212,979**	1.52 %	83,396,384	25	3.01	39	44.8 %	2,575,505,235
1990	**5,327,231,061**	1.81 %	91,261,864	24	3.44	36	43.0 %	2,290,228,096
1985	**4,870,921,740**	1.79 %	82,583,645	23	3.59	33	41.2 %	2,007,939,063
1980	**4,458,003,514**	1.79 %	75,704,582	23	3.86	30	39.3 %	1,754,201,029
1975	**4,079,480,606**	1.97 %	75,808,712	22	4.47	27	37.7 %	1,538,624,994
1970	**3,700,437,046**	2.07 %	72,170,690	22	4.93	25	36.6 %	1,354,215,496
1965	**3,339,583,597**	1.93 %	60,926,770	22	5.02	22	N.A.	N.A.
1960	**3,034,949,748**	1.82 %	52,385,962	23	4.90	20	33.7 %	1,023,845,517

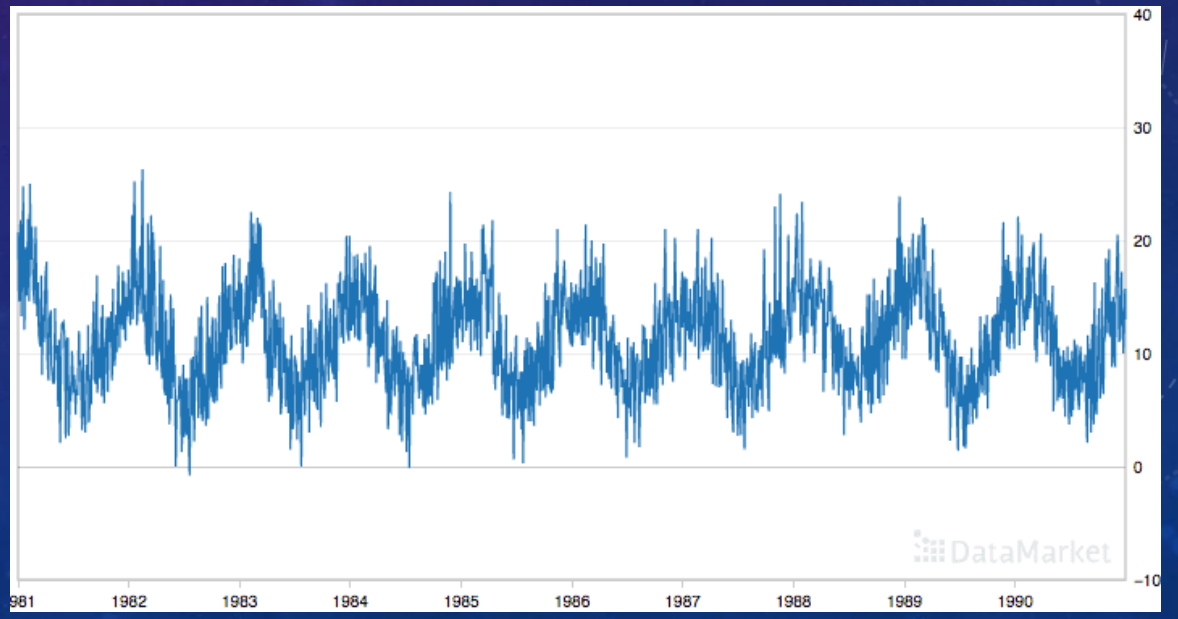
Tabula Data



Images

Third, in the Danish data we explore whether business manager effects are due to the selection of individuals who are more prone to take a hard line against labor into business degrees, or whether instruction and socialization in business degree programs produce managers who do not share rents with their employees. To answer this question, we exploit the presence of "role model" effects in the choice of college majors—whereby individuals are more likely to follow the choices of their role models (e.g., see Lockwood and Kunda, 1997; Lockwood, 2006 for general discussion, and Arcidiacono and Nicholson, 2005; Rask and Bailey, 2002; Poldin et al., 2015; Wiswall and Zafar, 2015). We collect the high school records of Danish managers and define the role models for each individual as students in the same high school and with similar academic performance, but one year ahead. We confirm that role models matter for college choice, with a 10% increase in the fraction of role models pursuing a business degree associated with a 1%–3% increase in a student's likelihood of enrolling in a business program. Exploiting this source of variation, we estimate that most of the wage and labor share results we present can be explained as the treatment effect of business education rather than a selection effect.

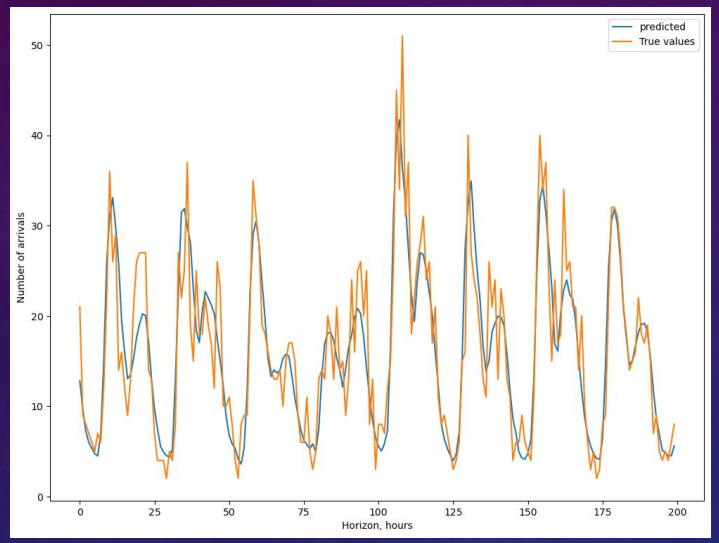
Text

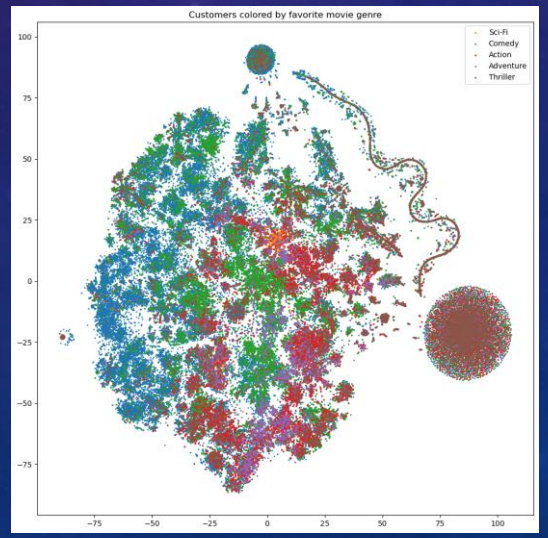


Time series

A SPECTRUM OF AI APPLICATIONS

- Customer churn prediction
- Fraud detection
- Spam email filtering
- Product recommendation
- Customer Segmentation
- Sentiment Analysis
- Document Labeling
- Topic Modelling
- Named Entity Recognition (NER)
- Time Series Forecasting
- Basic Machine Translation & Summarization





- Document Cleansing and Structuring
- Metadata generation
- Synthetic Data Generation
- Intelligent Chatbots
- Question Answering over custom documents (RAG)
- Function calling and AI agents:
  - Agents that use tools, memory, human-in-the-loop

WHY DO WE NEED CUSTOM AI MODELS AT THE AGE OF LLM?

- Specialist <-> Generalist
  - Specialized models will always be more “efficient” than general models:
    - For narrow tasks (classification, ranking) small models can beat LLMs
  - Each domain has its own jargon, private data that frequently updates,
- LLM’s are not well suited for a variety of tasks
  - Cannot comprehend a large corpus of documents
  - Time series analysis & tabular prediction, recommender systems, ranking, outlier detection
- LLM’s have huge hardware requirements and are expensive to run
  - This also poses privacy concerns & introduces dependencies
- LLM’s are difficult to control & evaluate on specific tasks (e.g. RAG evaluation is a challenge)

COURSE STRUCTURE

Module 0: Python

- Python fundamentals, debugging
- Setting up development environment

Module 1: Data Science and Classical Modelling

- Data collection and storage
- Data manipulation: NumPy, Pandas, SQL
- Data preprocessing, EDA, visualization
- Classical ML models: supervised & unsupervised learning
- Model evaluation and hands-on practice

Model 2: NLP & Deep Learning

- Text Processing: Lexical, topic modelling, word embeddinsg
- Use of word/doc embedding models for non-text tasks
- Deep Learning with pytorch: tensors, training loops, optimization
- Network Architectures: MLP, CNNs, LSTMs, Autoencoders

Module 3: Transformers and Large Language Models

- NLP and Deep plearning : From attention to transformers and LLMs.
- Hands-on with Hugging Face tools and pretrained models.
- Finetuning for information retrieval: embeddings, cross-encoders, bi-encoders
- Use of LLMs: data transformation, structured generation, function calling.

Module 4: Engineering LLM-Powered Applications

- Prompt engineering and vector search with FAISS/ChromaDB.
- RAG architecture
- Autonomous Agents
- Building real-world AI projects
- AWS tools for generative AI

AI ENGINEERING

Prof. Lev Muchnik
Hebrew University
Head of Data Science Dep.
School of Business

Ms. Maya Malamud
Senior Data Scientist
AI for Healthcare

Dr. Libby Kosolapov
Data Scientist

Liran Moshe
Data Scientist

Dr. Zvi Ben-Ami
Senior Data Scientist
Entrepreneur

Google drive folder:
https://drive.google.com/drive/folders/1Q6ZGsoj69Wp2yivNsaCTrtq6NDzMws6G?usp=sharing
Slides, Recordings, Code & Data, Assignments
Assignments – not for submission
TA's office hours:
WhatsApp group (administrative issues)

TERMINOLOGY

Artificial intelligence refers to the overarching idea of machines performing tasks in a manner we deem "intelligent."

