Engineering Anatomy of a Search App

Which can be applicable to any app.....

Marcelo De Barros

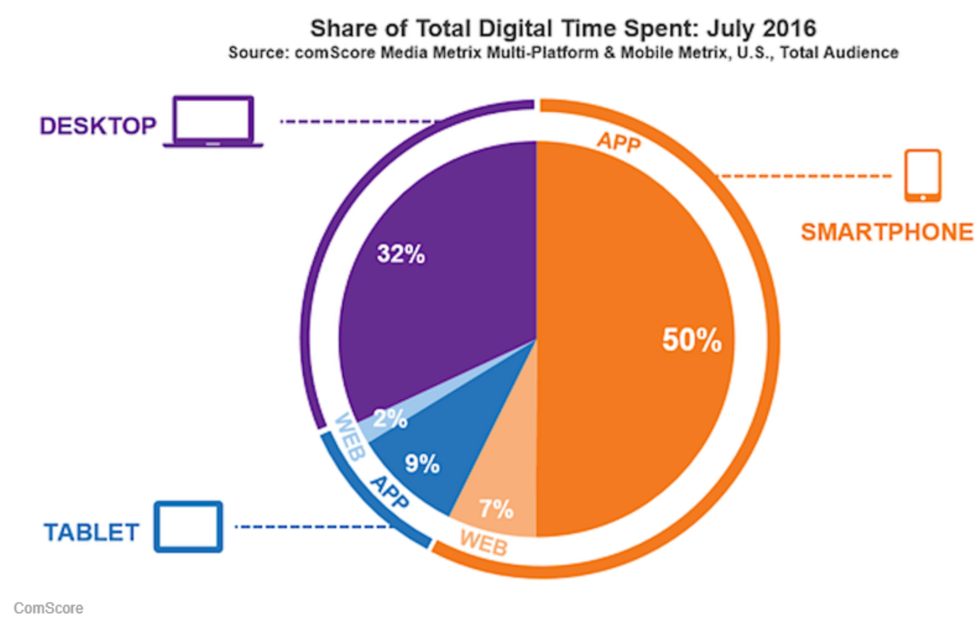
Principal Group Engineering Manager – Bing - Microsoft

The rise (and potentially fall) of mobile apps!

Huge market with millions of apps (iOS and Android)

The pros: native experience, OS integration

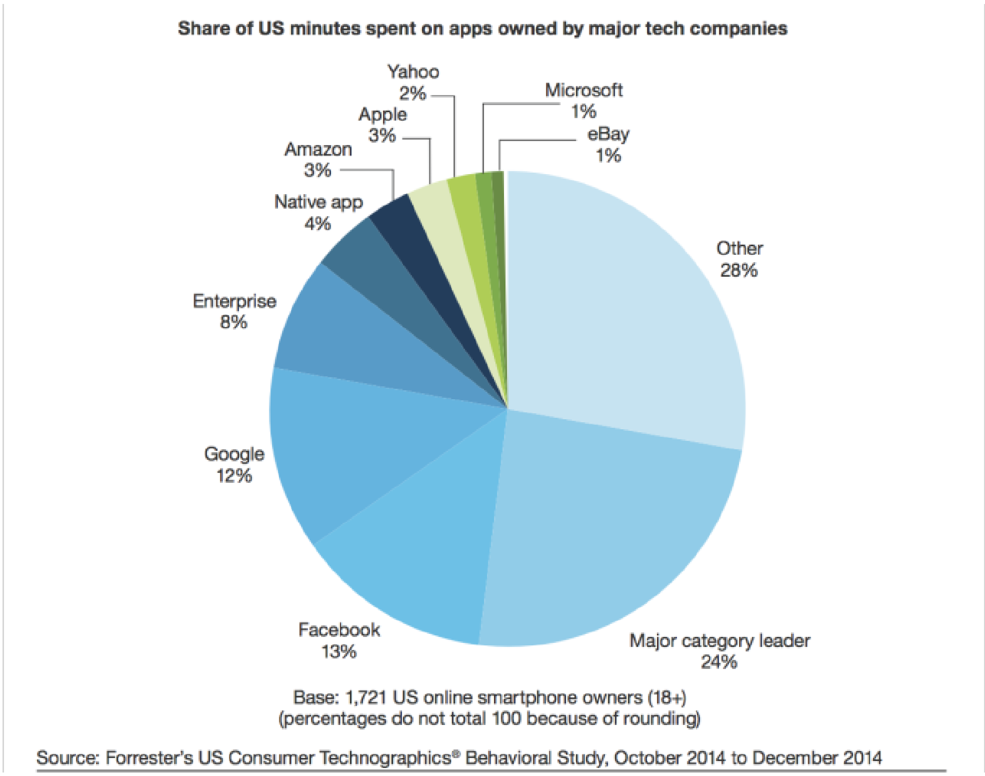
Users are clearly spending more time on Apps than on the Web



The rise (and potentially fall) of mobile apps!

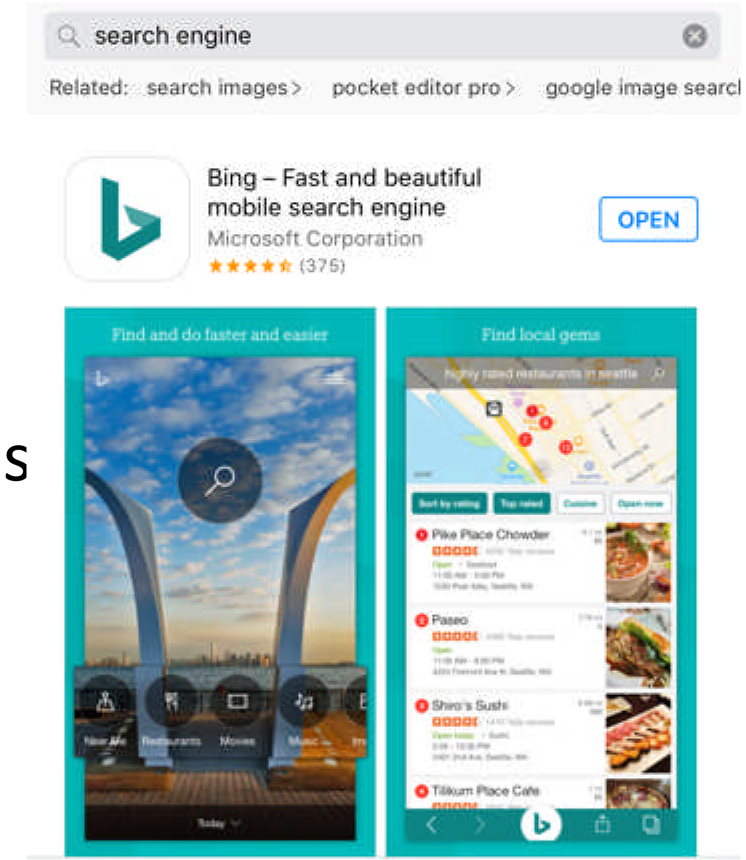
However, users are using fewer apps (basically, they use five apps)

Still, an eco-system that will live for many-many years to come…



Who are we?

