# Usearch

## Technology

The Usearch web search engine is built entirely from AI-generated data. It eliminates the need to collect users' data, such as search queries to be bootstrapped or improved. Its groundbreaking technology generates synthetic data that is identical to real users' .data, thus making it accurate, scalable, independent and truly private

# Introduction

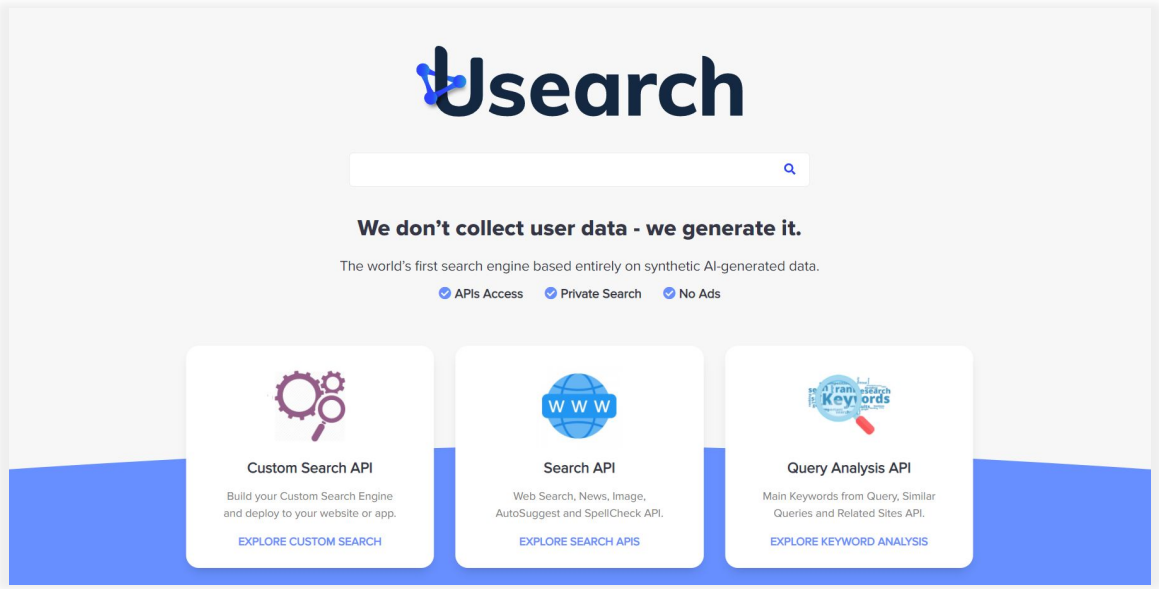
Web search engines are an essential part of many aspects of society today. They serve as the primary portal for accessing the rich abundance of knowledge that is accessible through the world wide web. Web search engines are expected to answer any user question in real time and show the most relevant webpages for every topic.

Building a competitive web search engine is notoriously difficult; and so far, every attempt has ended in the internet graveyard. Google's technology is light years ahead of everyone else's. During the 20 years of its evolution, Google has been collecting massive quantities of user data (such as user's search queries and the webpages they click on). It's massive head start and humongous success have set it at the game's pinnacle and has raised a colossal technological barrier of entry – the so-called *web search bootstrapping problem*. This barrier of entry into this field is so insurmountable that even the most tenacious and deep-pocketed organizations shy away from it.

It is true that you cannot build a competitive web search engine without huge amounts of users' data. **However, *the myth that a web search engine cannot be built without collecting real user's data has now been exploded. That's where AI comes into the picture.***

Instead of **collecting** user data – we **generate** it.

Our recent technological breakthroughs have solved the bootstrapping problem by generating synthetic data that it is identical to real users' data. Our innovative search technology enables us to build completely independent web search engines entirely from scratch, without collecting any user's data – and it's extremely competitive with Google and Bing.



# 1. Why is building an internet search engine so hard?

Building a competitive internet search engine is notoriously difficult.

- Why is it so hard?
- Why is Google's technology light years ahead of everyone else?

Here's why –

- **The sheer size and breadth of the web itself is overwhelming**. With over 150-trillion webpages, the Internet's noisy nature makes webpage indexing a significant hurdle. Only 1 of every 3,000 webpages has sufficient quality to be indexed, so deciding what webpages to index is a key factor to search engine.

- **It is practically impossible to index all the words in every webpage.** With an average of 300 words per webpage, indexing requires an in-depth, thorough understanding of each page, as well as an understanding of which parts of each page are worth indexing. Even after building an index, a huge noisy space still remains in which it's extremely difficult to find results.