Narrow (weak) AI: Focused specific tasks with predefined scope (recommender systems, image recognition, autonomous driving)

Characterized for being Task-specific and with predefined scope

General (strong) AI (AGI): A broadly accepted definition – AI can perform most of the intellectual tasks a human can.

Super AI/ Superintelligence (ASI) – surpasses human intelligence in all fields.

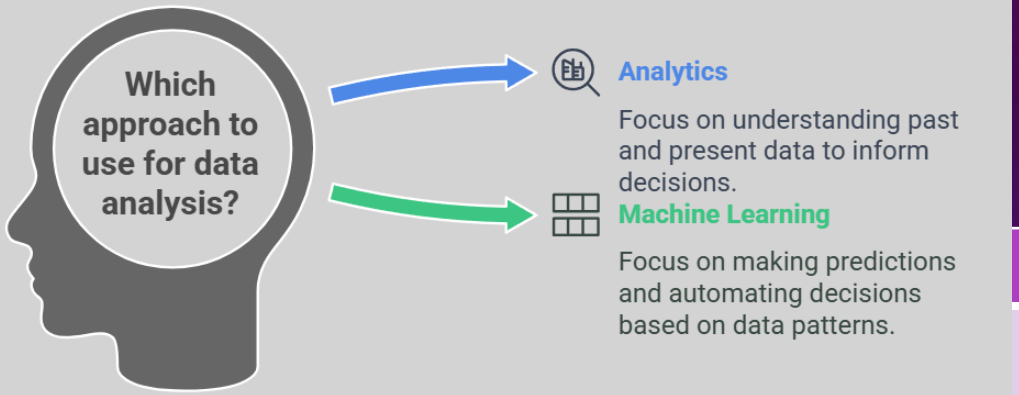
Machine learning represents the contemporary approach to creating Artificial Intelligence, focusing on the belief that machines can derive rules from data when provided with sufficient information.

TERMINOLOGY

Analytics versus Machine Learning: Descriptive Compared to Predictive/Prescriptive

Big Data is a term used to describe data analysis challenges that traditional tools cannot address.

Data Science is an academic field that examines data using scientific techniques to achieve practical insights.



Analytics versus Machine L

Big Data is a term used to c
address.

Data Science is an academi
achieve practical insights.

	Analytics	Machine Learning
Focus	Understanding past and present (descriptive)	Make prediction and automate decisions
Method	Statistics, visualization, human interpretation	Machine learning to find patterns in data
Outputs	Reports, dashboards, insights	Integration into systems. Predictions, classifications, automatic decision making
Applications	BI, KPI monitoring, performance analytics	Personalized recommendation, fraud detection, time series prediction
Key questions	"What happened?" "Why did it happen?"	"What will happen?" "What action should be taken?"
Advanced analytics could use ML		

TERMINOLOGY

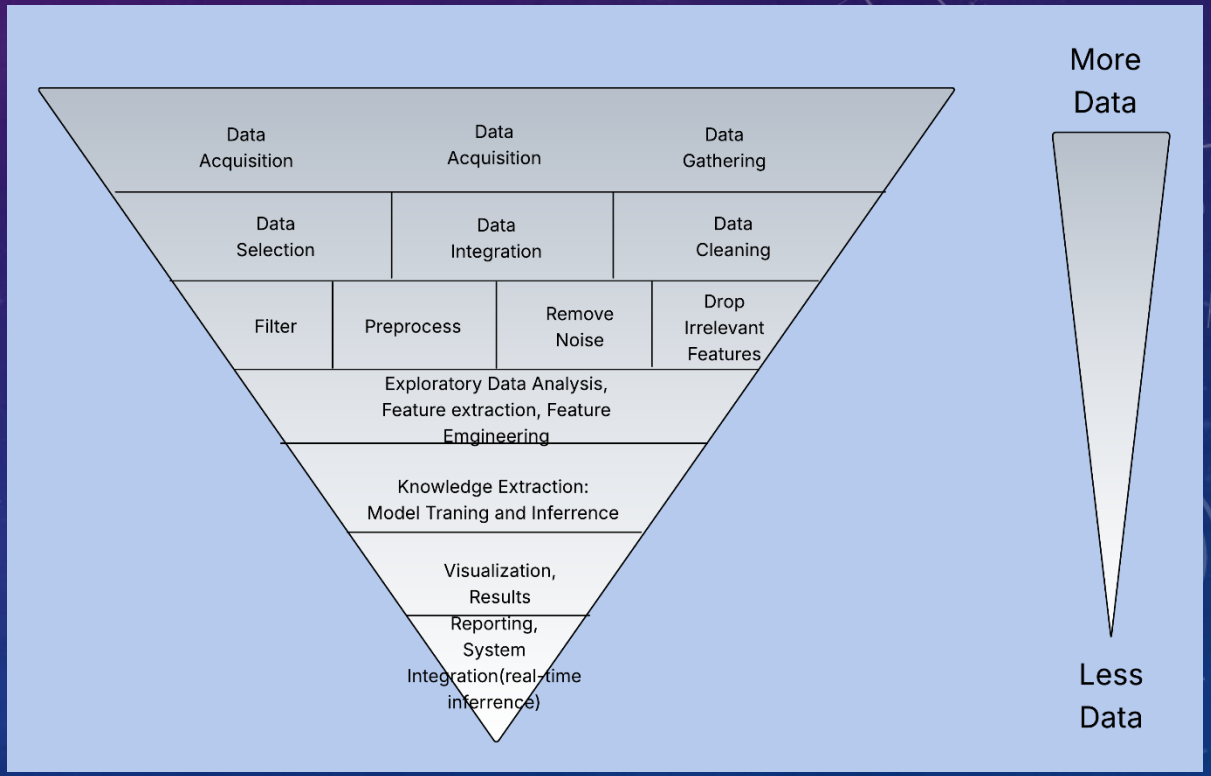
Analytics versus Machine Learning: Descriptive Compared to Predictive/Prescriptive

Big Data is a term used to describe data analysis challenges that traditional tools cannot address.

Data Science is an academic field that examines data using scientific techniques to achieve practical insights.

FUNNEL MODEL OF DATA PROCESSING: DATA REDUCTION PIPELINE

- Data-centric projects often start with **large raw datasets**
- As the project advances — through cleaning, filtering, modeling, and evaluation — the **data volume decreases**
- **Each stage** may require **different storage formats** and data representations



SOME DATA ACCESS AND MANAGEMENT CONSIDERATIONS

- Training & production may require *different approaches* to data management:
  - Training – lots of read-only access, going through the entire dataset.
  - Production - application of the trained model may require Compute & RAM, not disc access.
- Hardware:
  - HDD – OK streaming (especially RAID), horrible random access
  - SSD – Great for streaming & random access
  - Network – worst!
- Cloud storage – Google drive / one drive / dropbox:
  - Make sure to place your write folder outside their reach.
- Apache Spark, Hadoop map-reduce – distributed file management, access and computation.
  - Beneficial for very large datasets
  - Slow on single machine
- SQL Databases:
  - Analysis typically utilizes READ and (fewer) WRITE operations.
  - Typically, inefficient for iteration or massive data access: Optimized for concurrent access (locking, etc), extraction of a small set of records.

POPULAR DATA FILE FORMATS

- CSV
  - + Popular, Human Readable, Broadly supported, can be stored inside archives, supports appending data
  - - No data types, relatively SLOW (requires parsing), Messy for nested data
  - Notes:
    - Could be difficult to transfer: encodings (UTF-8?), custom delimiter (TAB – separated?), escape, NaN's, etc
    - Can be read via random access via custom index and file seek.
- XLSX
  - - Size-limited (1M rows), not necessarily a table (may contain other elements), Excell app locks the file.
- JSON / JSONL
  - + Popular, very flexible, supported nested data, supports appending data
  - - Verbos, Slow, No native support for vectors, some popular types are not supported
- **Pickle**
  - + Native for python, Stores binary objects, relatively fast, can store many variables in the same file.
  - - Fails to load if the stored class had changed. Not build for many variables. Proprietary python format, Slow for some complex data types (e.g. multiple strings) , Security issues (may execute arbitrary code on load).

POPULAR *BINARY* DATA FILE FORMATS

- **Parquet**: (*pip install pyarrow* or *pip install fastparquet*)
  - + Columnar orientation, Fast for columnar access, supports compression
  - - Not huma-readable, inefficient for row-wise access, data cannot be appended
- **Feather**: (*pip install pyarrow)*
  - + Very fast read/write for tables
  - - no appending
- **HDF5**: (*pip install pytables*)
  - + Supports hierarchical structure, allows appending data
  - - Complex, not suited for columnar analytics
- **SQL:**
  - + Popular, Powerful querying, supports indexing and random access, can perform some analysis (slow).
  - - Slow, requires setup, additional language, more complex backup (except for liteSQL)
- **liteSQL:**
  - + Lightweight, portable, querying and schema
  - - no concurrent writes, single-threaded: no concurrent reads & won't scale well for large databases
- **LMDB**: (pip install lmdb)
  - Very fast on-disk key-value store.