- Microsoft – Bing – UX Features and Visual System
- Search App
  - Global Search (web documents)
  - Local Search (location-based)
- Heavy focus on entertainment and browser features
- iOS and Android
- Currently in the en-* market only
- 5-star app* (**changes rapidly though*)



Characteristics of top search apps

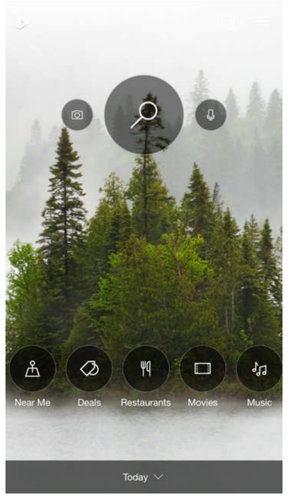
- **Fundamentals**
  - Availability/Stability
  - Agility
  - Instrumentation & Feedback
  - Experimentation
- **Performance**
  - Progressive Rendering
  - Pre-Fetching and Pre-Rendering
  - AMP (Accelerated Mobile Pages)
  - Poor-Network Detection and Optimization
- **Differentiating Features**
  - Offline Mode
  - Visual Search
  - Embedded Machine Learning (reading mode)
- Discoverability and Distribution Models

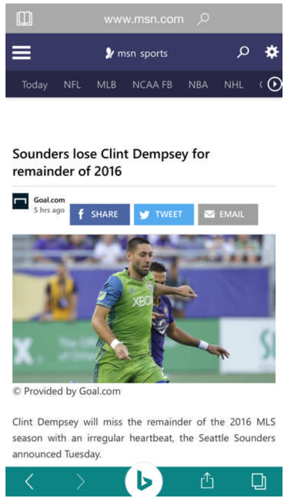
Characteristics of top search apps

- Availability
  - Memory management
  - **Crashes** are directly related to **reviews** and customer feedback
  - Reviews are directly related to **ranking**
  - Ranking is directly related to **downloads**
  - Shift paradigm to even more defensive techniques

Characteristics of top search apps

- Agility
  - Design decisions:
    - **Server-driven** (configs, flights, assets, experiences). Client changes are expensive
    - JSON endpoint, shallow UX (swift/Java) driven by the server changes
    - **ON/OFF features toggle** (server-side)
    - Server-side changes: propagates in **5 min** (7 data centers around the world)



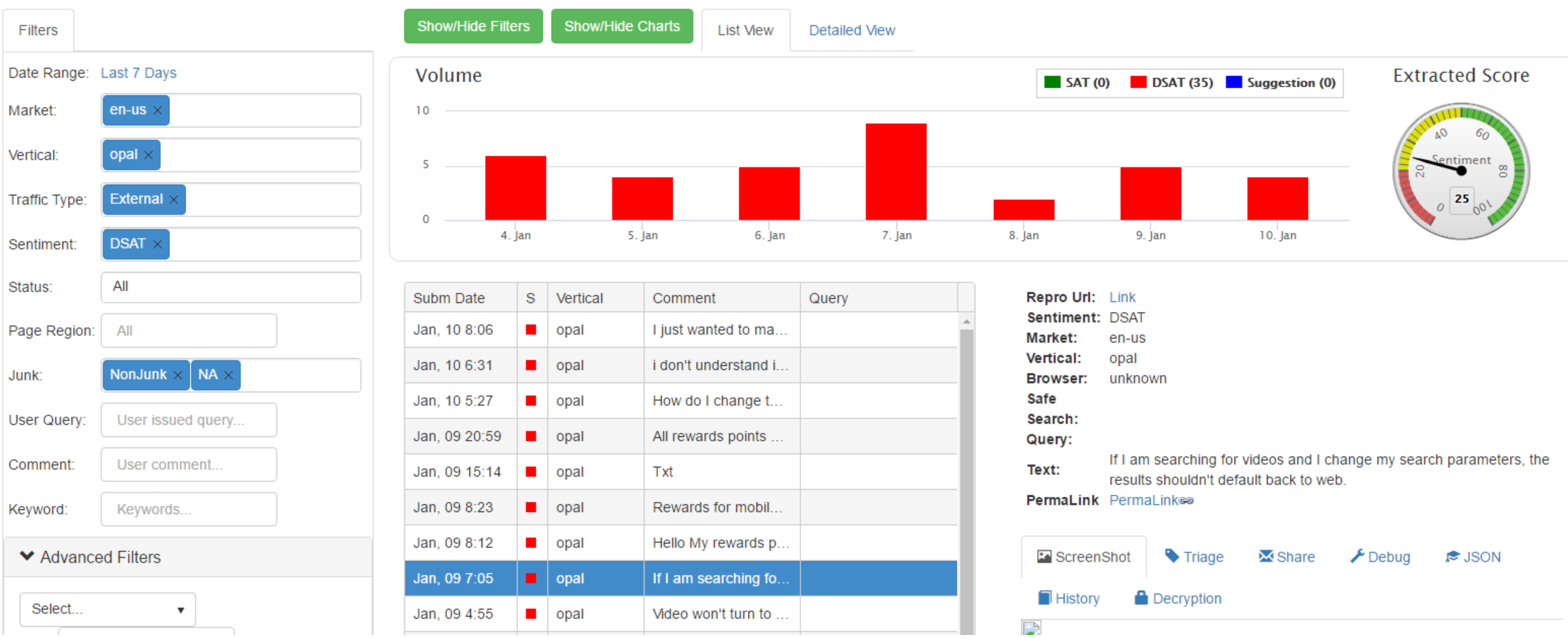


Characteristics of top search apps

- Instrumentation
  - **Every single aspect**: layout, clicks, dwell time, etc. (**no specific user metric!**)
  - Always learn: learn fast, fail fast (also, good code != pleasant features)
  - Options for instrumentation: custom or generic (e.g., localytics for iOS)

