Type Coercion in Watson

Leveraging Community-built Knowledge

Aditya Kalyanpur, Bill Murdock,

James Fan, Chris Welty

IBM Research

Who is Watson?

- Automatic Open-Domain Question Answering System

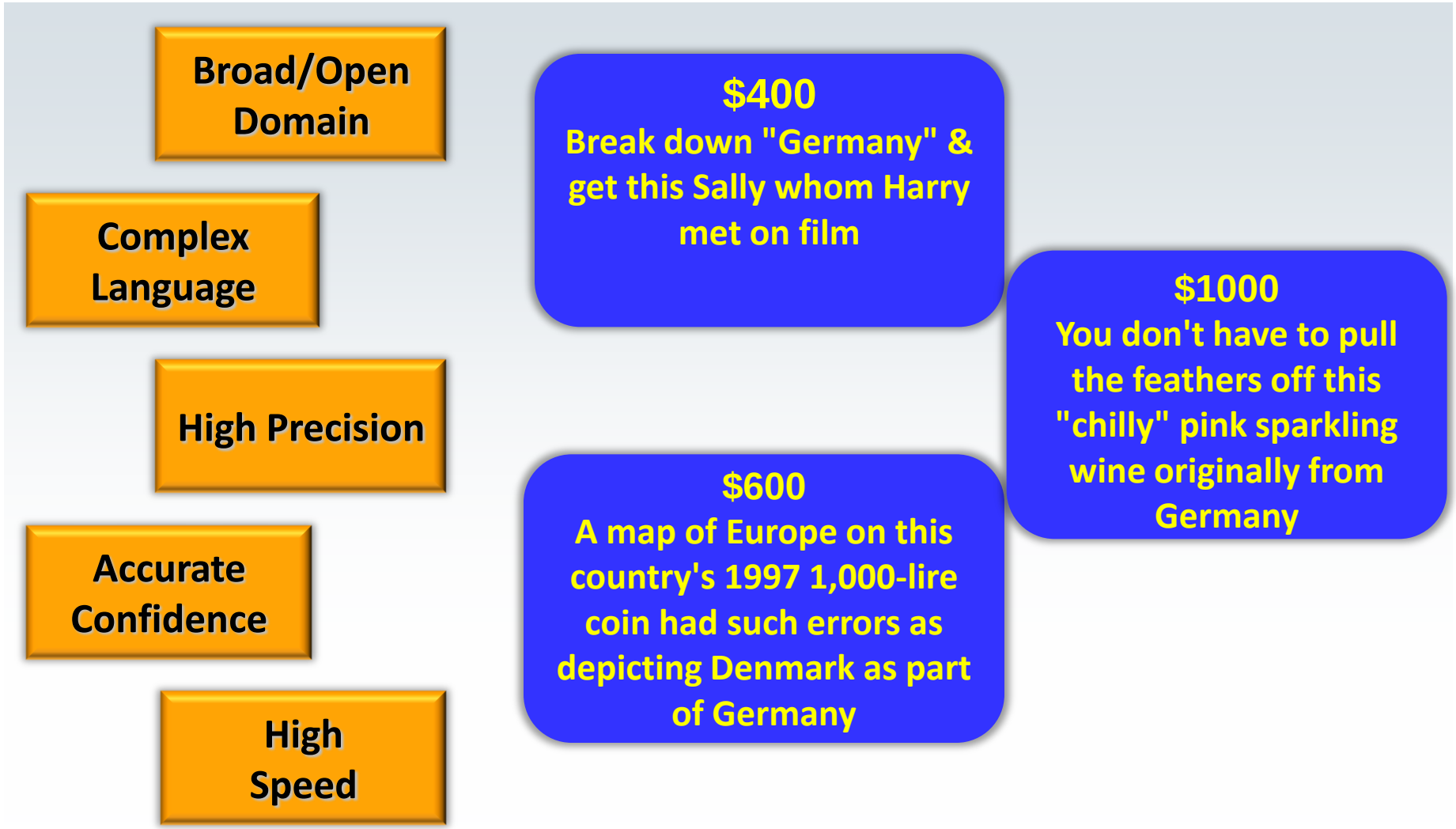
Webby “Person of the Year” 2011
(www.webbyawards.com)

- Given
  - Rich **Natural Language Questions**
  - Over a **Broad Domain of Knowledge**

- Deliver
  - **Precise Answers:** Determine what is being asked & give precise response
  - **Fast Response Time:** Results in few seconds
  - **Accurate Confidences:** Determine likelihood answer is correct
  - **Consumable Justifications:** Explain why the answer is right

The Jeopardy! Challenge

*A compelling and notable way to **drive** and **measure** the technology of **automatic Question Answering** along 5 Key Dimensions*



Typing in Jeopardy!