READING TEXT FILES LINE BY LINE

- In many cases, the JSON file would be too large to fit into memory
- Also, we might not need all fields or all lines of that file
- In those cases, one could read the file line-by line
  - json.loads()

WorkingWithDataFiles.ipynb

Read Json line-by-line

```
In [3]: import os
        import json

        file_name = os.path.join('local_data' , 'airline_details.json')

        airline_details = []

        with open(file_name,'rb') as in_file:
            for line in in_file:
                current_data = json.loads(line)
                airline_details.append(current_data)
```

executed in 1.17s, finished 11:48:47 2021-03-19

airline_details.json

C:\Users\zvibe\Documents\My_Study\HUJI\Teaching\Data Science\Data Science\Week_02\local_data\airline_details.json - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

airline_details.json


```
1 {"airport": {"code": "ATL", "name": "Atlanta, GA: Hartsfield-Jackson Atlanta International"}, "statistics": {"flig
2 {"airport": {"code": "BOS", "name": "Boston, MA: Logan International"}, "statistics": {"flights": {"cancelled": 7,
3 {"airport": {"code": "BWI", "name": "Baltimore, MD: Baltimore/Washington International Thurgood Marshall"}, "stati
4 {"airport": {"code": "CLT", "name": "Charlotte, NC: Charlotte Douglas International"}, "statistics": {"flights": {
5 {"airport": {"code": "DCA", "name": "Washington, DC: Ronald Reagan Washington National"}, "statistics": {"flights"
6 {"airport": {"code": "DEN", "name": "Denver, CO: Denver International"}, "statistics": {"flights": {"cancelled": 1
```

OBJECT SERIALIZATION IN PYTHON: PICKLE

Serialization – storing structured Python data in files and loading it back into Python.

- **Serialization**: pickle.dump(*obj*, *file*[, *protocol*])
  - obj – object to pickle
  - file – file object (open with write permission: *file = open(file_name, 'w+b')*)
  - protocol – (optional) Which protocol to use for pickling. Default: 0 (ASCII)

- **Deserialization**: pickle.load(*file*)
  - file – file object (open with read permission: *file = open(file_name, 'rb')*)

```
with open(filename,'wb') as fp:
    pickle.dump(dict1,fp)
    pickle.dump(dict2,fp)
    pickle.dump(dict3,fp)

with open(filename,'rb') as fp:
    d1=pickle.load(fp)
    d2=pickle.load(fp)
    d3=pickle.load(fp)
```

Pickle supports several protocols.
Use protocol 0 for backward compatibility (default)
Use protocol -1 (or any negative) for the newest protocol
Notes:

Pickle is used by python to pass data between processes.
Security risk: Pickle can be made to execute arbitrary code during deserialization

PICKLE

Advantages

- Natural to python (on all platforms!): *import pickle*
- Supports compression: *pickle.dump(object, out_file, protocol=-1)*
- Binary and (typically) quite fast
- Platform independent: pickle files can be moved from one machine to another
- Can store multiple objects in the same file.

Disadvantages

- Natural **only** to python
- Binary: Not human-readable

WorkingWithDataFiles.ipynb

Pickle File

In [1]:
```
import pickle

list_data = ['a', 'b', 'c', 'a']
dict_data = {'a': 'element a', 'b': 'element b', 'all': list_data}

# save data strcutures to file
with open('serialized.pickle','w+b') as out_file:
    pickle.dump(list_data, out_file, protocol=-1)
    pickle.dump(dict_data, out_file, protocol=-1)

# load data from file:
with open('serialized.pickle','rb') as in_file:
    list_data = pickle.load(in_file)
    dict_data = pickle.load(in_file)

#print loaded data
print(list_data)
print(dict_data)
```

executed in 27ms, finished 11:53:31 2021-03-19

```
['a', 'b', 'c', 'a']
{'a': 'element a', 'b': 'element b', 'all': ['a', 'b', 'c', 'a']}
```

Pickle alternatives:

Text-base: CSV / JSON / XML

Binary: Feather / Parquet / HDF5

Local database engine: sqlite3

OPTIMIZING WORK WITH PICKLE

WorkingWithDataFiles.ipynb

Read Pickle if not exist

In [2]:
```
import os
import pickle

pickle_file_name =  'dump.pickle'

if not os.path.isfile(pickle_file_name):
    # build the data & store it in the pickle
    data_list = [1,2,3,4,5]
    data_dict = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5}
    data_set = set([1,2,1,3,1,5,3,2,])
    with open(pickle_file_name, 'wb') as out_file:
        pickle.dump(data_list,out_file, protocol=-1)
        pickle.dump(data_dict,out_file, protocol=-1)
        pickle.dump(data_set,out_file, protocol=-1)
else:    # load data:
    with open(pickle_file_name, 'rb') as in_file:
        data_list = pickle.load(in_file)
        data_dict = pickle.load(in_file)
        data_set = pickle.load(in_file)
```

executed in 15ms, finished 12:40:37 2021-03-19

In most cases, loading data from pickle is significantly faster than rebuilding data containers from scratch.

READING DIRECTLY FROM ARCHIVES

WorkingWithDataFiles.ipynb
WorkingWithHugeFiles.ipynb

List files within archive

```
1  with zipfile.ZipFile(source_file_name, 'r') as z:
2      # List all files in the zip
3      file_list = z.namelist()
4      print(f'Files in zip: {file_list}')
5
```

[4] ✓ 0.0s

```
Files in zip: ['amazon_review_polarity_csv.tgz', 'test.csv', 'train.csv']
```

Read large file line by line

```
1  with zipfile.ZipFile(source_file_name, 'r') as z:
2      with z.open('train.csv', 'r') as f:
3          for row_id, row in tqdm(enumerate(f)):
4              pass
```

[5] ✓ 16.4s

```
3600000/? [00:16<00:00, 219991.77it/s]
```

RANDOM ACCESS TO LARGE TEXT FILES

text_file_index.ipynb – part A

Index File: e.g. index.csv

```
title,start_pos
ויקיפדיה,0
מתמטיקה,87
תוכן שיתופי,56515
אקסיומה,56608
פיזיקה,63945
Main Page,85807
ויקי,85886
מוזיקה,95632
ג'אז,126606
פיוז'ן (מוזיקה),147566
ספורט אתגרי,149625
טיפוס קירות,153626
ספורטאי,166019
לינוקס,170874
מערכת הפעלה,215434
קוד פתוח,233475
KDE,246069
החוק השלישי של ניוטון,258033
איזק ניוטון,258294
זיכרון (פירושונים),336353
מעבד,337056
כבידה,366418
```

Data File, e.g. *hewiki_articles.jsonl*

{"title": "צג", "summary": "__250|ללא_תוכן_עניינים__ שמאל|ממוזער|250pxדגם; צג מחשב מודרני; דק ורחב LCD 19 אינץ', בגודל \n...
{"title": "Integrated Drive Electronics", "summary": "", "text": "הפניה Parallel ATA"}
{"title": "SCSI", "summary": "ממוזער|מחבר סקאזי\nSCSI, ראשי תיבות - Small Computer System Interface (עוד יתרון ל...
{"title": "כרטיס טלוויזיה", "summary": "250|שמאל|ממוזער|250pxכרטיס טלוויזיה בחיבור PCI ...
{"title": "צורב", "summary": "250|שמאל|ממוזער|250pxצורב CDRW ו DVD\n...
{"title": "אלברט איינשטיין", "summary": "250|ממוזער|250pxפסל של אלברט איינשטיין באוניברסיטה העברית בגבעת רם. הפסל הוא גאורגי פראנגוליאן (2015)...

For compressed files use ***indexed_gzip*** (slow) or **indexed_bzip2** (very slow)

RANDOM ACCESS VIA MEMORY MAPPED FILE (MMAP)

- Memory mapped file - Optimized for random file access
- Both *standard file access* (f.read(), f.seak(), …) and *mmap* cache file access
- However:
  - MMAP cache is in user app address access (i.e. mmap doesn't necessarily issue **system call**).
  - MMAP reads data directly into a user space (vs file -> kernel -> user space)

text_file_index.ipynb – part B

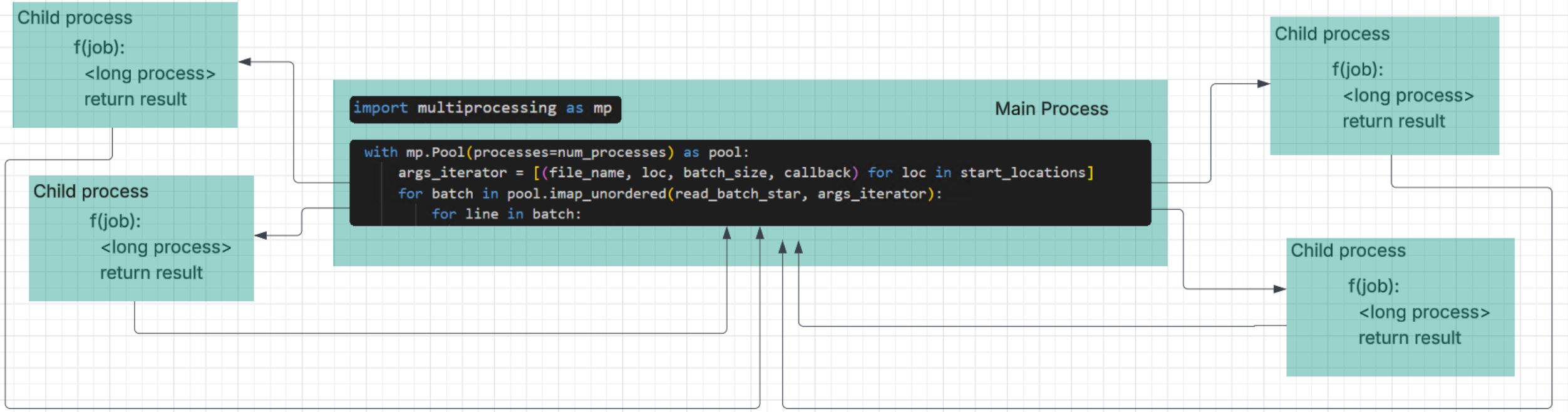
```
with open(source_file_name, 'r', encoding='utf-8') as source_file:
    with mmap.mmap(source_file.fileno(), 0, access=mmap.ACCESS_READ) as mm:
        for title in tqdm(sample_titles, desc="Loading Sample Data", unit="article"):
            start_pos = file_index[title]
            mm.seek(start_pos)
            # Read the line containing the article
            (variable) line_data: Any
            line_data = json.loads(line)
```

CONCURRENT READING FROM TEXT FILES: SPLIT AND CONQUER

- Use python multiprocessing to separate data preparation (read and preliminary analysis) from further use:
  - Read and preprocess data in different (concurrent) processes
  - Pass partially or fully processed data to the main process
  - In the main process, collect processed data.
- Each process may (for example)
  - receive a name of a file to read & process
  - receive a range of locations from the same file.