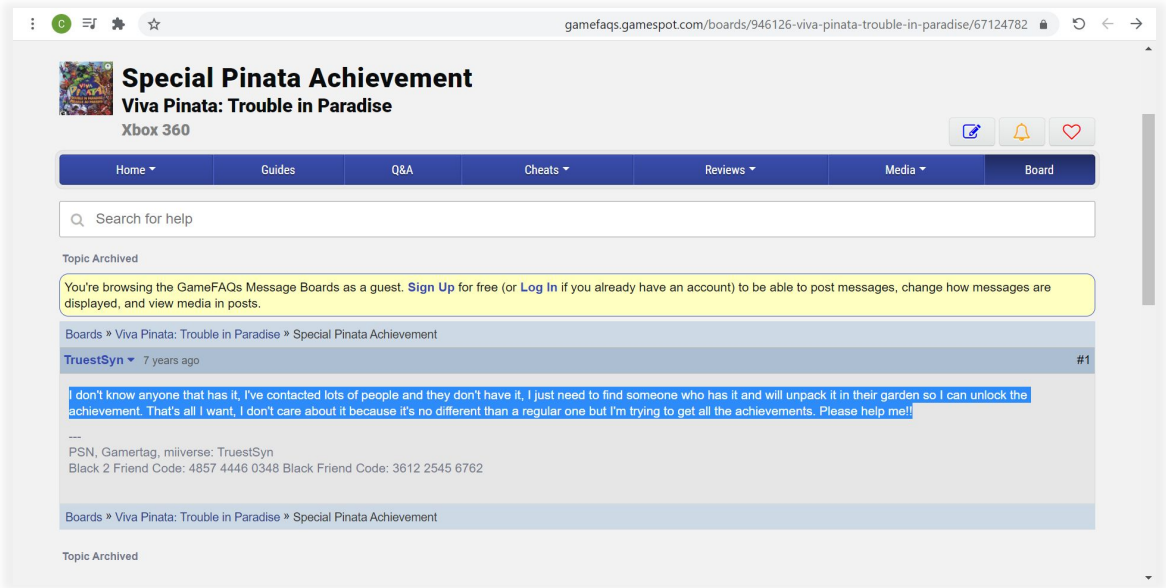
- **One cannot build a competitive search without huge amount of users' data.** Knowing which pages people visit after a query is vital for improving the search results and determining the quality of a webpage.

- **It's too expensive to compute search results on the fly.** So most queries must be precomputed in order to make a search engine practical. Caching is not viable either, because it depends on the exact same query having been encountered previously.

# 1.2 State of the art solutions – No Threat to Google

Among the most popular search solutions one can find Elasticsearch and Solr. They are distributed search engines based on **the inverted index scheme**. For example, Elasticsearch has become one of the most popular open-source search engines in the world today. So why hasn't anyone built a successful internet search engine using Elasticsearch? And why do Elasticsearch-based search engines perform so poorly compared to Google? In the following sections we will explain why **building an internet search engine using Elasticsearch as the core indexing technology has almost zero chance of success.**

Indeed, Google understood this long ago and began eliminating its massive inverted index infrastructures back in 2010. Evidently, if you copy/paste a long sentence from an article and try to search for it using Google, there's no guarantee you'll get that article in the results (see figure below). However, using a search engine based on Elasticsearch, you will always get this article as the first result, as this is a key characteristic of the inverted index scheme.




*Google dispensed with its inverted index scheme – evidently, even 38 identical words doesn't show this article*

Google made a smart move when they gave up indexing all the words in the text, even though now they cannot fully guarantee the retrieval of a webpage by copy/pasting sentences from it. However, this is a niche scenario in internet search, as more than 90% of the search queries have no more than four words, and can be solved, if needed, using a variety of techniques like local sensitive hashing. The benefit of carefully choosing which words to index is significant – both a smaller-sized index and a less noisy search space. This way Google achieved both scalability and more accurate results.

Here are some of the theoretical and practical barriers to **indexing** the web using Elasticsearch. These constraints also generally apply to any inverted-index-based technology –

- The core algorithm of an inverted index scheme is based on a list intersection. This is a computationally inefficient operation, which is impractical when it comes to the web and its billions of webpages.

- An N-gram is a contiguous sequence of N words from a webpage. Using N-grams is a popular way to avoid list intersections and to reduce web noise. But, as the number of N-grams increases along with the number of webpage words, it becomes increasingly impractical to store N-grams.

- A search query may contain redundant words, omitted words, altered words and so on. Elasticsearch does not provide a way to predict the possible forms of the query as they are likely to appear in the relevant webpages.

Even if you manage to overcome these challenges and index the webpages using Elasticsearch, you are still left with the even more challenging problem of **ranking the results**. As Elasticsearch ranking algorithms are based on textual measures this problem turns out to be intractable. This is explained in details in the following section.

## 1.3 The Textual Deadlock

Textual measures refer to the relationship of search query words to the text in a webpage. For example, how many search words appear in the webpage, the position of the search words in the webpage, the distance between two search words in the webpage and so on.

An internet search engine based on textual measures is doomed to fail, because it may perform well on one query, but perform poorly on another. Relying on textual measures in an internet search engine is akin to making blind decisions – and should be avoided at all costs. Evidently Google barely uses textual measures. This is precisely the opposite of Elasticsearch, which is solely based on textual measures.