The *type* of thing being asked for is often indicated but can go from specific to very vague

- It's basically a big **kettle** with a close-fitting lid, used to cook pot roasts & stews
- Category: EUROPEAN NATIONALITIES
- Answer: **Dutch Oven**

- Unlucky things happen at Camp Crystal Lake in this 1980 **scarefest**
- Category: MOVIE CALENDAR
- Answer: **Friday the 13th**

- Wanted for general evil-ness; last seen at the Tower of Barad-Dur; it's a **giant eye**, folks. Kinda hard to miss
- Category: LITERARY CHARACTER APB
- Answer: **Sauron**

Closed Domain Type Checking

- Used in Traditional QA Systems

  Based on "Type And Generate" Principle

  - Focus on a pre-determined set of interesting types

    People, Places, Organizations, Dates

  - For these types, run Named Entity Recognizers (NER) over text corpus

    People: {"Einstein", "Sir I. Newton"..}

    Places: {"Germany", "UK"..}

    Dates: {"1885", "3rd April 1715"..}

  - At run-time, given a question, detect lexical answer type (LAT) and:

    Generate candidates from pre-compiled list of LAT instances

Limitations

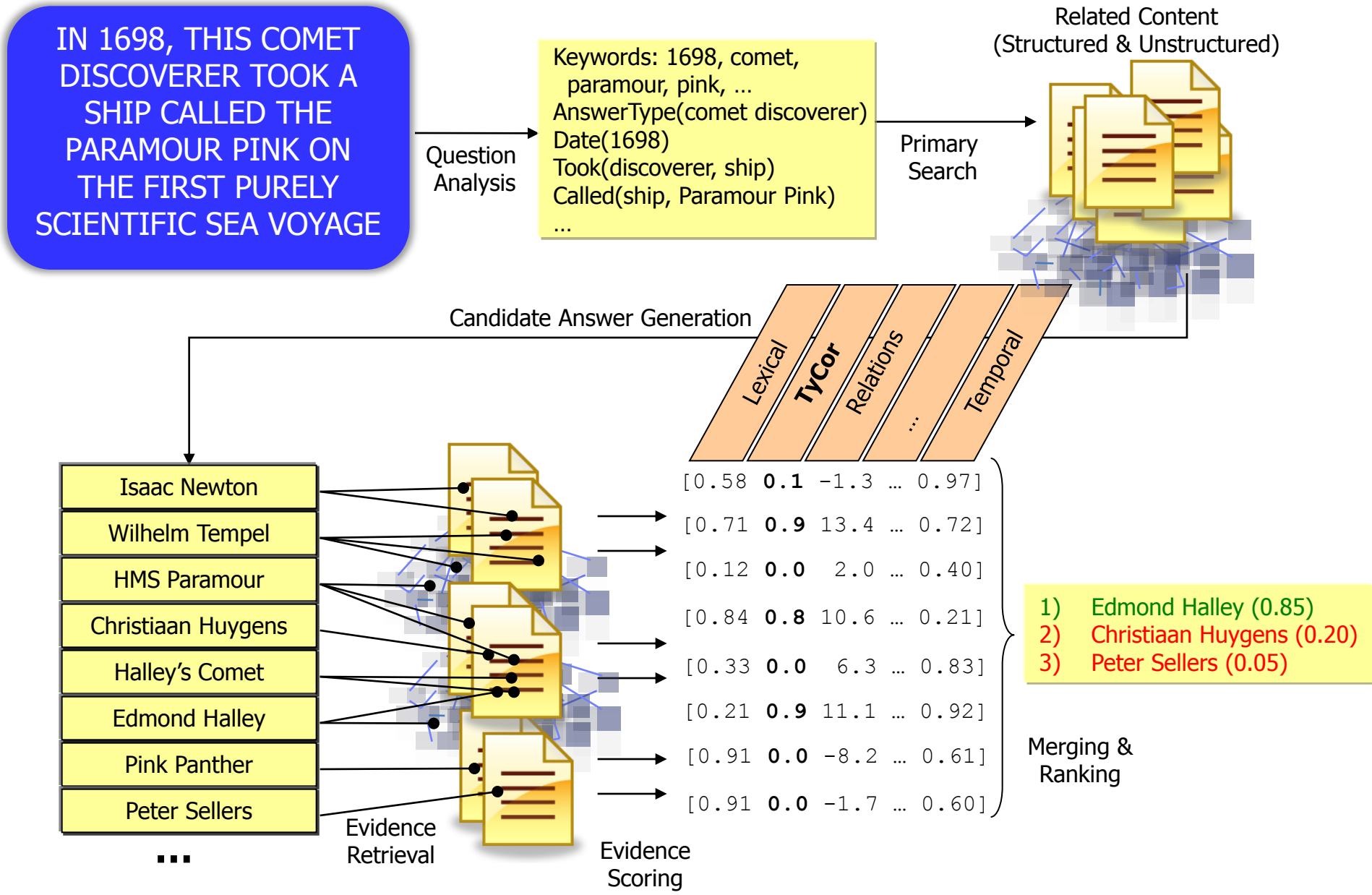
- Highly brittle – QA system breaks down if type not recognized
- Limited Coverage – need to enumerate all relevant types beforehand
- Dependent on quality of NERs used