CONCURRENT READING FROM TEXT FILES: SPLIT AND CONQUER

- Use python multiprocessing to separate data preparation (read and preliminary analysis) from further use:
  - Read and preprocess data in different (concurrent) processes



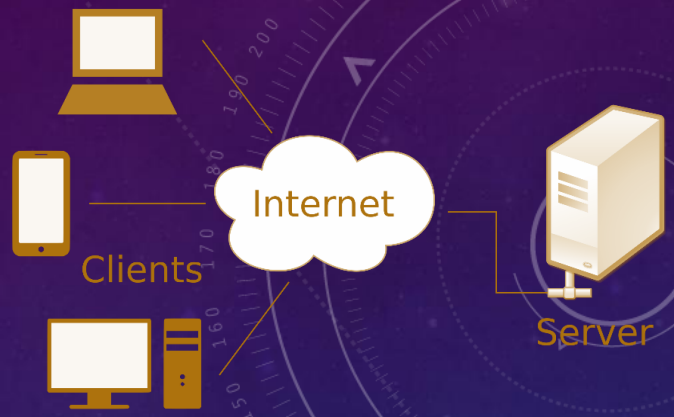
CONCURRENT READING FROM TEXT FILES: SPLIT AND CONQUER

- Use python multiprocessing to separate data preparation (read and preliminary analysis) from further use:
  - Read and preprocess data in different (concurrent) processes
  - Pass partially or fully processed data to the main process
  - In the main process, collect processed data.
- Each process may (for example)
  - receive a name of a file to read & process
  - receive a range of locations from the same file.

concurrent_read_word_count.py

concurrent_read.py

CLIENT-SERVER INTERACTION



Python application can interact with web server by requesting and posting information

It can even pretend to be a web browser.

1. The client (your python app) sends request to the web server
   - The request may contain: URL and parameters that specify your needs.
   - Your program will generate these parameters
2. The web server will call the required function and process the parameters
3. Web server generates response and sends it back to the client
4. The client receives the response and processes it

Web server response can be:

- Text file
- HTML
- JSON file
- XML
- Binary file (e.g. image)
- Anything else

HTTP REQUESTS IN PYTHON

WorkingWithWebData_partA.ipynb

HTTP requests

```
In [1]: import requests
        response = requests.get('http://bschool.huji.ac.il')
```

executed in 666ms, finished 19:44:12 2021-03-20

HEADERS:

```
In [4]: for k,v in response.headers.items():
            print ('{0:s} :{1:s}'.format(k,v))
```

executed in 11ms, finished 19:44:13 2021-03-20

```
Date :Sat, 20 Mar 2021 17:44:15 GMT
Server :Apache/2.2.34 (Linux/SUSE)
Set-Cookie :PHPSESSID=rqchilajakpgb6ph0tjlpnpr8qoigar9; path=/; HttpOnly
Expires :Mon, 18 Apr 2010 12:00:00 GMT
Cache-Control :no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Last-Modified :Sat, 20 Mar 2021 17:44:15 GMT
Pragma :no-cache
Keep-Alive :timeout=15, max=100
Connection :Keep-Alive
Transfer-Encoding :chunked
Content-Type :text/html; charset=utf-8
```

CONTENT:

```
In [3]: print(response.content)
```

executed in 35ms, finished 19:44:13 2021-03-20

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>בית הספר למנהל העסקים של האוניברסיטה העברית</title>
<meta name="Keywords" content="MBA, תואר שני במנהל עסקים, תכניות למנהלים, תואר ראשון בחשבונאות, תואר ראשון במנהל עסקים " />
<meta name="Description" content="בית הספר למנהל העסקים של האוניברסיטה העברית" />
<!--[if IE]>
    <link href="http://bschool.huji.ac.il/projects/hujibusiness/app/views/he/styles/ie.css" rel="stylesheet" type="text/c
<![endif]-->
<link href="http://bschool.huji.ac.il/projects/hujibusiness/app/views/he/styles/style.css" rel="stylesheet" type="text/cs
<link href="http://bschool.huji.ac.il/projects/hujibusiness/app/views/he/styles/av_hack.css" rel="stylesheet" type="text/
<link href="http://bschool.huji.ac.il/projects/hujibusiness/app/views/he/styles/menu.css" rel="stylesheet" type="text/css
<link href="http://bschool.huji.ac.il/projects/hujibusiness/app/views/he/styles/style_editor.css" rel="stylesheet" type="

<script type="text/javascript" src="http://bschool.huji.ac.il/.app/helpers/js/jquery-1.4.4.min.js"></script>
<script type="text/javascript" src="http://bschool.huji.ac.il/.app/helpers/js/jquery.watermark.min.js"></script>
<script type="text/javascript" src="http://bschool.huji.ac.il/projects/hujibusiness/app/helpers/js/jquery.superfish.js"><
<script type="text/javascript" src="http://bschool.huji.ac.il/projects/hujibusiness/app/helpers/js/adjustsubs_he.js"></sc
<script type="text/javascript" src="http://bschool.huji.ac.il/.app/helpers/js/global_scripts.js"></script>
<script type="text/javascript" src="http://bschool.huji.ac.il/projects/hujibusiness/app/helpers/js/scripts.js"></script>
<script type="text/javascript" src="http://bschool.huji.ac.il/.app/helpers/js/jquery.cycle.min.js"></script>
```

HTTP REQUESTS IN PYTHON

Other response fields:

- *content* – binary content
- text – content of the response in Unicode (*content* field is in bytes)
  - *text* is equivalent to *content.decode(`utf-8`)*
- json() – converts content into a json object
- status_code – OK (200) /error details (e.g. 404)
- *cookies*
- elapsed – time it took to acquire response
- encoding – e.g. 'utf-8'
- url – the actual url generated (with parameters)
- raw – file object (use it as regular file created with 'open'; call get with *streaming=True*)

WorkingWithWebData_partA.ipynb

HTTP requests

```
In [2]: print (response)
        executed in 16ms, finished 21:00:14 2021-03-20
        <Response [200]>
```

```
In [5]: response.content.decode('utf-8')==response.text
        executed in 32ms, finished 19:44:14 2021-03-20
Out[5]: True
```

BEAUTIFUL SOUP

- Beautiful Soup is a Python library for pulling data out of HTML and XML files
- https://www.crummy.com/software/BeautifulSoup/bs4/doc/

WorkingWithWebData_partA.ipynb

Beautiful Soup

```
In [5]: from bs4 import BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')
```
executed in 272ms, finished 21:00:14 2021-03-20

```
In [6]: print(soup)
```
executed in 32ms, finished 21:00:14 2021-03-20

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<title>בית הספר למנהל העסקים של האוניברסיטה העברית</title>
<meta content="תואר שני במנהל עסקים MBA, תכניות למנהלים ,תואר ראשון בחשבונאות ,תואר ראשון במנהל עסקים " name="Keywords">
<meta content="בית הספר למנהל העסקים של האוניברסיטה העברית" name="Description"/>
```

```
In [7]: print(soup.prettify())
```
executed in 32ms, finished 21:00:14 2021-03-20

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
  <title>
   בית הספר למנהל העסקים של האוניברסיטה העברית
  </title>
  <meta content="תואר שני במנהל עסקים MBA, תכניות למנהלים ,תואר ראשון בחשבונאות ,תואר ראשון במנהל עסקים " name="Keywords">
   <meta content="בית הספר למנהל העסקים של האוניברסיטה העברית" name="Description"/>
   <!--[if IE]>
        <link href="http://bschool.huji.ac.il/projects/hujibusiness/app/views/he/styles/ie.css" rel="stylesheet" type
ss" />
```

```
In [8]: print(soup.title)
```
executed in 16ms, finished 21:00:14 2021-03-20

```
<title>בית הספר למנהל העסקים של האוניברסיטה העברית</title>
```

```
In [9]: print(soup.get_text())
```
executed in 16ms, finished 21:00:14 2021-03-20

```
בית הספר למנהל העסקים של האוניברסיטה העברית
```

```
In [10]: print(soup.h1)
         print(soup.h2)
```
executed in 15ms, finished 21:00:15 2021-03-20

```
None
<h2>24/2/2021 :הקלטת היום הפתוח לתלמידי תואר ראשון מתאריך</h2>
```

HTTP REQUESTS IN PYTHON: PARAMETERS