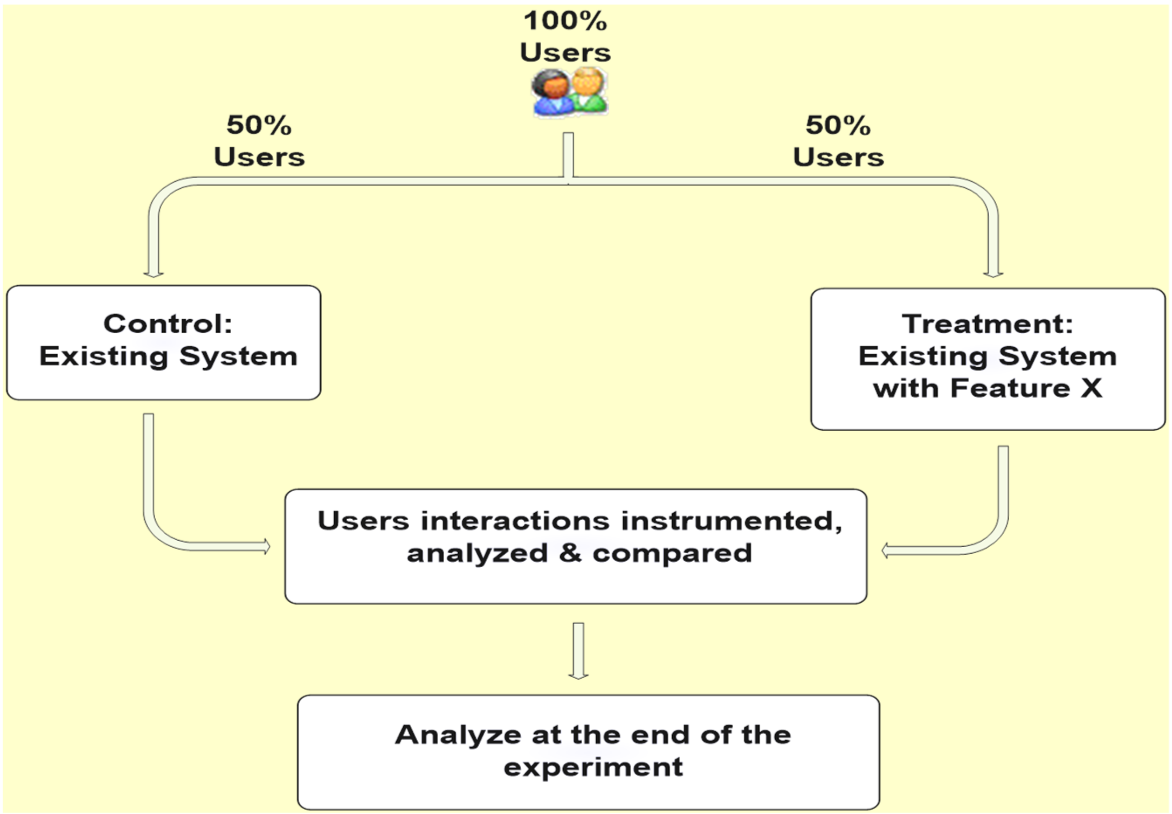
Characteristics of top search apps

- Feedback: app stores and custom
- Listen-Listen-Listen!!!



Characteristics of top search apps

Experimentation

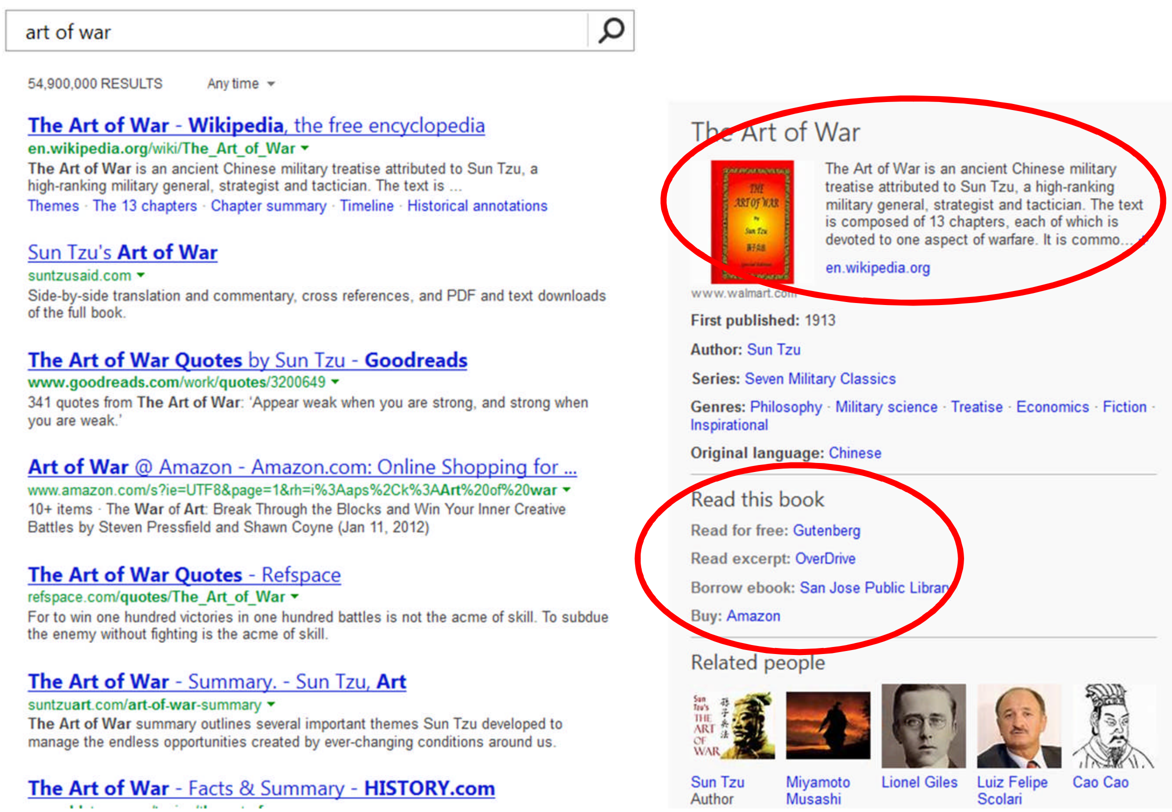


- A/B test is the simplest controlled experiment
- Users are selected randomly into control and treatment (statistics)
- Best scientific way to prove causality, i.e., the changes in metrics are caused by changes introduced in the treatment(s)