Open Domain Type Coercion (TyCor)

- Approach taken in DeepQA
  - Based on "Generate-and-Type" Principle
    - Generate candidates without considering answer type (LAT)
    - Later check whether candidate can be *coerced* into LAT
    - Use a suite of Type-Coercion Algorithms
    - Use machine-learning to combine information from TyCors

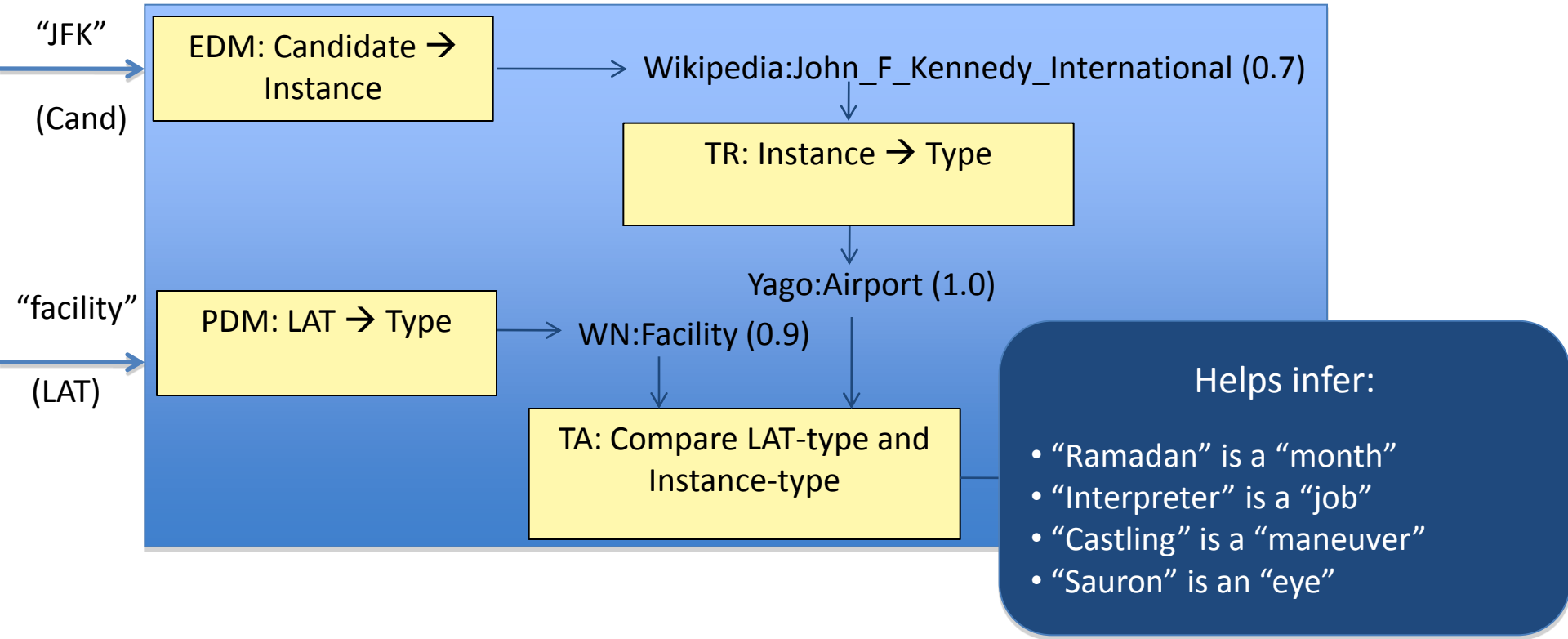
- Advantages
  - More flexible as QA system does not break down if type is not detected or meaningful
  - Much wider type coverage possible using a variety of sources and analytics for TyCor