```
In [6]: response = requests.get('https://stooq.com/q/?s=aapl.us')
```

executed in 1.61s, finished 20:21:50 2021-03-20

WorkingWithWebData_partA.ipynb

Requests - Stocks Data

https://stooq.com/q/?s=aapl.us

Related: WORK.US -1.16% | CL.F +4.99% | NFLX.US -1.09% | MSFT.US -2.59% | TSLA.US -10.30%

<< ZOOM VIDEO COMMUNICATIONS (NASDAQ: ZM.US) | Ticker Rank: 394 (-30) | ★ 86

Last $ 138.5600	Date 2020-05-01 22:00:01
Change +3.3900 (+2.51%)	High/Low 141.6300 132.6700
52W Change +65.8000 (+90.43%)	52W High/Low 181.5000 60.9675
Open 136.0000	Prev. 135.1700
Bid 138.7000x1.8k	Ask 139.0200x1.9k
Volume 13.8m	Turnover 1.47g
OpenInt	No. Trades 53 747

Rates of Return: 1 Year: **+90.43%**, YTD: +103.64%

Zmienność? Wykorzystaj sytuację na wzrost spadek

Range: 1d | 3d | 5d | 10d | 1m | 3m | 5m | **1y** | 2y | 3y | 5y | 10y | 20y | 30y | 50y | 100y | Max (from 2019)

Type: Candle | Bar | **Line** | P&F H/L % x 3 OK Scale: Linear | **Log** Size: **M** | L

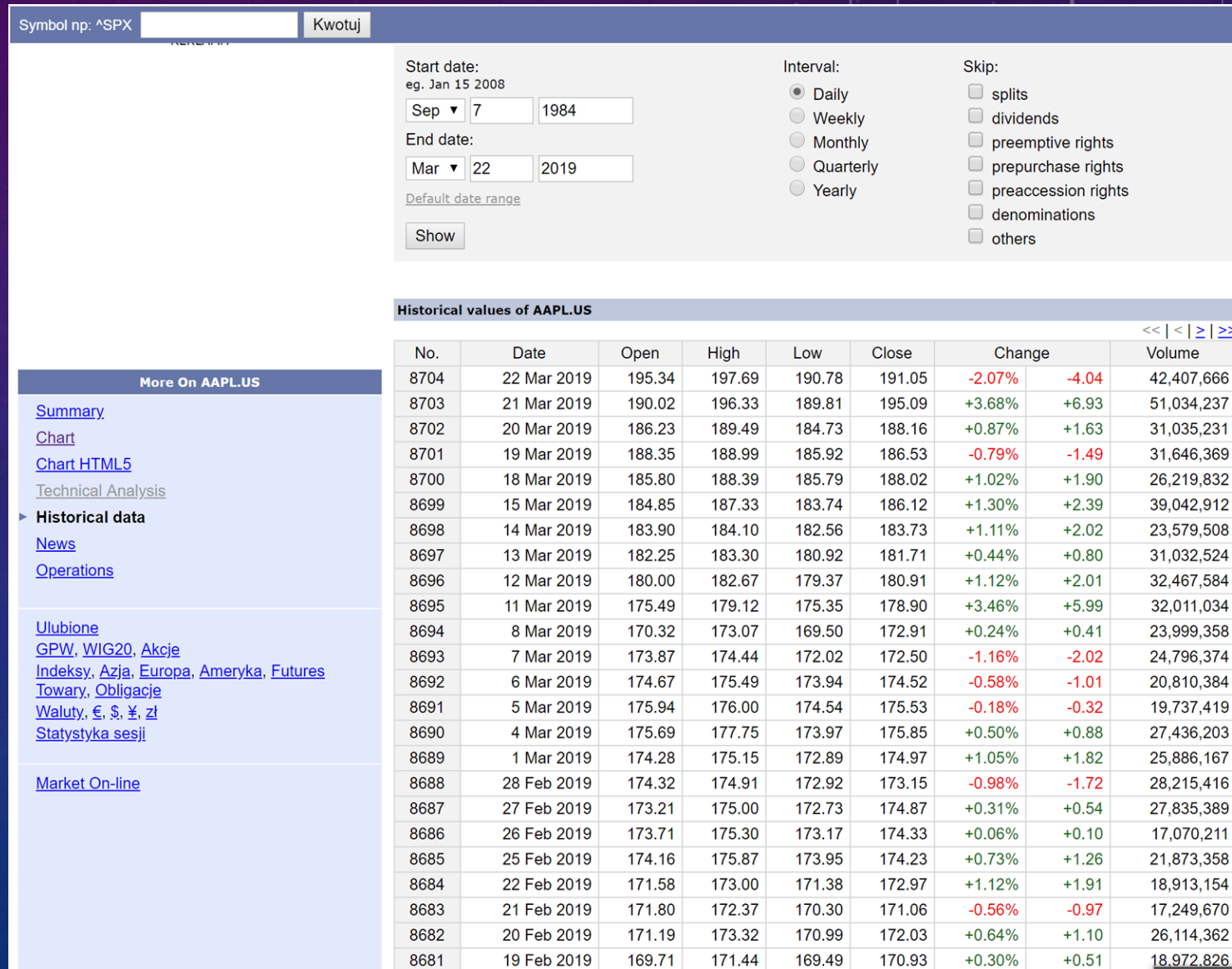
Date: **Last Session** | 2020-05-01 | 2020-04-30 | 2020-04-29 | 2020-04-28 | 2020-04-27

Relative Strength Chart: ZM.US to OK (beta version)

Compare with: WIG20 OK
(up to 8 symbols separated by a space)

HTTP REQUESTS IN PYTHON: PARAMETERS

- Historical data in table format
- Note the URL



https://stooq.com/q/d/?s=aapl.us

HTTP REQUESTS IN PYTHON

- Playing with parameters, we find out that dropping everything but the ticker (AAPL), we can request all available data in CSV format:

- **https://stooq.com/q/d/l/?s=aapl.us**

```
In [8]: URL = 'https://stooq.com/q/d/l/?s=aapl.us'
        response = requests.get(URL)
```

executed in 1.54s, finished 20:16:22 2021-03-20

```
In [9]: print(response.content)
```

executed in 85ms, finished 20:16:22 2021-03-20

```
b'Date,Open,High,Low,Close,Volume\r\n1984-09-07,0.10182,0.10306,
7,0.10122,75025205\r\n1984-09-11,0.10213,0.1049,0.10213,0.10306,
```

```
In [10]: print(response.text)
```

executed in 33ms, finished 20:16:22 2021-03-20

```
Date,Open,High,Low,Close,Volume
1984-09-07,0.10182,0.10306,0.1006,0.10182,96661645
1984-09-10,0.10182,0.10213,0.09937,0.10122,75025205
1984-09-11,0.10213,0.1049,0.10213,0.10306,176913886
```

	A	B	C	D	E	F
1	Date	Open	High	Low	Close	Volume
2	9/7/1984	0.42388	0.42902	0.41874	0.42388	23220030
3	9/10/1984	0.42388	0.42516	0.41366	0.42134	18022532
4	9/11/1984	0.42516	0.43668	0.42516	0.42902	42498199
5	9/12/1984	0.42902	0.43157	0.41618	0.41618	37125801
6	9/13/1984	0.43927	0.44052	0.43927	0.43927	57822062
7	9/14/1984	0.44052	0.45589	0.44052	0.44566	68847968
8	9/17/1984	0.45718	0.46357	0.45718	0.45718	53755262
9	9/18/1984	0.45718	0.46103	0.44052	0.44052	27136886
10	9/19/1984	0.44052	0.44566	0.43157	0.43157	29641922
11	9/20/1984	0.43286	0.43668	0.43286	0.43286	18453585
12	9/21/1984	0.43286	0.44566	0.42388	0.42902	27842780
13	9/24/1984	0.42902	0.43157	0.42516	0.42516	22033109
14	9/25/1984	0.42388	0.42388	0.41618	0.41618	46515020
15	9/26/1984	0.41618	0.4354	0.41111	0.41111	30947546
16	9/27/1984	0.41111	0.41366	0.41111	0.41111	29541971
17	9/28/1984	0.41111	0.41111	0.39316	0.40081	65093531
18	10/1/1984	0.39956	0.39956	0.39186	0.39186	27268068
19	10/2/1984	0.39443	0.40853	0.39443	0.39443	32977801
20	10/3/1984	0.40081	0.40724	0.40081	0.40081	33583772
21	10/4/1984	0.40593	0.40853	0.40593	0.40593	34995586
22	10/5/1984	0.40593	0.40593	0.39443	0.39699	27211851
23	10/8/1984	0.39699	0.39956	0.39699	0.39699	13099922
24	10/9/1984	0.39699	0.39956	0.39316	0.39316	34933112
25	10/10/1984	0.39316	0.39316	0.38164	0.38164	1.02E+08

WorkingWithWebData_partA.ipynb

HTTP REQUESTS IN PYTHON: PARAMETERS