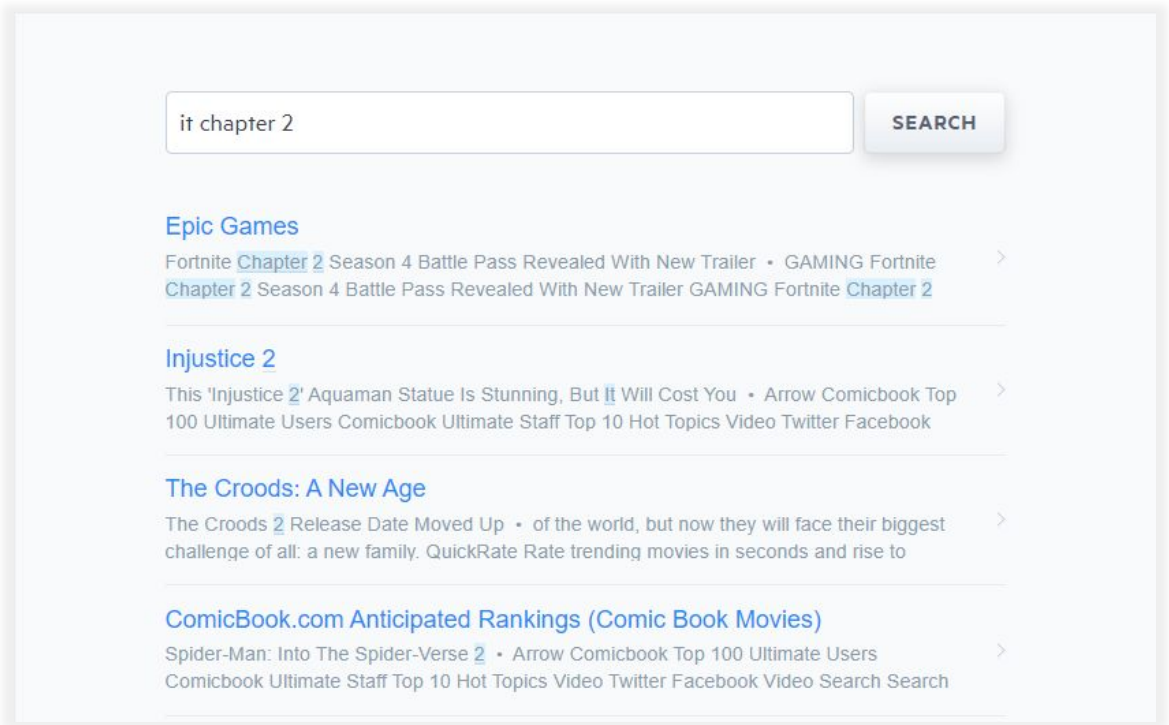
Here are a few examples of why not to rely on textual measures.

### Example 1

Someone enters the search query *outerworlds*. Most likely, they mean the PC game Outer Worlds. The search engine must chop up the query correctly in order to get relevant results. One approach is to use a dictionary to separate the query outerworlds into its two words – outer and worlds. This might work…. But, if you apply the same approach when someone enters the query *endgame* (which is the name of a popular movie), the query is chopped up into the words *end* and *game*. In this case, the search engine returns webpages containing both the words *end* and *game* separately, and such webpages are unlikely to be relevant to the movie.

### Example 2

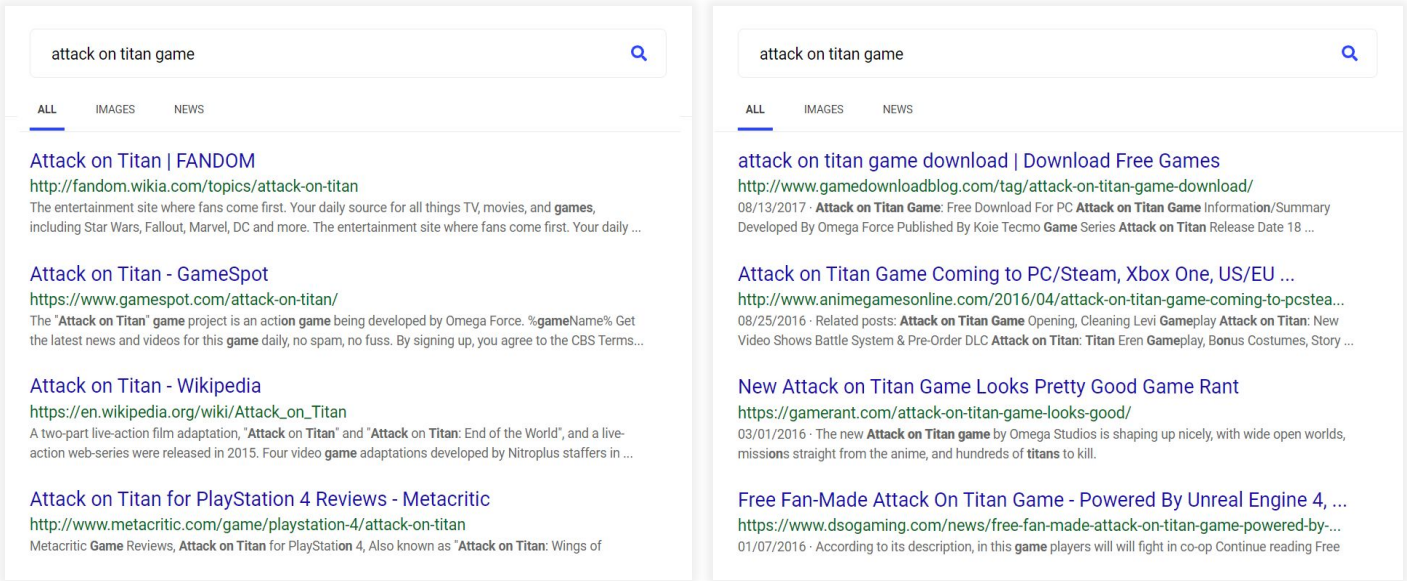
Here's another example. Consider the query *It chapter 2*, which is a popular horror film. This movie title consists of three generic words – *it*, *chapter* and *2*. Google knows that this query is about a movie and shows the correct results. Trying this same query in an Elasticsearch engine, most likely produces irrelevant results. This is because the word *it* is a stop word (commonly used word), and search engines typically ignore stop words in order to reduce noise. This same idea is behind td-idf, which is the statistical technique used by Elasticsearch to determine how important a word is to a webpage. However, in this particular query, *it* is the most important word and ignoring this word produces irrelevant results. See figure below.