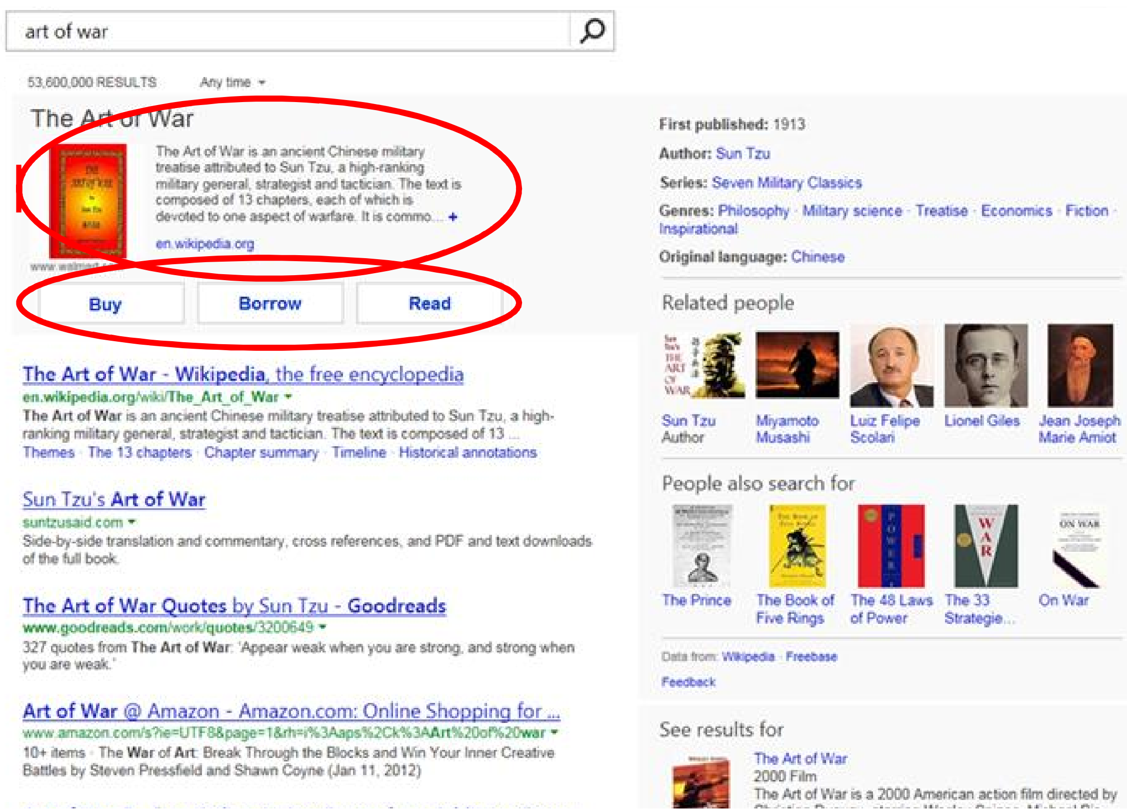
Characteristics of top search apps

Example: True User Intents

Control



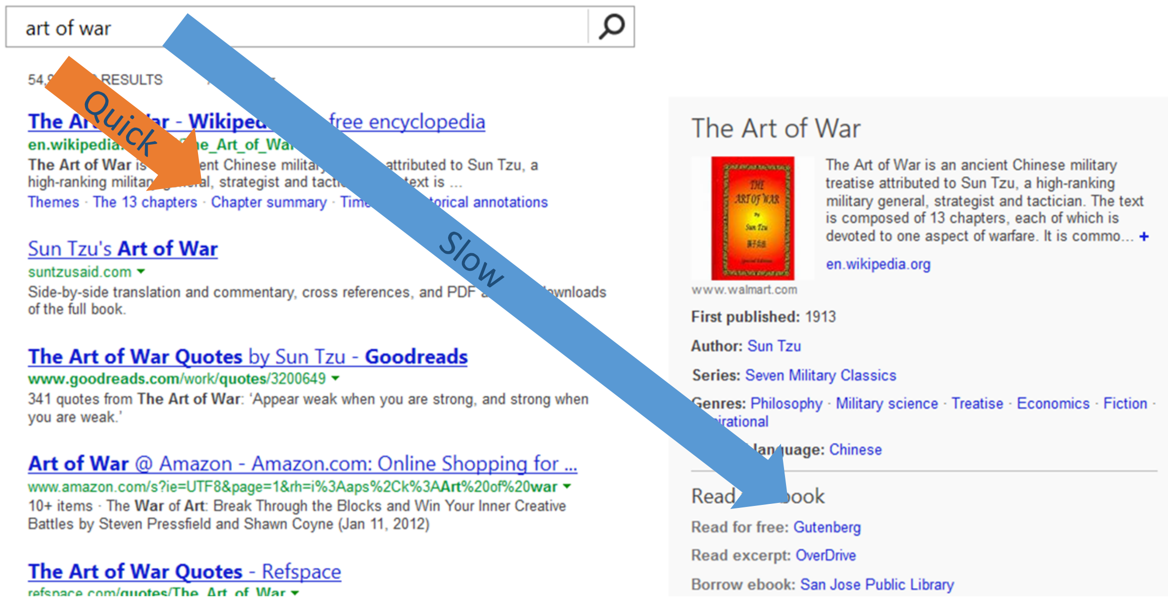
Treatment



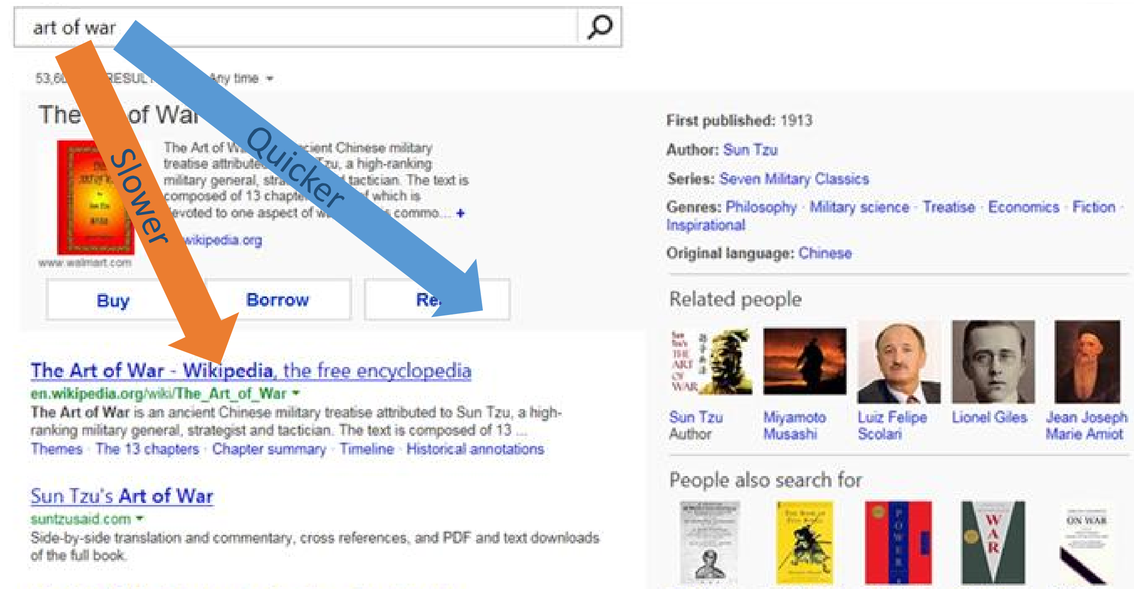
Characteristics of top search apps

Example: True User Intents

Control



Treatment



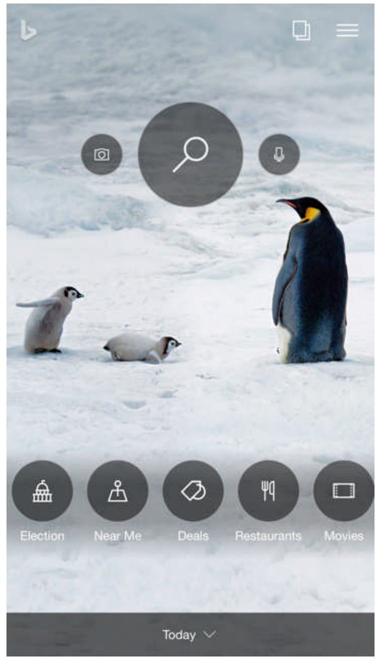
- By modifying the position of the elements on the page (closer/farther from the query box), intent can be inferred through A/B experimentation