```
In [11]: response = requests.get('https://stooq.com/q/d/l',params={'s':'AAPL.US'})
         print(response.url)
         print(response.headers)
```

executed in 2.35s, finished 20:24:57 2021-03-20

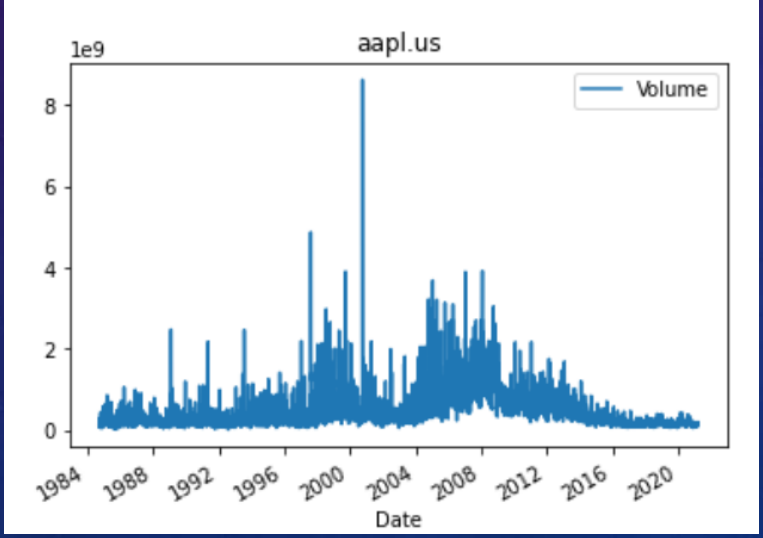
```
https://stooq.com/q/d/l/?s=AAPL.US
{'Date': 'Sat, 20 Mar 2021 18:24:57 GMT', 'Server': 'Apache', 'Content-disposition': 'attachment;filename=aapl_us_d.csv', 'Keep
-Alive': 'timeout=3, max=119', 'Connection': 'Keep-Alive', 'Transfer-Encoding': 'chunked', 'Content-Type': 'text/plain'}
```

WorkingWithWebData_partA.ipynb

Requests - Stocks Data

HTTP REQUESTS IN PYTHON: PARAMETERS

- Download data for several stocks
- Create CSV file with record of which stocks were downloaded
- Compute daily trading volume for all the stocks in the list
- Plot timeline of daily trading volume for each of the stocks



WorkingWithWebData_partA.ipynb

Requests - Stocks Data

```
Downloaded  1 of 4. aapl.us 401.902344 KB
Downloaded  2 of 4. ibm.us 651.128906 KB
Downloaded  3 of 4. googl.us 152.522461 KB
Downloaded  4 of 4. msft.us 383.884766 KB
finished
```

DOWNLOADING IMAGES

WorkingWithWebData_partA.ipynb

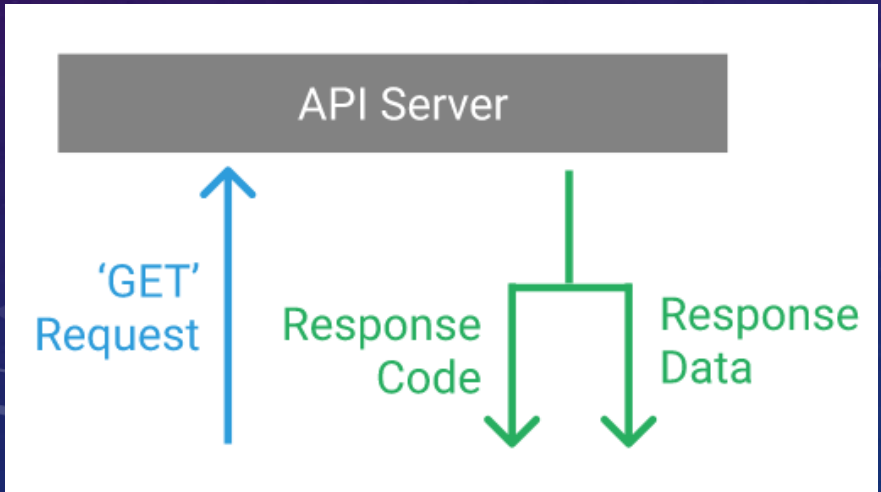
Get Image from the web

In [1]:
```
import shutil
import requests
from PIL import Image

target_file_name = r'local_data/Lena.png'