*Search results for **it chapter 2** on Swiftype (an engine based on Elasctisearch). td-idf tends to ignore the word it, which yields irrelevant results.*

While trying to solve this problem, you might come up with the idea of keeping the phrase *it chapter 2* as an exact 3-gram in the index and configuring Elasticsearch to show webpages that contain these three words as an exact-match phrase. This approach might work…. But let's see what happens when someone else enters the query *Attack on titan game*?. Google knows that it's a PC game. In fact, the word *game* is redundant here. Evidently, in Google, the queries *Attack on titan game* and *Attack on titan* both return the exact same results. However, the exact-match approach will favor webpages that contain the exact phrase *attack on titan game*.

In the figure below we present two options for the search results page for the query "attack on titan game". In the left results page we allowed the phrases "attack on titan" and "game" to appear separately in the text while in the right results page we favored web pages that contain the exact phrase "attack on titan game". Evidently, by favoring web pages that contain the exact phrase "attack on titan game" we miss the top webpages which are related to the game "attack on titan" - definitely not a Google-like experience. To remedy this situation, you could declare the word *game* as a generic word and exclude it from N-grams. But, remember the query *it chapter 2*. Both the words *it* and *chapter* are generic, and you definitely can't exclude them.
***This results in a textual deadlock!***

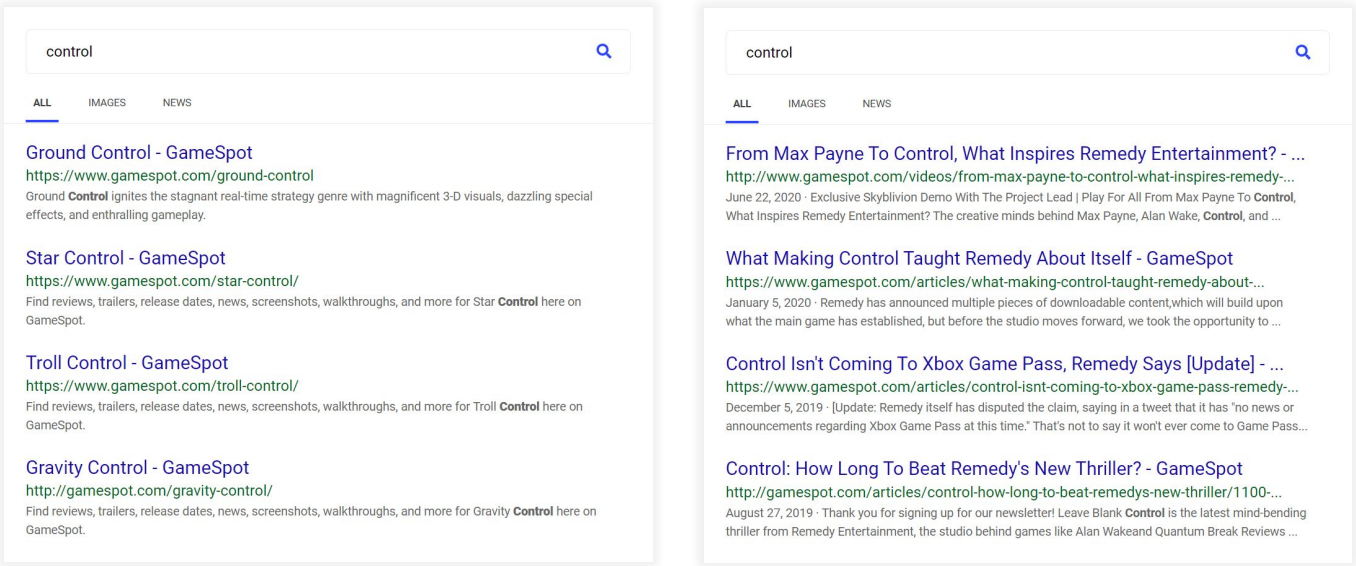



*Search results for **attack on titan game**. The word **game** is redundant. In this case, keeping the phrase as an exact match yields poor results.*

## Example 3

If you're still not convinced yet, here's the best example. Let's look at someone who enters the query *control*, referring to the video game Control.


Here are two possible options for the search results page –
The left results page is based on naive textual measurers, while the right results page is based on more sophisticated measures that require a deeper understanding of the webpage content.




*Search results for the query **control** – textual vs. contextual measures.*

At first, you might think that the results in the left figure are much better. But, No! In fact, none of the results in the figure on the left relate to the video game Control (this is despite the fact that they all contain the word control in their title) whereas all of the results in the figure on the right are highly relevant to the game Control.

**Without a thorough understanding of every webpage, a search engine cannot correctly determine which webpages contain the right results.**

## 1.4 Search Engines based on Textual Measures– No Chances of Success

There are many thousands more examples like the ones above. No matter which textually-based approach you take, it might work well on one query, but poorly on the other. To date, no one has ever come up with a general unified textual approach that covers all cases.

**It's conclusively – exact matches, n-grams, word intersections, word distances, word positions, td-idf, dictionaries, ontologies, encyclopedias, Wikipedia, DBpedia or whatever textual approach you use are seriously limited!**

That's why search engines and search companies like Elasticsearch, Solr, Alogia, Ativio and IBM Watson always perform poorly on webpages. Their technology is fundamentally based on an inverted index scheme, which ultimately relies on textual measures.

**In order to successfully provide a comprehensive, reliable search engine, the use of textual measures must be eliminated. Google understood this over 10 years ago.**

# 2. The World's First Search Engine Based on Synthetic AI-Generated Data

Building a web search engine starts with the acceptance that it is undesirable (and more importantly impossible) to index every word in every webpage. Even Google, evidently, doesn't do it. There is no point trying to index the parts of a webpage that are unlikely to appear in prospective user search queries.
Suppose that for each webpage you had a list of all the search queries that ended with a user clicking on this webpage (say from Google). Clearly any part of a webpage that is not identified with any of the search queries leading to it is completely irrelevant for the retrieval process. Hence, you could focus on only indexing the search queries.