Intent: Read the book

Intent: Wikipedia (read about the book)

Characteristics of top search apps

- Always experiment!
  - Data-driven
  - Majority (> 50%) of your traffic should be experiment
  - Use experimentation to decide intent and subjective info, not to find bugs!
  - Examples:
    - Order of the bubbles
    - Heuristics such as for poor-network detection (sliding window)
    - Colors/Fonts/Spacing/Padding
    - AMP (Accelerated Mobile Pages)

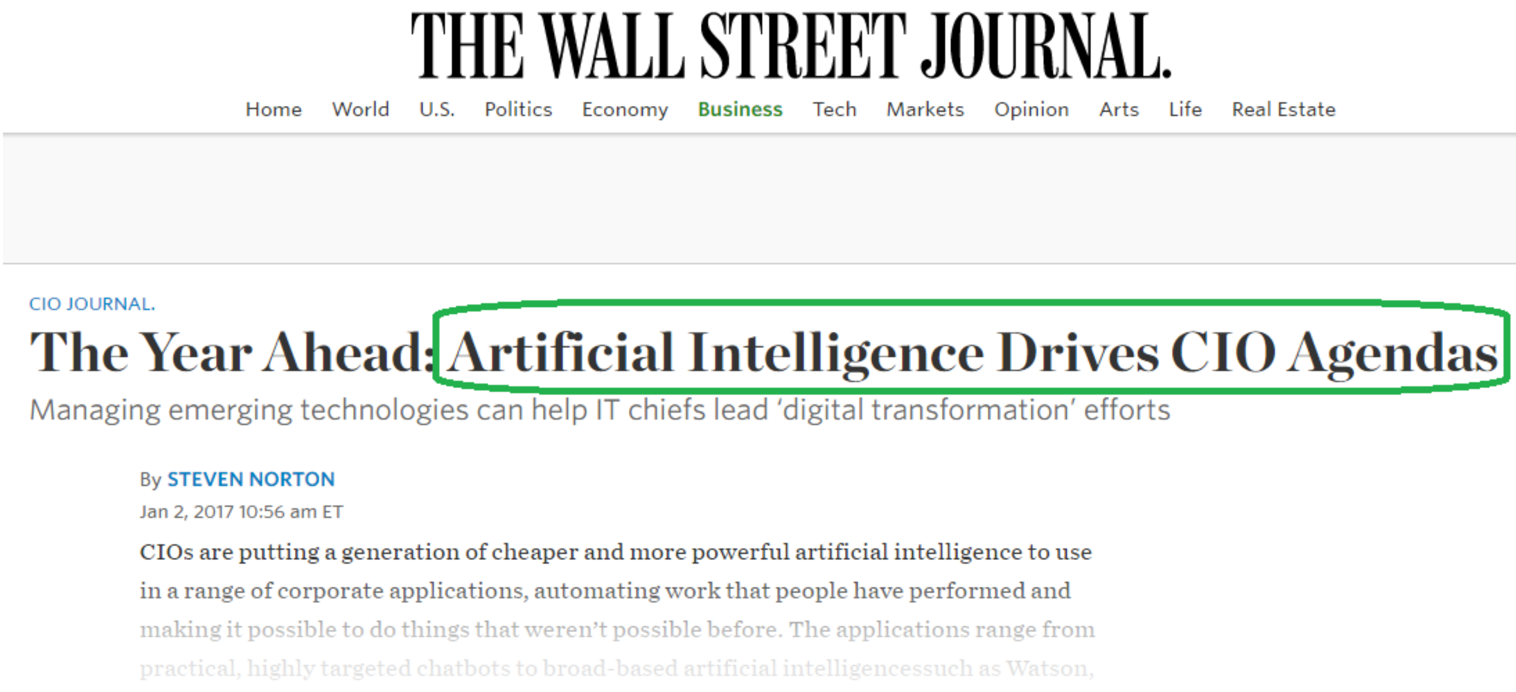


Embedded Machine Learning

The Reading Mode

Machine learning is everywhere now!

- From chat-bots to self-driving cars
- From server-side to client-side features



THE WALL STREET JOURNAL.

Home World U.S. Politics Economy Business Tech Markets Opinion Arts Life Real Estate

CIO JOURNAL.

The Year Ahead: Artificial Intelligence Drives CIO Agendas

Managing emerging technologies can help IT chiefs lead ‘digital transformation’ efforts

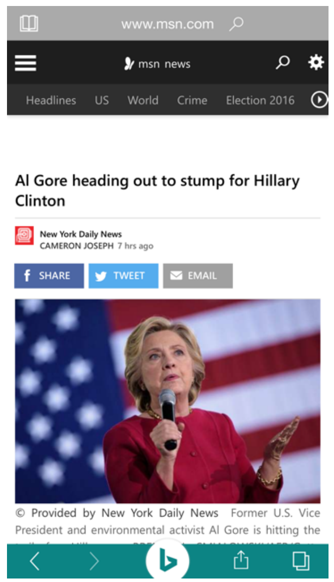
By STEVEN NORTON

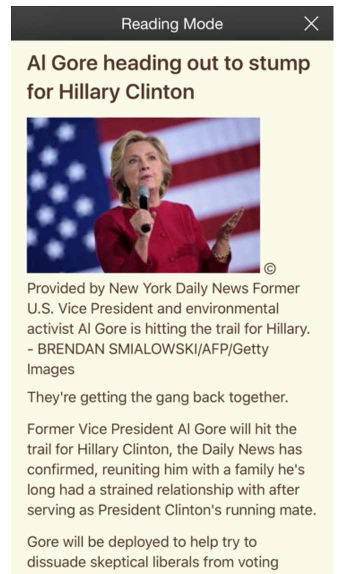
Jan 2, 2017 10:56 am ET

CIOs are putting a generation of cheaper and more powerful artificial intelligence to use in a range of corporate applications, automating work that people have performed and making it possible to do things that weren’t possible before. The applications range from practical, highly targeted chatbots to broad-based artificial intelligencessuch as Watson,

Reading mode

- Problem Statement: *given an HTML page, which elements to keep and which elements to throw away in order to maximize readability?*





- Typical supervised machine learning problem:
  - Training data (laborious)
  - Features selection (images, videos, source, tags, etc.)
  - Select the machine learning model (Neural-Net, Decision Trees, **Custom Clustering**)
  - Training → Test → Repeat