url = 'https://upload.wikimedia.org/wikipedia/commons/thumb/f/f2/Jim-Carrey-2008.jpg/330px-Jim-Carrey-2008.jpg'
with  requests.get(url, stream=True) as response:
    with open(target_file_name, 'wb') as out_file:
        shutil.copyfileobj(response.raw, out_file)
img = Image.open(target_file_name)
img.show()
```

executed in 2.85s, finished 14:18:20 2021-03-19

WEB API

- API – Application Program Interface
- API is used to:
  - Retrieve data from remote web sites
  - Upload data
  - Execute remote code
- When we want to receive data from an API, we need to make a **request**.



Source: https://www.dataquest.io/blog/python-api-tutorial/

Advantages:

- Not everything is available via web pages. API may provide functionality not used by humans
- Consistent interface –web pages' format may change
- Efficient
  - Web server doesn't need to generate web pages
  - Client doesn't need to parse HTML
  - Data is well structured (JSON/XML)
- Regulated (e.g. Authentication keys)
- Well documented (sometimes)

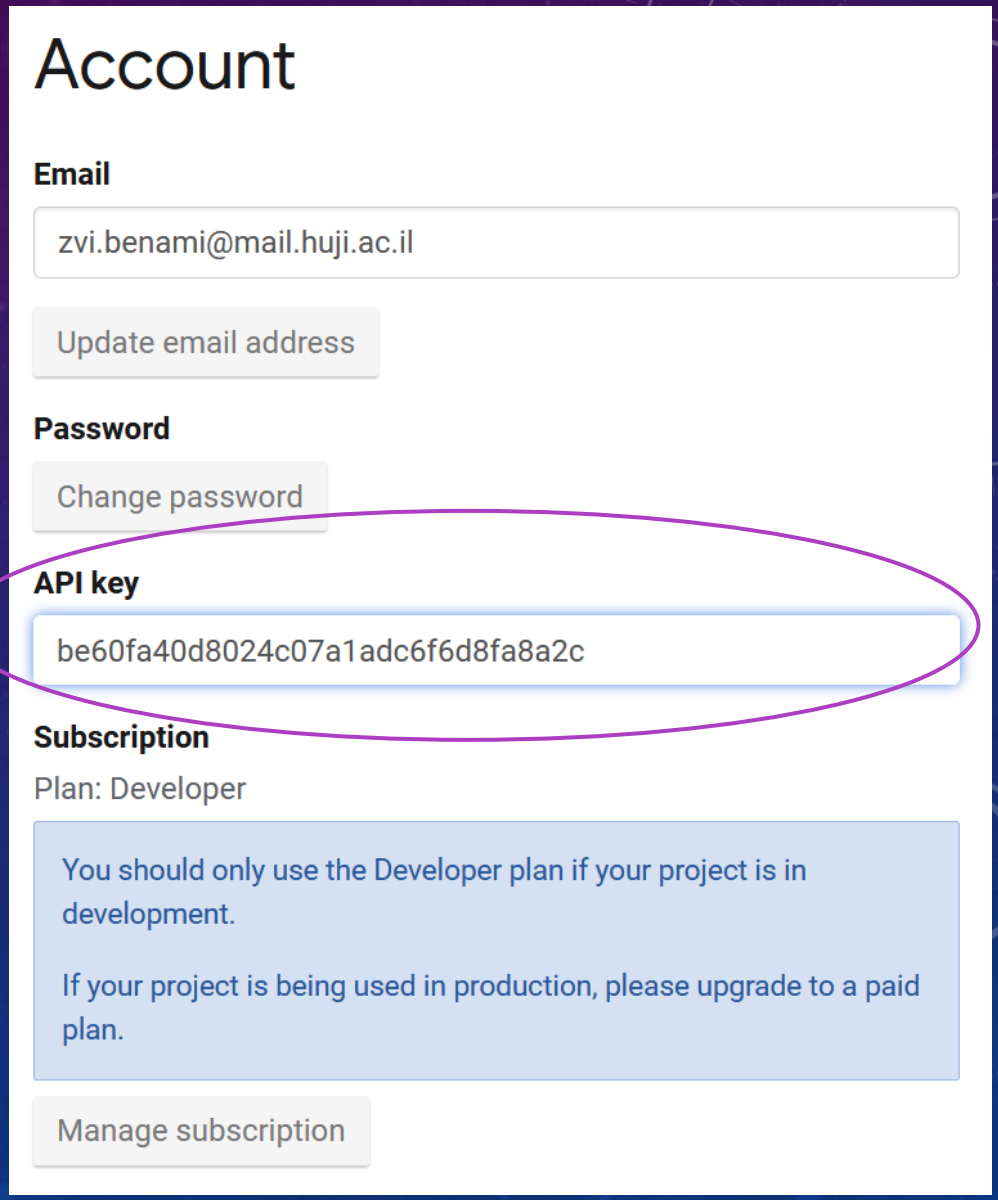
WEB API

- To use Web API of a specific web site one needs to know:
  - Endpoint – server URL
  - Authentication Key (not always)
  - Interface – functions, their parameters and what they return
- Next, you have to develop the program logic

GETTING THE API KEY

- http://newsapi.org/

- The key must be sent with every access to the API
- These calls are counted (and you could be charged)
- Never share the key!!!! (like I do here)

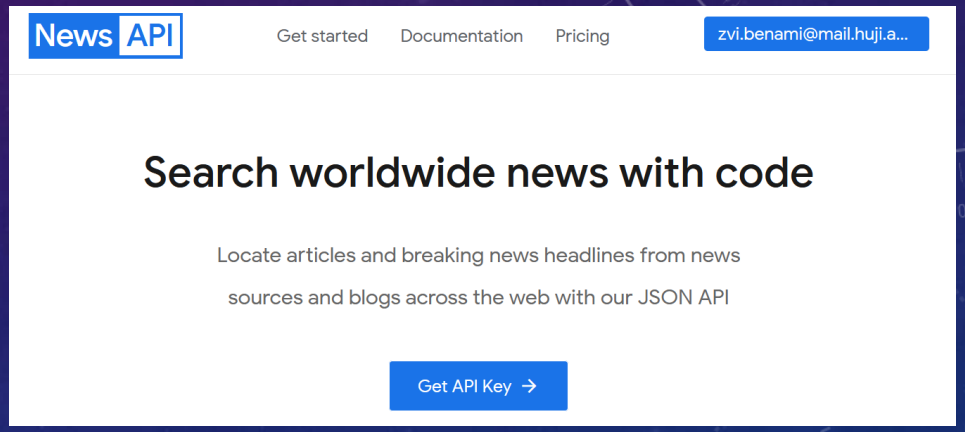


http://newsapi.org/

EXAMPLE: COLLECT NEWS ON SOME TOPIC

WorkingWithWebData_partA.ipynb

Web API - News API



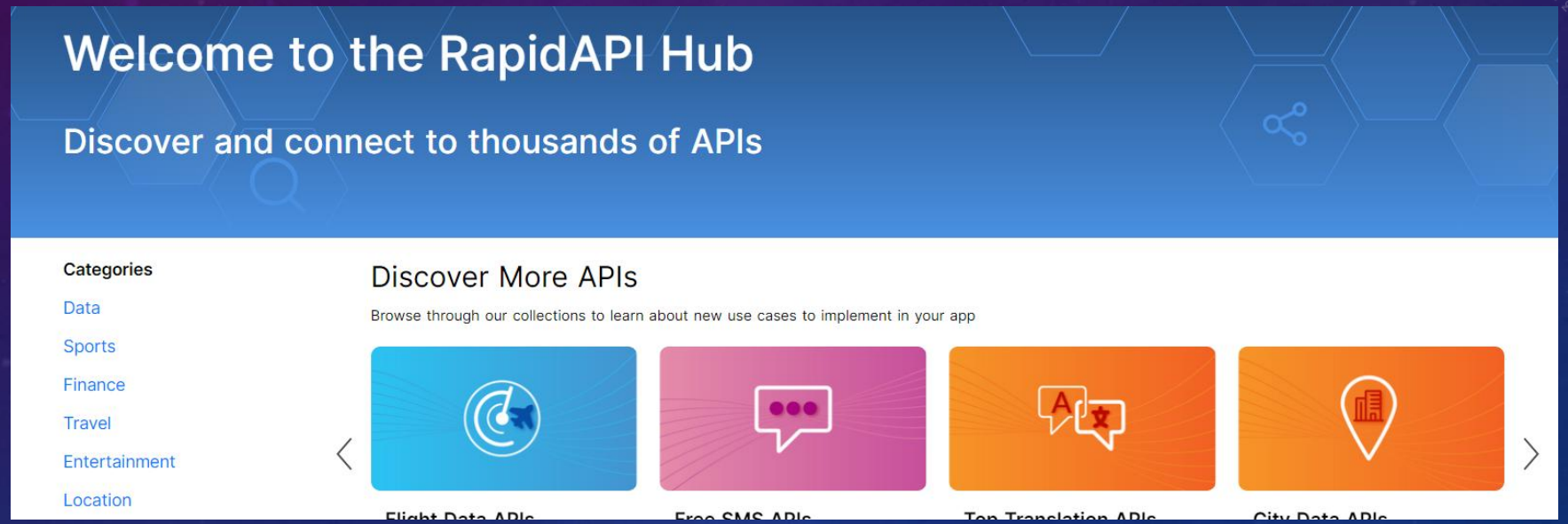
- Option 1: scrap news web sites
- Option 2: subscribe to some news api (e.g. newsapi.org)
  - Endpoint: https://newsapi.org/
  - Methods:
    - everything , eg: http://newsapi.org/v2/everything
    - top-headlines, eg: http://newsapi.org/v2/top-headlines
  - Learn parameters:
    - https://newsapi.org/docs/endpoints/everything
    - Category: http://newsapi.org/v2/top-headlines?http://newsapi.org/v2/everything&category=business&apiKey=be60fa40d8024c07a1adc6f6d8fa8a2c
    - Search: http://newsapi.org/v2/top-headlines?q=biden&apiKey=be60fa40d8024c07a1adc6f6d8fa8a2c
    - Dates: http://newsapi.org/v2/top-headlines?q=biden&from=2021-04-07&to=2021-04-15&apiKey=be60fa40d8024c07a1adc6f6d8fa8a2c
    - Current headlines at TechCrunch:
      - http://newsapi.org/v2/top-headlines?**sources=techcrunch**&apiKey=be60fa40d8024c07a1adc6f6d8fa8a2c
    - Search specific domain:
      - http://newsapi.org/v2/everything?**domains=wsj.com**&apiKey=be60fa40d8024c07a1adc6f6d8fa8a2c

RAPID API

- Rappid API: https://rapidapi.com/
- Rapid Hub: https://rapidapi.com/products/hub/

WorkingWithWebData_partA.ipynb

Rapid API - News API



- Sample enpoint: https://rapidapi.com/SmartableAI/api/coronavirus-smartable/

WEATHER API: HTTPS://OPENWEATHERMAP.ORG/

Current weather and forecasts collection

Free	Startup	Developer	Professional	Enterprise
Get API key	30 GBP/ month Subscribe	140 GBP/ month Subscribe	370 GBP/ month Subscribe	1500 GBP/ month Subscribe
60 calls/minute 1,000,000 calls/month	600 calls/minute 10,000,000 calls/month	3,000 calls/minute 100,000,000 calls/month	30,000 calls/minute 1,000,000,000 calls/month	200,000 calls/minute 5,000,000,000 calls/month
Current Weather	Current Weather	Current Weather	Current Weather	Current Weather
3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days	3-hour Forecast 5 days
Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days	Hourly Forecast 4 days
Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days	Daily Forecast 16 days
Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days	Climatic Forecast 30 days
Bulk Download	Bulk Download	Bulk Download	Bulk Download (global cities)	Bulk Download (global cities + ZIPs of US, EU, UK)
Basic weather maps	Basic weather maps	Advanced weather maps	Advanced weather maps	Advanced weather maps
Historical maps	Historical maps	Historical maps	Historical maps	Historical maps
Global Precipitation Map - Historical data	Global Precipitation Map - Historical data	Global Precipitation Map - Historical data	Global Precipitation Map - Historical data	Global Precipitation Map - Historical data
Trigger Dashboard	Trigger Dashboard	Trigger Dashboard	Trigger Dashboard	Trigger Dashboard
Road Risk API (basic configuration)	Road Risk API (basic configuration)	Road Risk API (basic configuration)	Road Risk API (basic configuration)	Road Risk API (basic configuration)
Air Pollution API	Air Pollution API	Air Pollution API	Air Pollution API	Air Pollution API
Geocoding API	Geocoding API	Geocoding API	Geocoding API	Geocoding API
Weather widgets	Weather widgets	Weather widgets	Weather widgets	Weather widgets

WEATHER API: GET YOUR APP KEYS

OpenWeather

Dear Customer!

Thank you for subscribing to Free OpenWeatherMap!

API key:
- Your API key is **08162e18aacf2951a66eebe3a6af1459**
- Within the next couple of hours, it will be activated and ready to use
- You can later create more API keys on your account page
- Please, always use your API key in each API call

USING WEB API TO BUILD A SIMPLISTIC LLM-POWERED AGENT

- Prerequisite: OpenAI Key
  - https://platform.openai.com/api-keys
  - setx OPENAI_API_KEY “<your_key>”
- Send and receive messages
- Define tools (python functions) and let LLM use them

- Task:
  - Use LLM to define sqlite db using image of a schema
  - Create local DB
  - Use LLM to fill the DB with sample data
  - Define Python functions to access the DB
  - Let LLM use these functions to study the DB and create report

WebAPItoLLM.ipynb

```python
url = "https://api.openai.com/v1/responses"

headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {openai_api_key}"
}

prompt = """
Inspect the DB schema in the image.
Generate instructions to reproduce these tables in sqlite3 dialect.
Double-check sqlite3 syntax.
DO not include any text, return SQL code only.
"""

data = {
    "model": "gpt-5-mini",
    "max_output_tokens": 15000,
    "input": [
        {
            "role": "user",
            "content": [
                {"type": "input_text", "text": prompt},
                {
                    "type": "input_image",
                    "image_url": "https://www.astera.com/wp-content/uploads/2024/05/Database-schema.png"
                }
            ]
        }
    ]
}