By only indexing these search queries , two significant benefits are achieved –

- Smaller index size.
- Less noise with more accurate results.

It follows that the way to start building an internet search engine is not to naively index all the words and/or phrases in a webpage, but rather to start by generating a Query Log. A Query Log is a huge database of pairs – <query,webpage>, meaning queries and their associated webpages.
You can build this Query Log database  by collecting it from real user queries (such as SERPing Google) or by synthetically generating it from the webpages themselves.

When a search query is entered into a search engine, the search engine shows the most relevant results by doing the following –

- Applying advanced similarity techniques to find the queries in the query log that are the most similar to this user query.
- Fetching the webpages associated with the similar queries.
- Applying refined ranking heuristics in order to select the best webpages from the associated webpages retrieved.

## 2.1 The Race to Query Logs

**The quality of a Query Log (that is, the quality of the associations between the queries and the relevant webpages) is the essence of a search engine's quality.**

A Query Log requires a huge underlying set of queries, but queries themselves are not enough. Collecting queries without their associated webpages, leaves you with the just-as-tough problem of linking the queries to the relevant webpages since each webpage can potentially be associated with innumerable queries. This leaves you with the task of selecting the most relevant queries for each webpage. This is a major hurdle.

While there is widespread consensus that a comprehensive Query Log is crucial to starting a search engine, no company has yet been able to generate pairs <query,webpage> that even come close to Google's quality and quantity. Eventually, they all gave up generating these pairs, and were left to collect pairs from Google by monitoring the web.

One such company was Cliqz, an internet search company that operated between 2015 and 2020. Their vision was to present Europe with an independent digital ecosystem. By forking Firefox and combining it with a browser extension named HumanWeb they collected Query Logs from people searching using Google. They then collected billions of <query,webpages> pairs and maintained an updated Google-based list of pairs. Using sophisticated supervised machine-learning algorithms they trained their system to match user queries to their most relevant collected pairs. But due to the inherent nature of the problem, they failed to generate reliable pairs on their own, and were highly dependent on the pairs they collected by tracking Google. Cliqz's method of collecting Query Logs from Google using browser extensions involved a humongous effort requiring large quantities of users and large infrastructures. Their experience is best described in their farewell blog –
"We failed to reach a scale that would allow our search engine to be self-financing. We have reached several hundred thousand daily users. But – and this is the disadvantage of running our own technology – this is not enough to run a search engine, to cover our costs."

## 2.1.1 The Barrier of Entry

Google built a Query Log by learning from massive quantities of users during its 20 years of evolution and enhancements. It's massive success and head start have set it ahead of the game with a colossal technological barrier of entry. Everyone shies away from what is known as the **web search bootstrapping problem** – even the most successful companies in the world, and even those with the resources to tackle the problem.

**It seemed like building a new successful internet search engine is practically impossible; and indeed, this has been the consensus for the last decade.**

The only other leading internet search companies left in the game (such as Bing, Yandex and Baidu) have all overcome this bootstrapping problem by starting over 15 to 20 years ago. They all have popular browsers and solid core technology. By getting into the game early and creating their own massive eco-systems, these players were able to become large enough to collect the required scale of users' data to maintain their own search engines and to remain profitable.

## 2.3 The Solution – The New AI Paradigm

At Usearch, we decided to take on this previously insurmountable task, despite everyone else having given up. We took on the challenge of inventing new technology that builds large quantities of synthesized quality pairs <query,webpage> using a new independent paradigm. Being independent means that we are not reliant in any way on collecting Query Logs from Google (like Cliqz did) or using Bing API (like DuckDuckGo, StartPage, Dogpile and so on did). Moreover, we don't need any users, browser or query pool to get started.

### 2.3.1 User Query as Memory Contexts

Our solution was inspired by the human brain.

The brain maintains its own *contextual log* (<**context**, **memory**>) that associates context with memories for future retrieval. **Context** is a *cognitive* representation of **memory**. A naive definition of a *context* is the minimal set of entities that best describes the memory. For example, if you had dinner at a three-star Michelin restaurant in Italy last summer, the entities **Michelin**, **Italy** (the **context**) are likely enough to trigger your entire experience from that evening (the **memory**).

We recognized that queries are extremely similar to the contexts that our brains create from our daily experiences.

For example, suppose you read a webpage containing the reviews of three-star Michelin restaurants in Italy, and then a few months later, you want to find this webpage again. You will most likely enter a search query that contains the words "**Michelin**, **Italy**, **Review**" into a search engine. The search query that you will enter is simply the context your brain extracted during the experience of reading the webpage.

Users expect a high correlation between their search queries and the contexts of the search result webpages. To clarify this point, suppose you pick 100 review articles on **Michelin** restaurants in Italy (let's say from Google), and then check out all the search queries that ended up with a user clicking on one of these articles. Most likely, these search queries will contain the words "**Michelin**", "**Italy**" and "**Review**".

***We can conclude that the context of a webpage is practically equivalent to the prospective user queries that will lead to this webpage.***

## 2.3.2 Using AI to Create a Query Log

Thinking of *contexts* as *search queries* and *memories* as *webpages*, we used AI algorithms to create a Query Log of pairs <**queries**,**webpage**> in the same way that the brain creates its contextual log of pairs <**context**, **memory**>. Our implementation is based on Autoassociative Memory Networks and Statistical Pattern Recognition.

This process results in a Synthetic Query Log containing hundreds of billions of synthetically generated <query, webpage> pairs that is almost identical to a Query Log based on real users data.