Entity Extraction

```
▶<header>…</header>
▼<div class="row two-columns-fixed-rr">
   ::before
 ▼<section class="column">
  ▼<article class="module">
   ▼<div id="storycontent">
     <!--storytext-->
    ▼<div id="storytext">
     ▶<div id="js-ie-storytop" class="ie--storytop" style="height: auto;">…</div>
     ▶<div class="share-tools share-tools--floater" id="js-sharebar-floater" style="top:
      <h2>OPEC is finally agreeing to cut oil production.</h2>
     ▶<p>…</p>
     ▶<div id="ie_column">…</div>
      <p> Oil prices surged more than 5% in reaction. </p>
     ▶<p>…</p>
     ▶<p>…</p>
     ▶<p>…</p>
     ▶<p>…</p>
     ▶<p>…</p>
      <div id="storyFooter"></div>
     ▶<div class="clearfix">…</div>
     ▶<div class="storytimestamp">…</div>
     </div>
     <!--/storytext-->
    ▶<div class="foot">…</div>
    </div>
    <!-- main -->
    <div class="share-tools pull-left" id="js-sharebar-main"></div>
   </article>
  ▶<div class="cnnoutbrain outbrain-recommended" id="js-outbrain-recommended">…</div>
```

- Idea: find the grouping of **tags that define an article**, while ignoring ads/social/etc.
- **Score reader friendly tags higher** (p, div, article, h5)
- Score tags that contain **large bodies of text higher**
- **Penalize** certain tags/phrases: **ad, share, social, sidebar**
- Apply **scoring** over the DOM tree (**clustering with thresholds**)

Performance Characteristics

Performance Characteristics

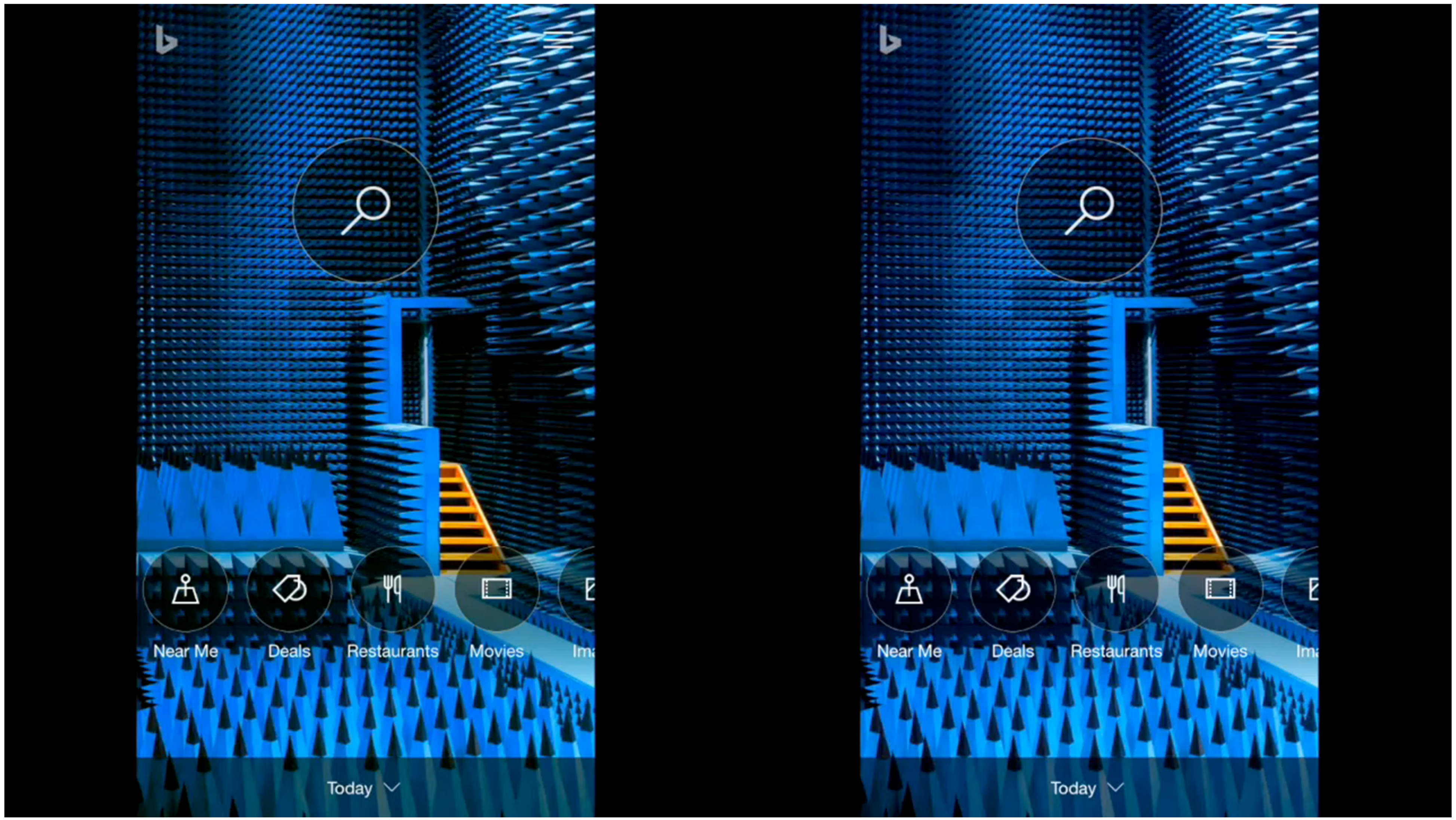
- App benefits: memory (but that can also be the problem…)
- Core ideas:
  - Progressive Rendering (JSON → render above the “fold” → render the rest)
  - **Pre-fetch and Pre-render content**
  - **Detect poor-network** and adjust based on it
  - “Perceived” performance (which is what users care….)

Content Prefetch - Design

Example-1: auto-suggest pre-fetch comparison

No pre-fetch

With pre-fetch



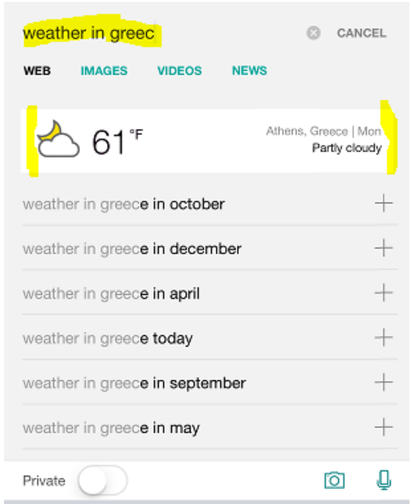
Example-2: related entities pre-fetch comparison