How TyCor Fits in DeepQA



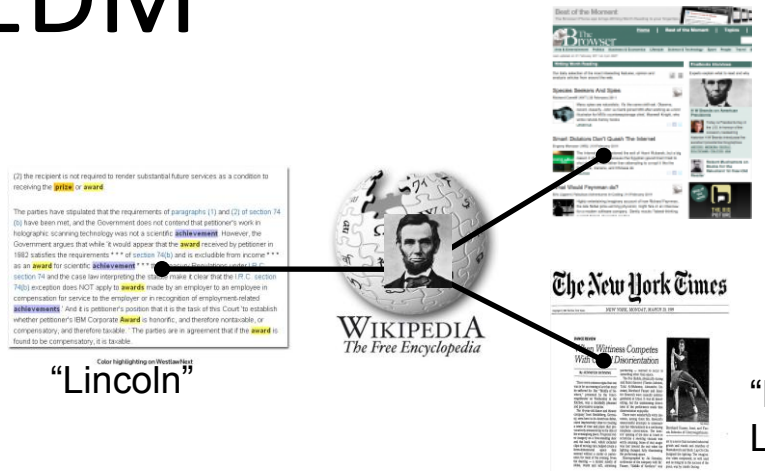
TyCor Framework

- **Problem**: Compute type match for candidate w.r.t. LAT
  - Both candidate and LAT expressed as Strings
- **4 Steps:**
  1. **EDM: Entity Disambiguation and Matching**
  2. **TR: Type Retrieval**
  3. **PDM: Predicate Disambiguation and Matching**
  4. **TA: Type Alignment**



EDM

Fundamental Task in NLP: Map entity string to meaningful reference



Issue 1: Synonymy
Many different ways to refer to the same entity (spellings, aliases, nicknames, abbreviations)


World All World Topics

Gaddafi? Kadafi? Qaddafi? What's the correct spelling?

You say, Gaddafi, we say Qaddafi. Other variations on the leader of Libya include "Gathafi," "Kadafi," and "Gadafy," creating an unholy mess for newspaper editors.

Issue 2: Polysemy
Sense Disambiguation depends on context

Flight took off from JFK…	JFK was assassinated…	Film critics loved JFK…

Using Community-built Knowledge in EDM

For Matching

- Wikipedia redirects (Myanmar ->> Burma)
- Synonyms / aliases extracted from WP Intro
  - "*IBM's distinctive culture and product branding has given it the nickname Big Blue*"
- DBPedia "name" labels (*firstName*, *lastName etc*…~100 props)

For Disambiguation

- Wikipedia Disambiguation Pages (wide coverage)
  - ~150K disambiguation pages in 2008
  - E.g. "Java" has >20 Distinct Types
- Measure similarity b/w sense text and entity context (using BOW, LSA etc)

Output: Ranked list of entity resources (Wikipedia URIs)

- Ranking based on: *Source, Popularity, Similarity*

Results

- Evaluation on Wikipedia: **Precision: 75%, Recall: 95%** (state-of-the-art)

Type Retrieval (TR)

- Obtain Types for Instances
- Sample Taxonomies Used In DeepQA:

- Interesting Points
  - Type Systems are linked
    - Yago → WordNet
  - Wiki-Categories and Lists contain extra information (modifiers)
    - Einstein : German-Inventor, Swiss-Vegetarian, Patent-Examiner
    - List of “German Cities”
  - Automatically Mined Types reflect real world usage
    - Fluid -is-a- Liquid (strictly speaking incorrect)