In order to demonstrate the quality of our query log, here's a few examples comparing the *synthesized queries* that we created (in the columns on the left) with the *queries created by actual humans* collected using popular SEO tools (in the columns on the right).

We give examples for the following searches –

- Red Dead Redemption 2 – A popular video game.
- Game of Throne – A famous TV show.
- Grand Theft Auto V – A popular video game. Its name may be entered by humans in a variety of forms (for example using synonyms and acronyms).

| Synthesized user queries | Real user queries |
|---|---|
| red dead redemption 2 | red dead redemption 2 |
| red dead redemption 2 game | red dead redemption 2 pc |
| red dead redemption 2 wiki | red dead redemption 2 online |
| red dead redemption 2 playstation 4 | red dead redemption 2 cheats |
| red dead redemption 2 pc | red dead redemption 2 map |
| red dead redemption 2 rockstar games | red dead redemption 2 mods |
| red dead redemption 2 online | red dead redemption 2 ps4 |
| red dead redemption 2 tips | red dead redemption 2 wiki |
| red dead redemption 2 cheats | red dead redemption 2 xbox one |
| red dead redemption 2 funny moments | red dead redemption 2 tips |

*Figure 1 – Popular User Queries – Red Dead Redemption 2*

| Synthesized user queries | Real user queries |
|---|---|
| game of thrones | game of thrones |
| hbo game of thrones | game of thrones cast |
| game of thrones season 8 | game of thrones season 8 |
| game of thrones tv | game of thrones season 1 |
| game of thrones cast | game of thrones books |
| watch game of thrones | game of thrones episodes |
| game of thrones finale | watch game of thrones |
| game of thrones season 7 | game of thrones season 5 |
| game of thrones john snow | game of thrones season 4 |
| history behind game of thrones | the mountain game of thrones |

*Figure 2 – Popular User Queries – game of thrones*

| Synthesized user queries | Real user queries |
|---|---|
| gta 5 | gta 5 |
| grand theft auto v | gta 5 cheats |
| gta v | gta 5 online |
| gta 5 online | grand theft auto v |
| grand theft auto 5 | gta 5 cheats |
| gta 5 wiki | gta v mods |
| grand theft auto v pc | gta 5 pc |
| download gta 5 | grand theft auto 5 |
| gta 5 mods | gta 5 map |
| gta v free cd key generator | gta 5 release date |

*Figure 3 – Popular User Queries – grand theft auto v*

You can check out this data at usearch.com by entering a user query and looking at the results in the **Top Similar Queries** dashboard.

***Evidently, the queries that were predicted by our Query Log (the Synthesized queries that we created) are almost identical to these most popular real-user queries.***

# 3. What Google is Hiding from You

Google's mission is "*to organize the world's information and make it universally accessible and useful*". This universal accessibility is only reflected to us via its search engine. This flawless search engine processes 5B queries a day and has more than 20B indexed webpages in English. It provides blazingly fast answers, with an impressive ratio of success that casts a shadow over all other search technologies and competitors. As users, we only experience the final results, while the entire processing flow, data organization and all ranking considerations are completely kept hidden from us.

- What data does Google collect?
- How does Google organize and analyze the data?
- How does Google's data organization and analysis output provide accurate search results for almost every possible query?

It's worth mentioning that, many data aggregators are already in possession (in part) of similar data to Google's. This data is primarily obtained by monitoring and analyzing the web's traffic. For example, by using browser extensions installed on users' browsers that monitor their web traffic, track their search queries and take note of the webpages to which each search query leads by a click. In other words, they capture a snapshot of Google's so called Query Log. But this snapshot does not provide any real insight into how Google attached the queries to the relevant webpage.

The ability to organize this data is the real essence of Google's technological advantage and is its core technology. Monitoring is doable, but data organization requires a deep understanding of internet data.

## 3.1. Understanding Queries – Knowing Where to Find Answers

To get a sense of how to organize the data, you must first understand Google's query processing flow and ranking considerations, meaning the lifecycle of a search query, from the moment it hits Google's servers until a user gets the results. Here's a simplified heuristic overview –

- **Understanding the query context.** In order to understand a user's intent, each query must be analyzed in order to distinguish between the primary keywords (the context) and the supplementary (secondary) keywords. The secondary keywords help focus the search on the aspect of the main keyword in which the user is interested.

- **Finding similar queries.** It is unreasonable to expect a user to enter a query that exactly matches the relevant webpages. Hence, similar queries with the same context must be found that have already been processed and have performed well, in the sense that they lead to the relevant webpages with a high degree of certainty.

- **Getting popular and relevant domains.** Showing results from the most popular and relevant domains for each search query increases the results' reliability and relevance. It also significantly reduces web noise and shrinks the search space.

With this goal in mind, an internet search engine has the three main building blocks accordingly –

- **Query Analyzer** – To understand the query context and user's intent.

- **Similar Queries Finder** – To find the most popular similar queries with the same context.

- **Popular Sites for Query Finder** – To prefer the webpages of the most relevant and reliable sites for each query.