response = requests.post(url, headers=headers, data=json.dumps(data))
```

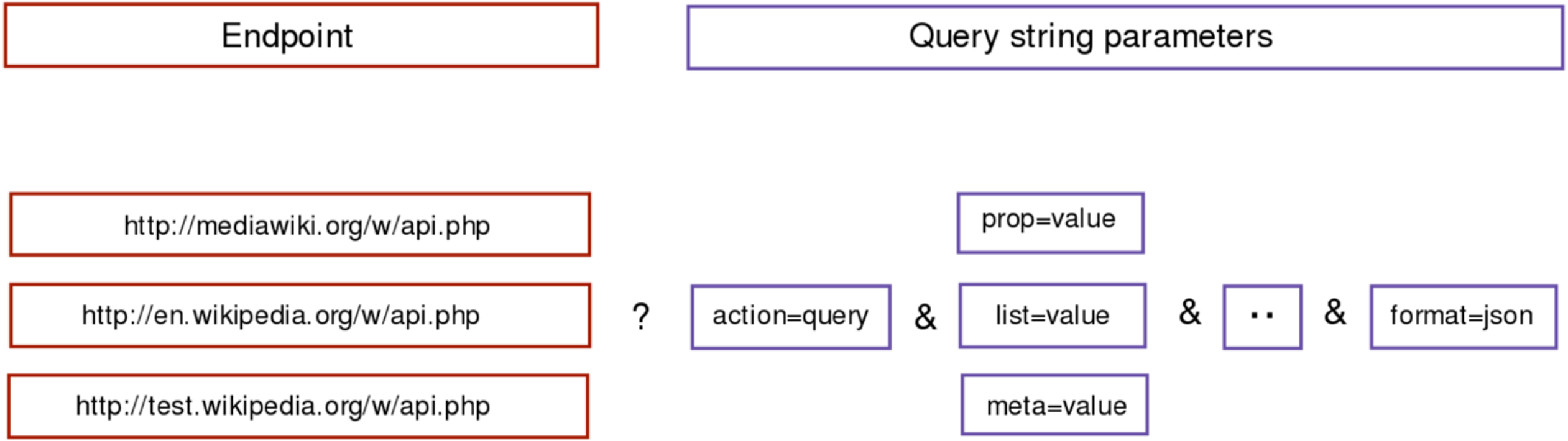
OPENWEATHERMAP

- Endpoint: http://api.openweathermap.org/data/2.5/
- Methods:
  - Weather
  - Forecast
- Parameters:
  - appid – get from https://home.openweathermap.org/api_keys
  - City: http://api.openweathermap.org/data/2.5/weather?q=Jerusalem,IL&appid=08162e18aacf2951a66eebe3a6af1459
  - Location: http://api.openweathermap.org/data/2.5/forecast?lat=31.77&lon=35.22&appid=08162e18aacf2951a66eebe3a6af1459

WorkingWithWebData_partB.ipynb

More Web APIs

WIKIPEDIA API



WIKIPEDIA API

WorkingWithWebData_partB.ipynb

More Web APIs

- Search

https://en.wikipedia.org/w/api.php?action=query&list=search&srsearch=albert%20ainstein&format=json

Snap to the most relevant page

https://en.wikipedia.org/w/api.php?action=opensearch&search=Liverpool&limit=10&format=json

Get Page Content:

https://en.wikipedia.org/w/api.php?action=query&prop=revisions&rvprop=content&rvsection=0&titles=Albert%20Einstein&format=json

- Get list of images:

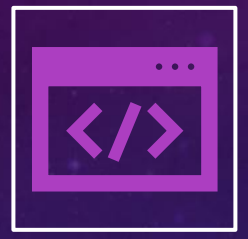
https://www.mediawiki.org/w/api.php?action=query&list=allimages&aifrom=Albert%20Einstein&format=json

List all links:

https://en.wikipedia.org/w/api.php?action=query&titles=Albert%20Einstein&prop=links&pllimit=max&format=json

Get pages with specific properties (e.g. using a specific infobox, such a company or state)

AUTOMATING BROWSER



Certain websites utilize JavaScript and run extensive code in the browser.

Selenium browser is an application that allows python code to automate your browser.

Can closely mimic user interaction with a web site:

- All JavaScript code is executed as in true browser
- Type content, scroll, click links, etc.
- Can be used to render and generate images of the web page.

Selenium

uses the browser installed on your computer (Google Chrome , MS Edge, Firefox)

Must download a comparable driver (executable).

Since browsers are frequently updated, it's best to update the driver with webdriver-manager

- `pip install -U selenium`
- `pip install -U webdriver-manager`

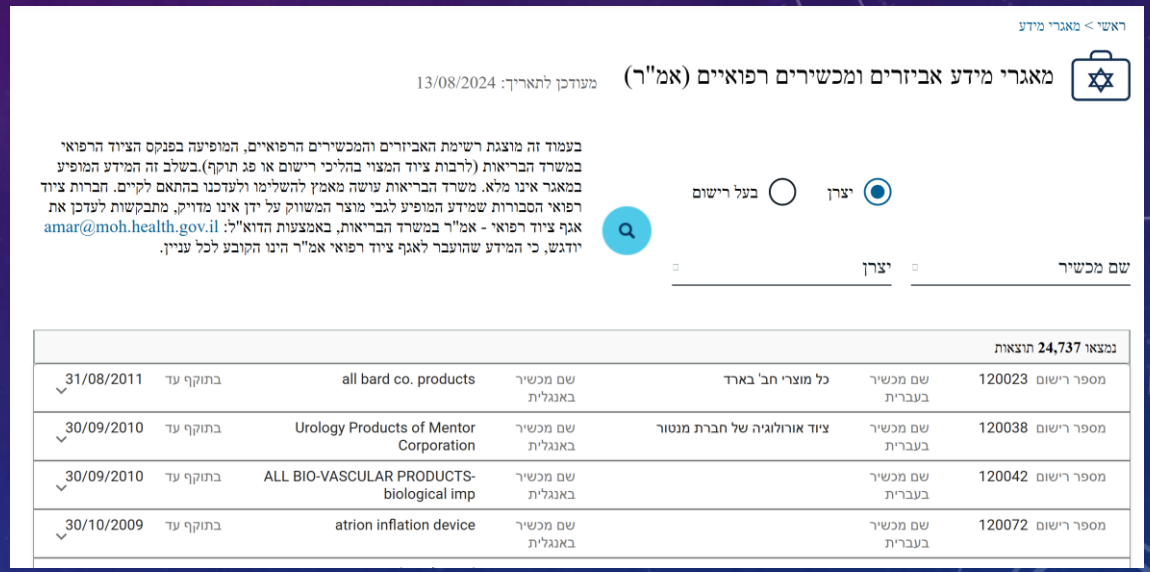
SeleniumDemo.ipynb

CRAWLING WEB SITES

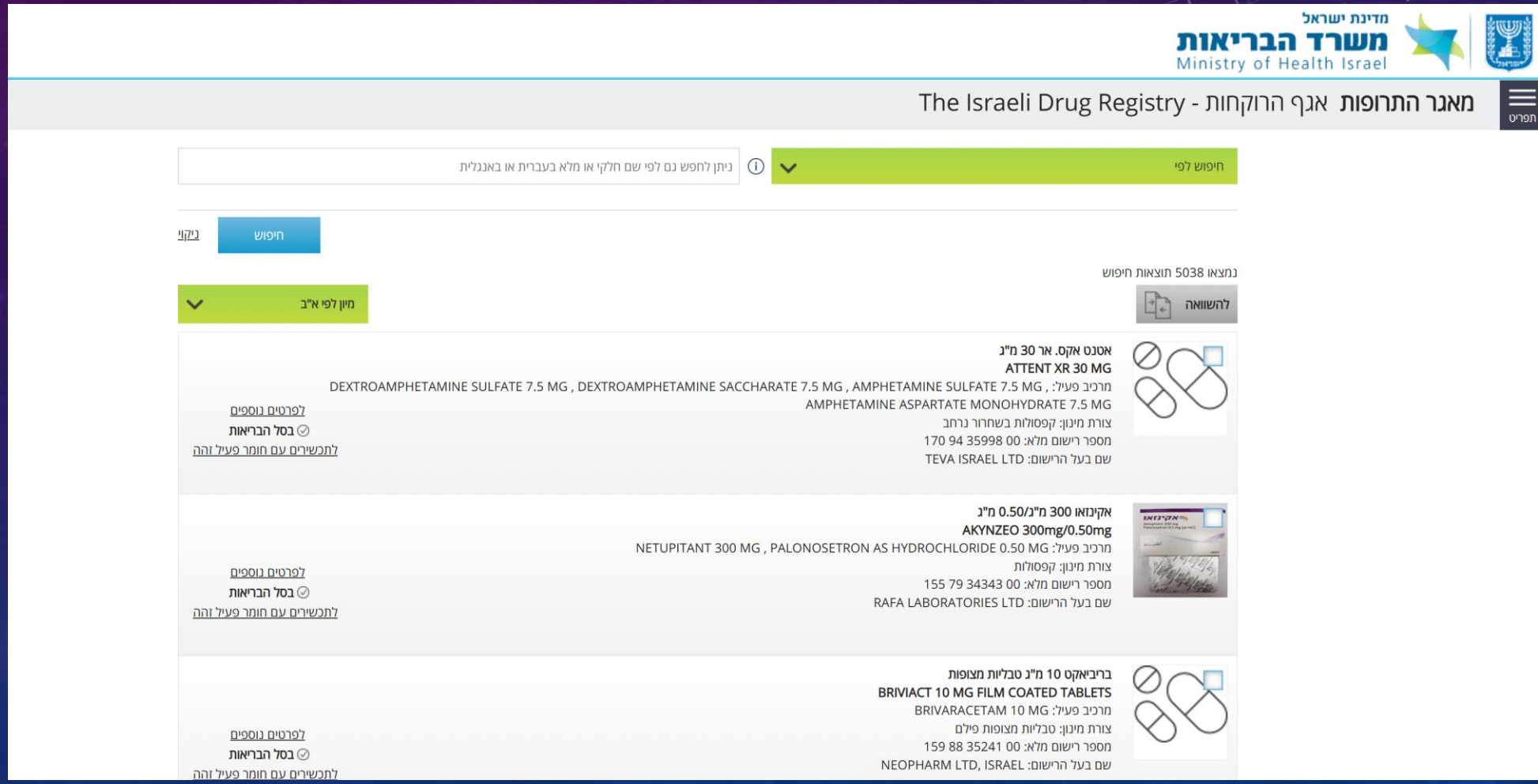
- Robots.txt – contains information that regulates access to the web site content by crawlers
- Sitemap: contains web site schema / details about the web site content.
  a. https://a-z-animals.com
  b. Harvard.edu

HACKING WEB APPLICATIONS : פנקס האמ"ר
רישוי ציוד רפואי - אביזרים ומכשירים רפואיים - אמ"ר

- מאגרי מידע
- (אמ"ר) אביזרים ומכשירים רפואיים
- https://www.gov.il/he/pages/amr-licensing
  https://medicaldevices.health.gov.il/



HACKING WEB APPLICATIONS: THE ISRAELI DRUG REGISTRY



https://israeldrugs.health.gov.il/#!/byDrug