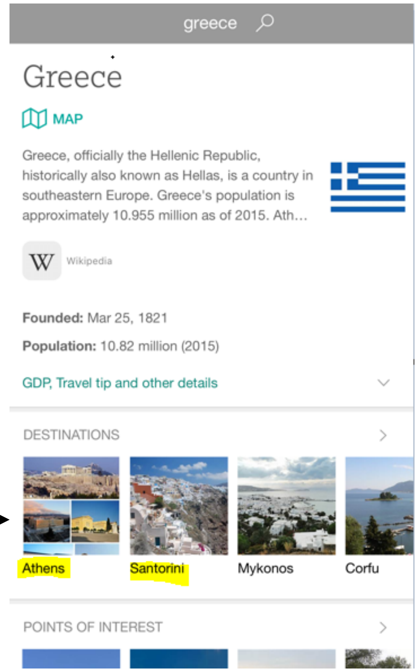
No pre-fetch

With pre-fetch

Feature Principles

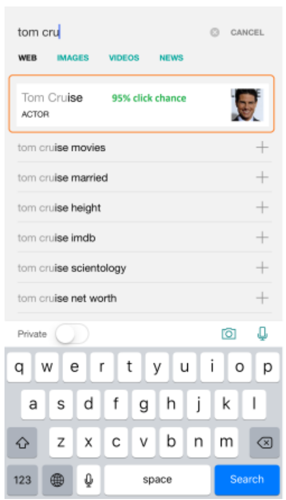
- 2 target scenarios
  - Answer Autosuggest
    - **High confidence** on user intent
  - Top 2 Related carousel's first entity
    - Chance of clicking on first entity of top 2 related entity carousel is high
      - **Over 96% probability** (for the carousel items)
    - Carousel count to pre-fetch = 2 (server side configurable)
- Can individually turn off different aspects
  - Autosuggest pre-fetch
  - Related entities pre-fetch
  - Entire pre-fetch feature

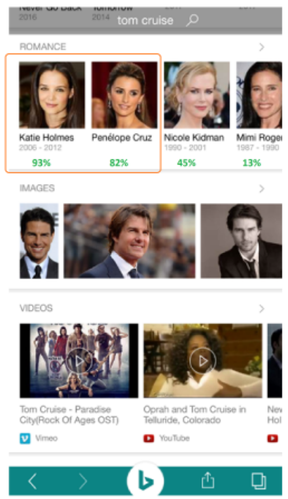




Pre-Fetch Principles

- Spin off a background thread to download (pre-fetched) content
- Determination of content to pre-fetch:
  - Network considerations:
    - Good network (heuristic)
    - WiFi (data plan consideration)
    - Not “in-private” mode (privacy consideration)
  - Scenarios where the user will be likely to engage (heuristic-based)
  - Server-configured





Pre-Fetch + Pre-Render

- Extra trick: pre-fetch and *pre-render*
- Pre-Render: set of Hidden WebViews (HWV)
- Upon pre-fetch, **content is pre-render in one of the HWV**
- On click, swap the views
- Pros:
  - Speed (immediate)
- Cons:
  - Memory pressure (potential “silent” crashes)
  - Technical nuances (such as “sound”)
  - Privacy (sharing cookies with “unseen” pages)

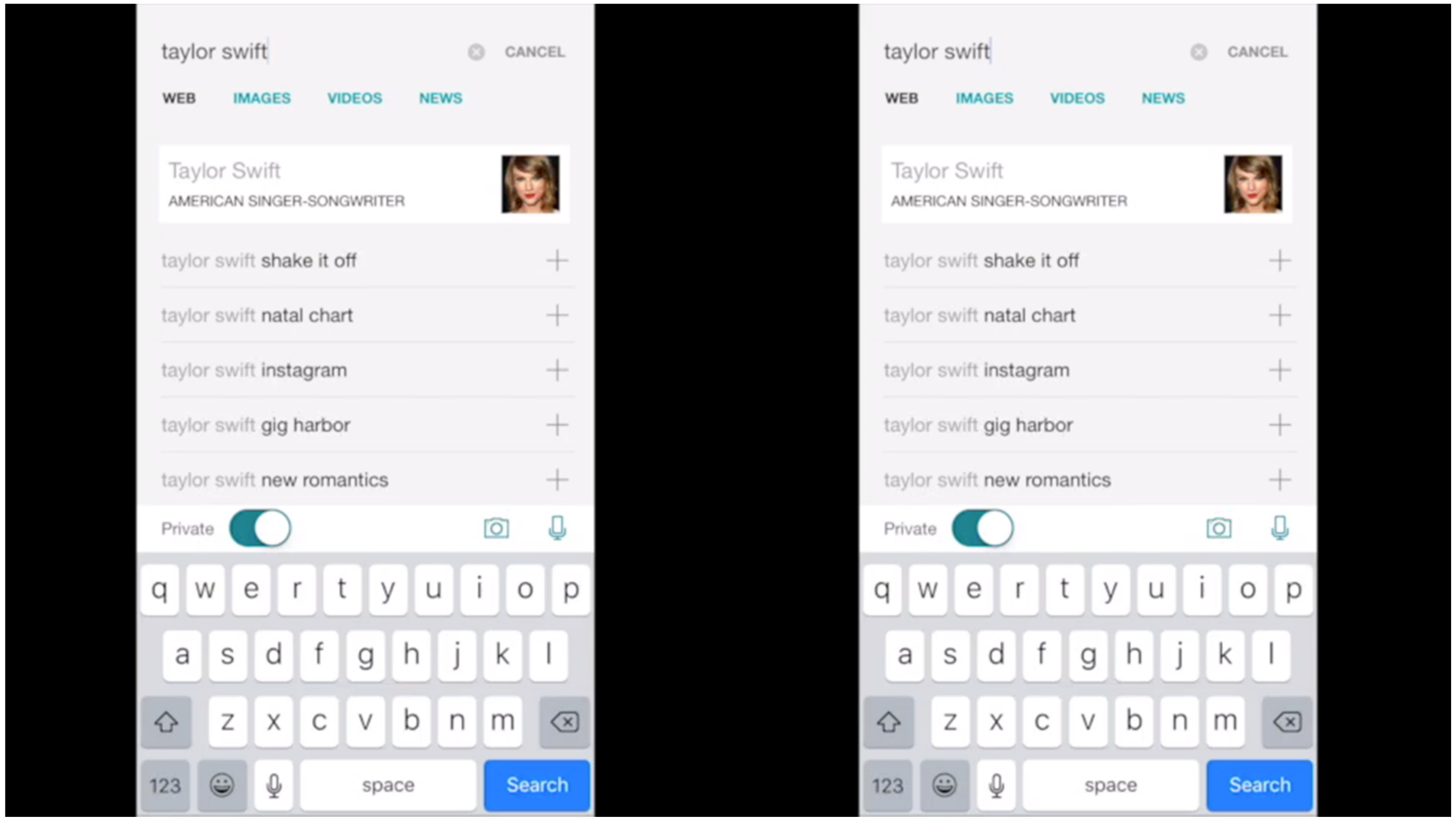
Detection and Optimizations for Poor Networks

Motivation

- iOS apps in general are **painful in slow/high-latency networks**
- These poor networks are everywhere
  - 3G with poor signal
  - Public WiFi (bus/coffee shop/airport)
  - Crowded environments
- If we could **detect our network speed** ...
- ... we could **tailor our actions** to improve the experience!
  - Use less data & decrease load times