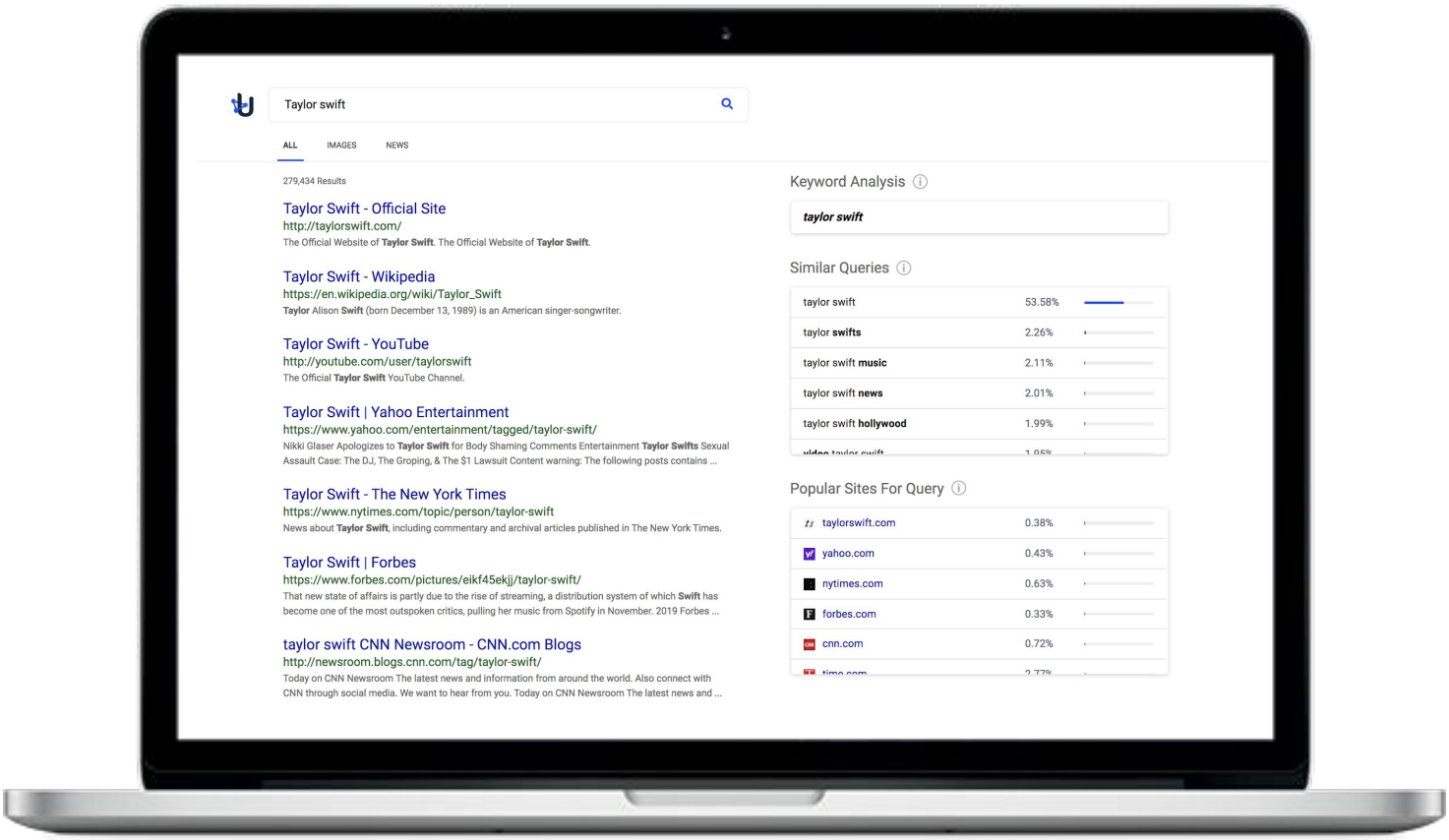
These three building blocks help organize the web, understand what the user is looking for and know where to find the results. These are also the building blocks of other state-of-the-art technologies, such as – search engines, voice assistants, chatbots and many more. For example, a voice assistant follows a similar route –

- It processes and analyzes a huge amount of data and information.
- It must understand the users' intent and find all similar forms of the same question already in the system.
- It finds and scores a set of relevant answers.
- It shows the most likely answer to the question.

Google does not expose its underlying data or allow access to it, such as by API access. In fact, Google shut down its search API services back in 2013.

# 3.2 The Building Blocks Finally Revealed

Usearch.com is our internet search engine. It is accessible for free for anyone. It was developed completely from scratch  using novel AI algorithms. It is not based on any other existing search engine or search technologies. It does not collect any data from other search engines and does not monitor, aggregate or analysis user queries. The  striking resembles to Google's search experience  indicates that indeed we managed to synthetize data similar to Google's and our query processing flow overlap. Entering a search query in usearch.com shows a dashboard that visualizes these three building blocks, as well as showing the search results.




# 3.2.1 Query Analyzer

The Query Keyword Analyzer Block breaks down the query into keywords in order to understand the user's intent. The input to the Keyword Analyzer is the query entered by the user. The output consists of two types of data –

- **Primary keywords.** These are the main keywords of the query and are shown in bold.
- **Secondary keywords.** These are additional keywords that are not regarded as the primary keywords, but are rather additional keywords that add supplementary information about the primary keywords.

For example, consider the query “cyberpunk 2077 release date for ps4”. The user is asking about the planned release date of the game cyberpunk 2077 for the PS4 console. The output of the Keyword Analyzer is –

***cyberpunk 2077***,*release date,ps4*

As you can see, the primary keyword (shown in bold) is indeed **“cyberpunk 2077”** and the secondary keywords are **“release date”** and **“ps4”**. This pinpoints the exact type of information users are looking for about the primary keyword **“cyberpunk 2077”**.

## 3.2.2 Similar Queries

The Similar Queries Block shows the top most similar queries to the user’s query. Similar queries are naturally divided into two types –

- Popular variations of the user query. These variations include synonyms, acronyms and so on.
- Augmentation queries that reveal what other users have searched for in the context of the given query.