PDM

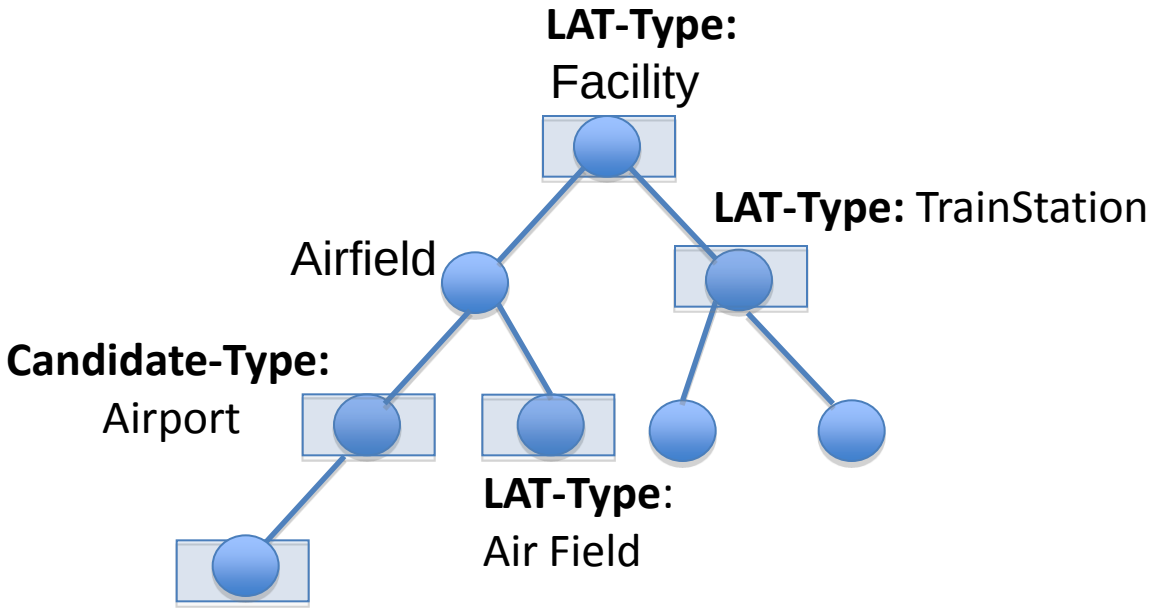
- Predicate (LAT) Disambiguation and Matching
  - LAT: star

In the northern hemisphere, latitude is equal to the angle above the horizon of this star, Alpha Ursae Minoris	***This star of "The Practice" played Clint Eastwood's Secret Service partner in the film "In the Line of Fire"***

- Similar in principle to EDM
  - EDM – map named entity → instance
  - PDM – map generic noun → class/type
- LATTE in DeepQA:
  - Map LAT to WordNet Concept(s): Order based on sense ranks
  - Pull in LAT Types that are **statistically related** in DBpedia
    - "Brand" → "Product" (0.83)

Type Alignment

- Type Matching Problem
  - Compare candidate types with LAT types
  - Produce a score depending degree of Match
- Various Types of Match Considered



Subclass Match (1.0)

Sibling Match (0.5)

Deep LCA Match (0.25)

Disjoint Types (-1.0)