Demo

Dynamic Quality On Dynamic Quality Off

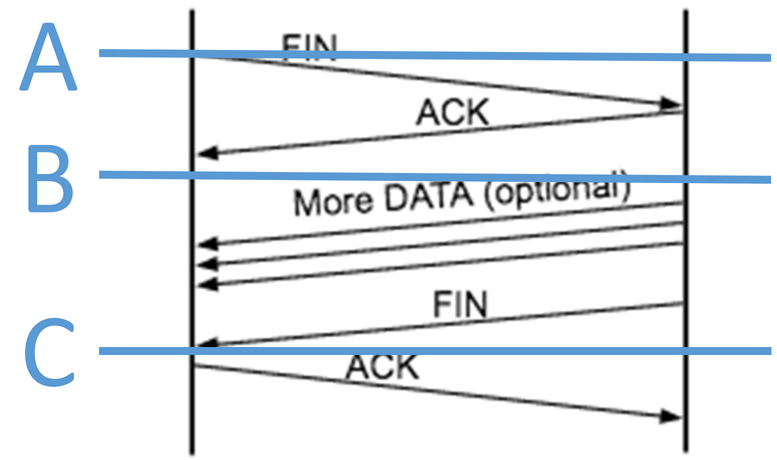


Core Concepts

- Goal: Detect all forms of slow connections
- Idea: Use past connections to predict future ones
- Requirement: Accomplish everything client side!
- How: Track latency, bandwidth, and failed connections
- How: Sliding window to smooth the detection

Detection for Poor Networks (iOS)

- A – dataTask.start()
- B – first call of didReceiveData()
- C – connection completed callback
- Latency is estimated by B – A
- Bandwidth is estimated by size / (C – B)
- Store connection data in circular queue



#	9	6	7	8
Latency	0.50s	1.02s	0.37s	0.78s
Bandwidth	131k	035k	208k	154k

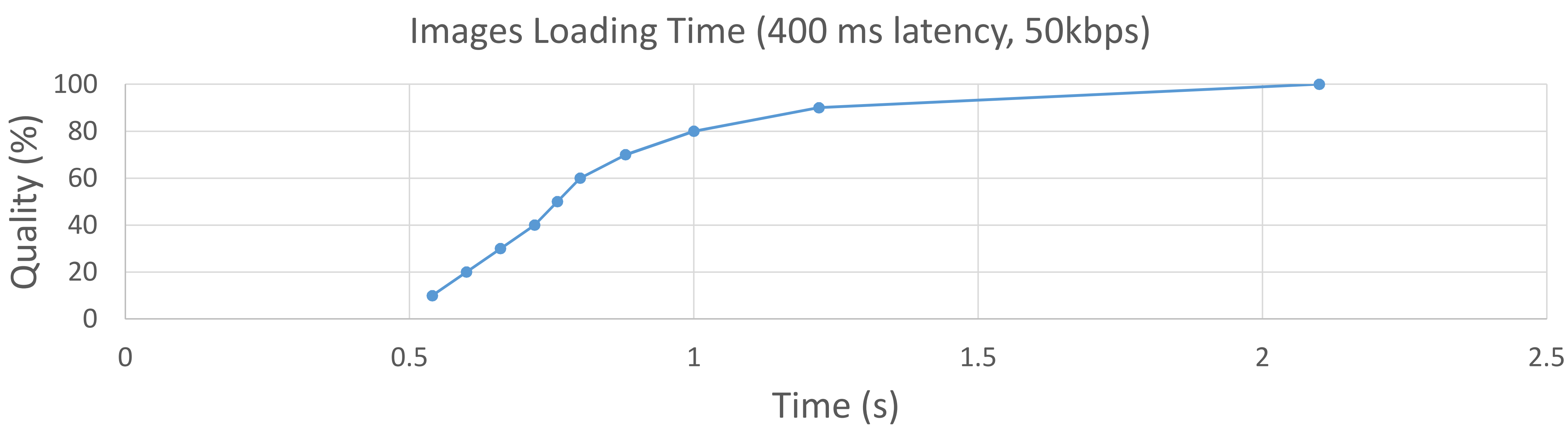
Avg Latency: 0.668s
Avg Bandwidth: 132k

Detection Considerations

- Existing solutions (FB's Network Connection Class for Android)
- Should reset on network changes (ex: 3G -> WiFi)
- Tune circular queue size
  - Too high -> fail to respond to changes
  - Too low -> susceptible to noise in network
- Latency dependent on connection setup
- Doesn't help cold queries
  - Pre-fetch/Config/Instrumentation can warm the queue!
- Handle caching gracefully (ignore cached content in the calculation)

Optimizations for Poor Networks

- Based on network status, **customize the actions**:
  - Reduce size of paginated entries (no pagination)
  - Use lower quality images (gradually improve the quality)
  - Change timeout/retry policy for requests (lower the bar)
  - Stop requests from auxiliary features (such as “auto-suggest”)

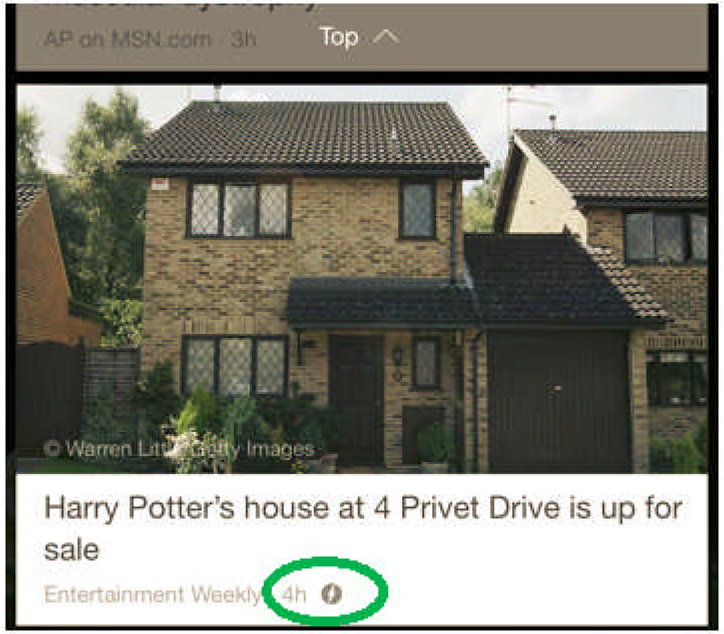


AMP Integration

AMP: Accelerated Mobile Pages

AMP: Accelerated Mobile Pages

- Subset of HTML heavily optimized for Mobile
  - Light: restriction on libraries
- Open Source (Google, Microsoft, Other Companies)
- 80% faster (based on Bing App data)
- 40% adoption (en-us, based on Bing App data)
- http://blogs.bing.com/search/September-2016/I amp-open-source-effort



In Summary…

- Apps ecosystem are being challenged, but it still in ascendance
- Fundamentals (availability, stability, agility, instrumentation) matters
- Performance is paramount – or else your app will be uninstalled…
- Machine Learning and AI must be part of your app
- Invest on discoverability, upselling and distribution models