For example, consider the query **“Grand theft auto 5”**, which is a popular video game. The top five similar queries are –


*Top 5 queries for **“grand theft auto 5”***

Indeed, this example shows both types of similar queries mentioned above –

- The similar queries **“gta 5”** and **“grand theft auto v”** are of the first type, meaning that they are popular variations of the query, meaning acronyms, synonyms and so on.
- The similar queries **“gta 5 online”** and **“gta 5 wiki”** are of the second type, meaning that they are popular queries that reflect the user interests in the context of the game **“grand theft auto 5”**.

The progress bar shows the share of each query from the total number of similar queries.

## 3.2.3 Popular Sites for Queries

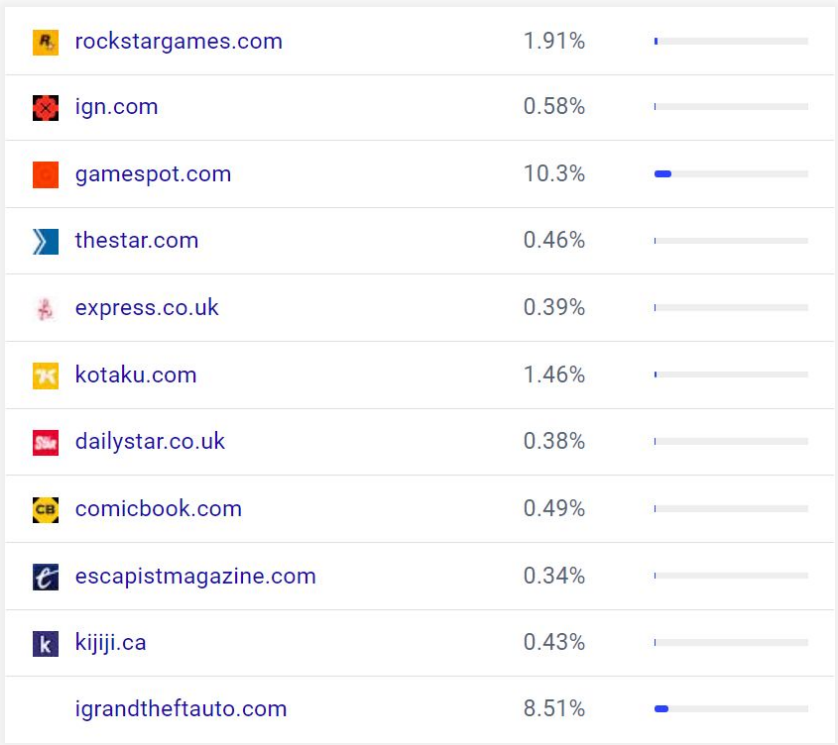
The Popular Sites for Query block shows the most popular sites that have web pages relevant to the query. Search engines prefer to show webpages from popular sites, because such sites have high traffic and quality results for the query.

The domains in this block naturally represent both of the following types of sites:

- Specific sites that are almost exclusively dedicated to the query.

- Sites that contain a high volume of webpages relevant to the query (high authority domains)
- Relevant sites with high global domain rank (high PageRank).

Consider for example the query **"grand theft auto"**. The following figure shows the most popular sites for this query listed at the top. The sites are ordered from the most popular site to the least – top down. For example, the most popular site is **rockstartgames.com** This is not surprising because the website **rockerstargames.com** is highly specialized to this game, as **"Rockstar Games"** is the company that developed the game. Our system indicates that it is preferable to show results from this site. Indeed, Google shows **rockerstargames.com** as its first result when you search for the query **"grand theft auto".**




*Popular websites for the query **"grand theft auto v"***

All the other sites are popular in the realm of gaming. For example, **ign.com** and **gamespot.com** are global gaming leading sites.

The indicator bar on the right shows the share of the total number of relevant webpages of each site. For example, the site **rockstartgames.com** contains 1.91% of the total number of webpages that were found relevant to the query, while the site **gamespot.com** contains 10.3%. This is not surprising as gamespot.com is a larger website than **rockstartgames.com** and contains more articles about the game **"grand theft auto"**.

There is delicate tradeoff between the popularity of a website and the share of relevant webpages a website has. For example take a look at the website igrandtheftauto.com that appears last in the figure above. This website contains plenty of updated articles and qualitative content, as indicated by its high share of relevant webpages (9. 61%) compared to the other websites. However, a webpage from this website is unlikely to appear in Google's top search result for the query, as it has to compete with webpages from sites that are more popular than it (for example, ign.com and gamespot.com, which are global leading gaming websites.)

## 3.3 Summary

The building blocks described above are at the underlying core of the organization of web data. These blocks provide significant value to search engines and many other web data usage scenarios, such as voice assistants and chatbots (that answer questions), content and user intent recommendation systems, data mining, machine learning and various data sciences.

In practice, building these block requires the collection of massive quantities of real user search queries, tracking of user browsing behaviors and running large scale machine learning algorithms in order to analyze and organize the data.

Our **usearch.com** search engine is built from the bottom up based on these building blocks and is solely based on AI-generated data.

**We don't collect user data – we generate it!**
**We don't track user behavior – we predict it!**

Our AI process automatically generates all the data and thus enables our inherent in-depth understating on this data. By knowing exactly how the data is generated, we can accurately classify and organize it in the best way possible in order to support the aforementioned use cases.

# Usearch

## Technology

The Usearch web search engine is built entirely from AI-generated data. It eliminates the need to collect users' data, such as search queries to be bootstrapped or improved. Its groundbreaking technology generates synthetic data that is identical to real users' .data, thus making it accurate, scalable, independent and truly private