Putting it all together

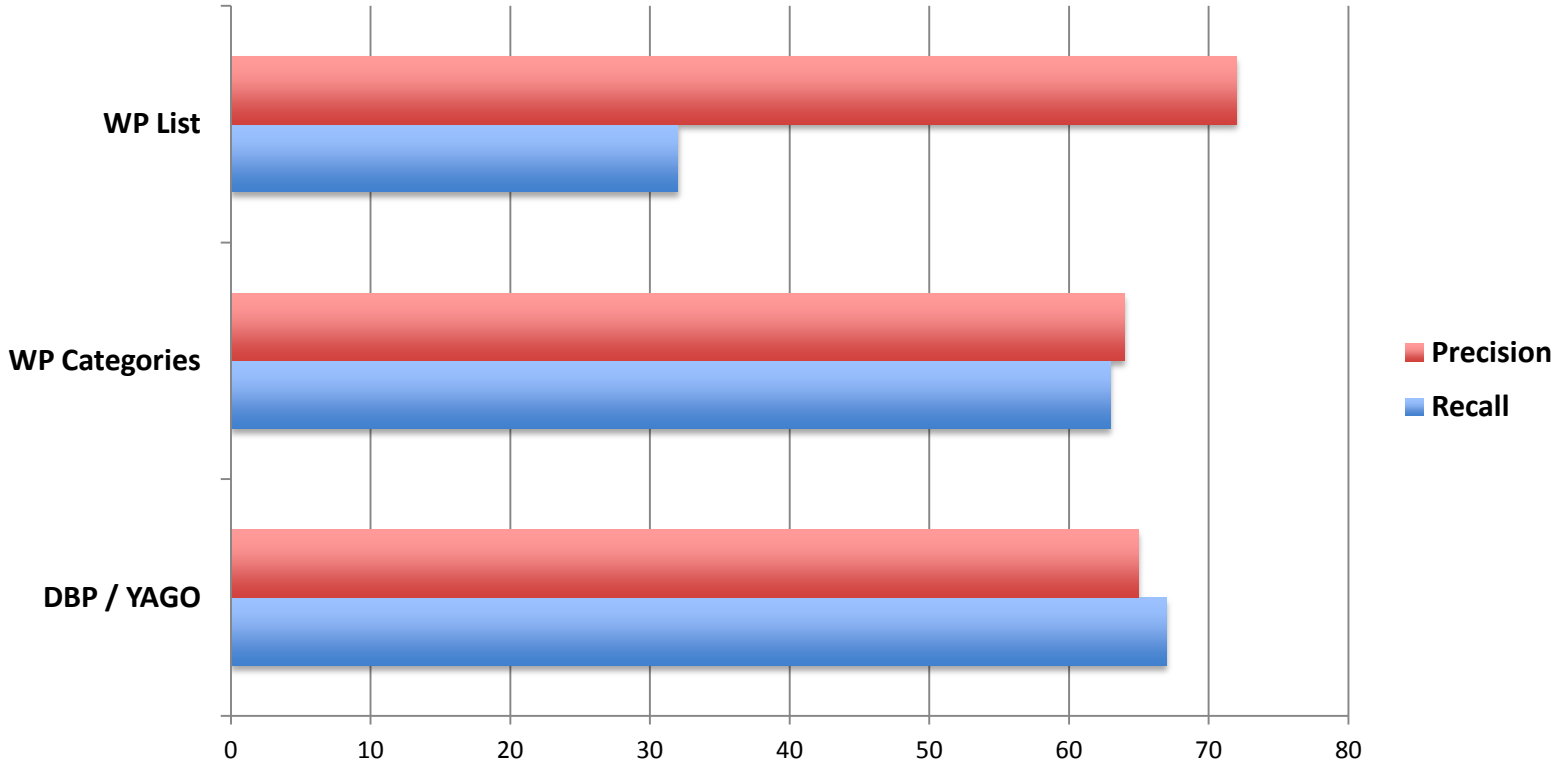
- TyCor Score = EDM * TR * PDM * TA
  - Intermediate Failure
    - If any step fails, Tycor Score = 0 (consider smoothing)
    - Expose which step failed to final model (EDM-Failure, PDM-Failure…)
  - **An-TyCor**
    - When TA score is -1 (Disjoint Types) → AnTyCor Feature added to model
    - Strong negative signal against candidate
    - Helps rules out candidates of wrong type (e.g. LAT: Country, Candidate: Einstein)
  - Multiple LATs
    - When multiple LATs in question with confidences: (L1, L2..Ln)
    - Final TyCor Score (weighted-sum) = (L1 * Tyc1) + (L2 * Tyc2) + .. (Ln * Tycn)

- TyCor Algorithm Suite in DeepQA
  - 14 TyCors Developed (3 that use Wikipedia and DBpedia)
  - All TyCors follow 4 key steps
  - Each TyCor score is a separate feature in model
  - Model learns weights on diff. TyCors: balances/combines type information

Evaluating TyCors on Ground Truth

Benchmark creation:

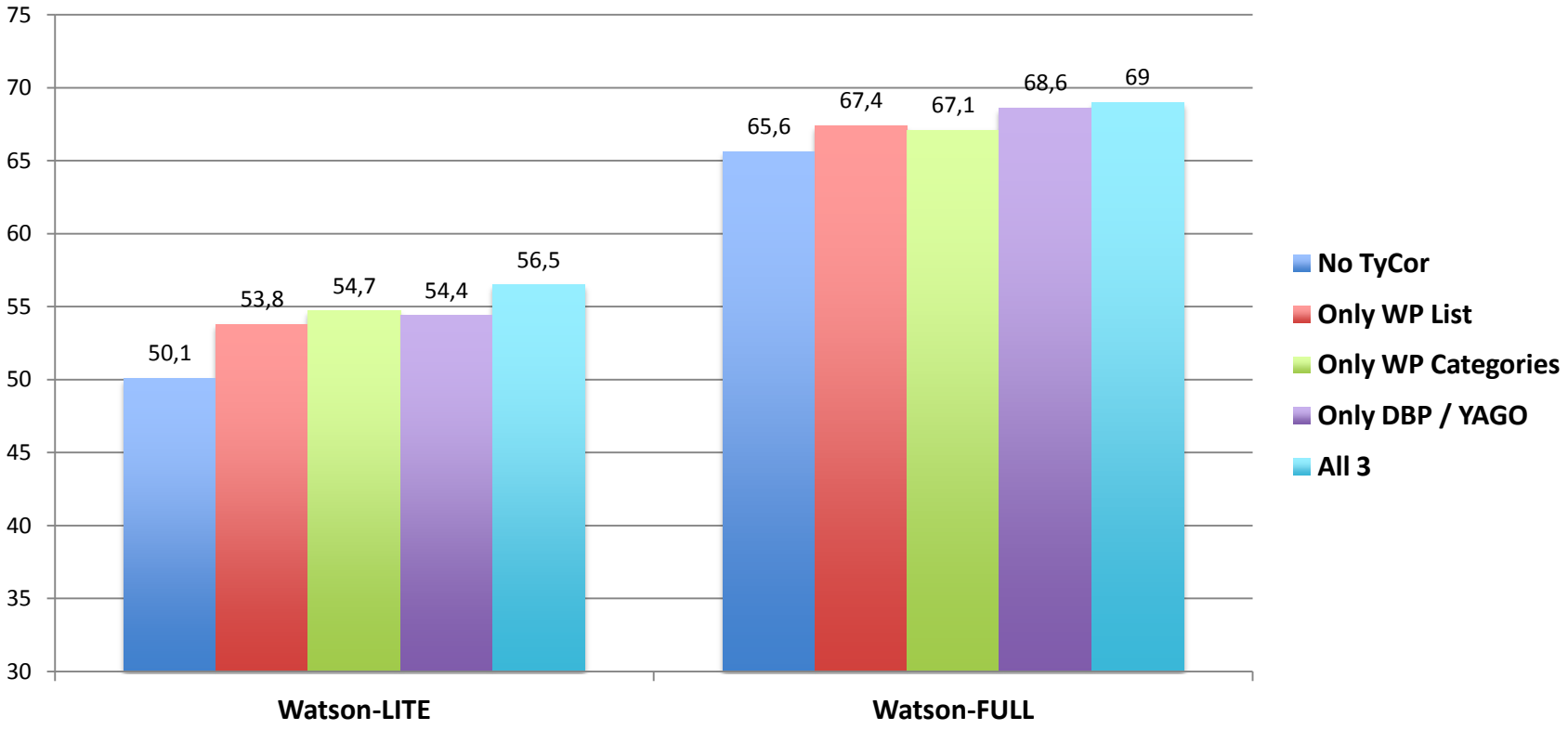
- Annotated Top 10 Candidates for 1615 Jeopardy! Questions
  - Judgement: Does candidate match LAT – Y/N?
- Total <LAT, Candidate> Pairs for Testing: 25,991 (due to multiple LATs)



Evaluating TyCors on end-to-end QA

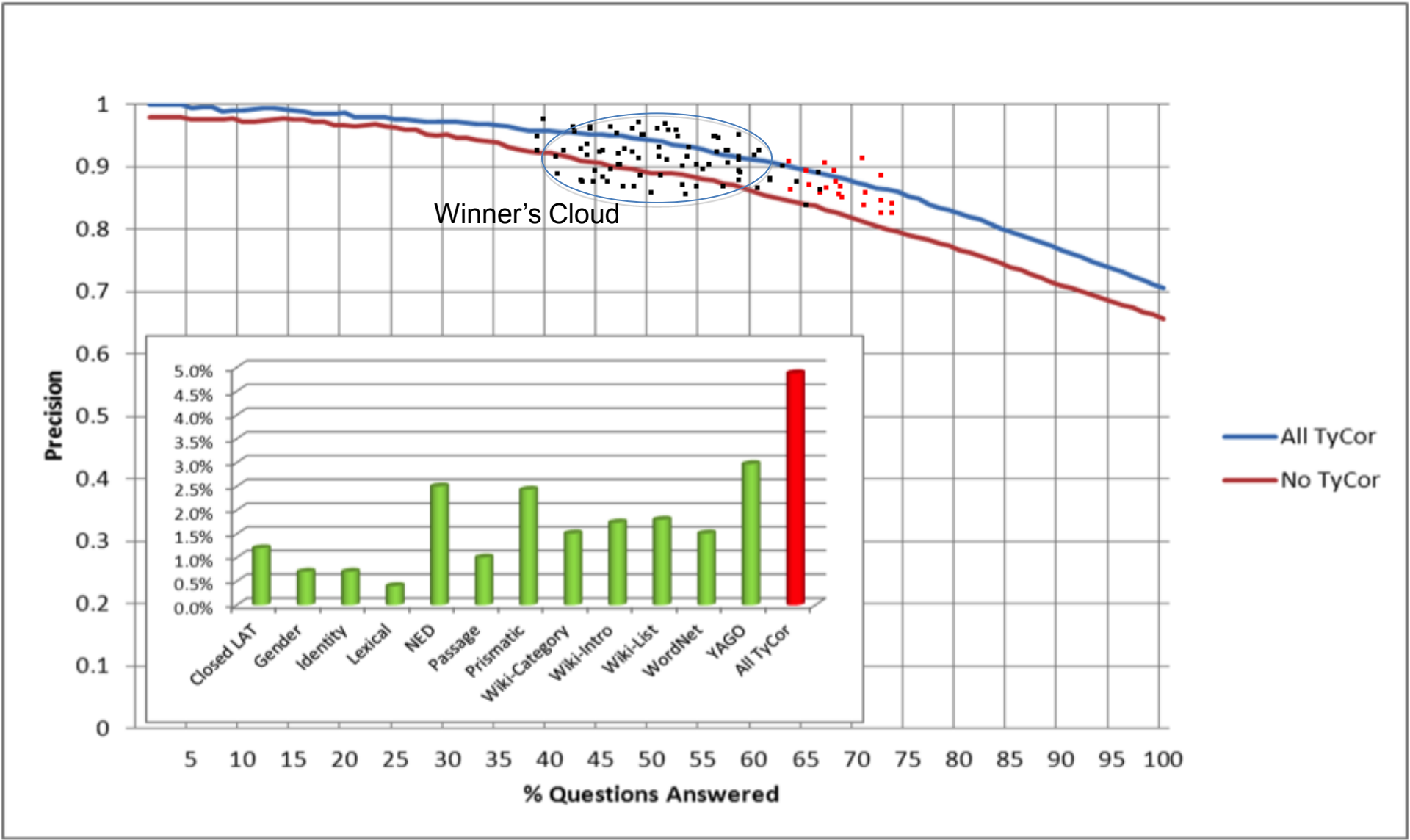
- Two Watson Configurations:
  1. **Watson-LITE**: Cand. Gen + Merging + Ranking (NO Answer Scoring)
  2. **Watson-FULL**: LITE + All Answer Scoring

- All gains over "No TyCor" are statistically significant
- Combining all 3 TyCors better than any one (Net gain: 5-6%)



Overall TyCor Impact

(Experiment done in Nov 2011)



Summary

THEORY

- **TyCor Framework provides flexible, robust answer typing**
  - **Core Idea: Treat type-match as *just another answer scoring feature***
  - Conceptual Separation of Steps: EDM, Type Retrieval, PDM, Type Alignment
  - Each step produces score reflecting uncertainty of mapping
  - Scores are features in ML model (with special features for failures)

IMPLEMENTATION

- **Community-built Knowledge useful in TyCor**
  - Scrape information from **Wikipedia**
    - Lists, Categories, Redirects, Anchor-Links, Intro-text
  - Map to **DBPedia**
    - Utilize Alternate names, Type Information, Links to YAGO / WN
  - Extend **YAGO** with Disjoints

APPLICATION

- **TyCor has significant impact in open-domain QA**
  - …and Watson won the Jeopardy! challenge
- **Beyond Jeopardy!: Watson MD**
  - Leverage UML-S and other Medical Ontologies in TyCor

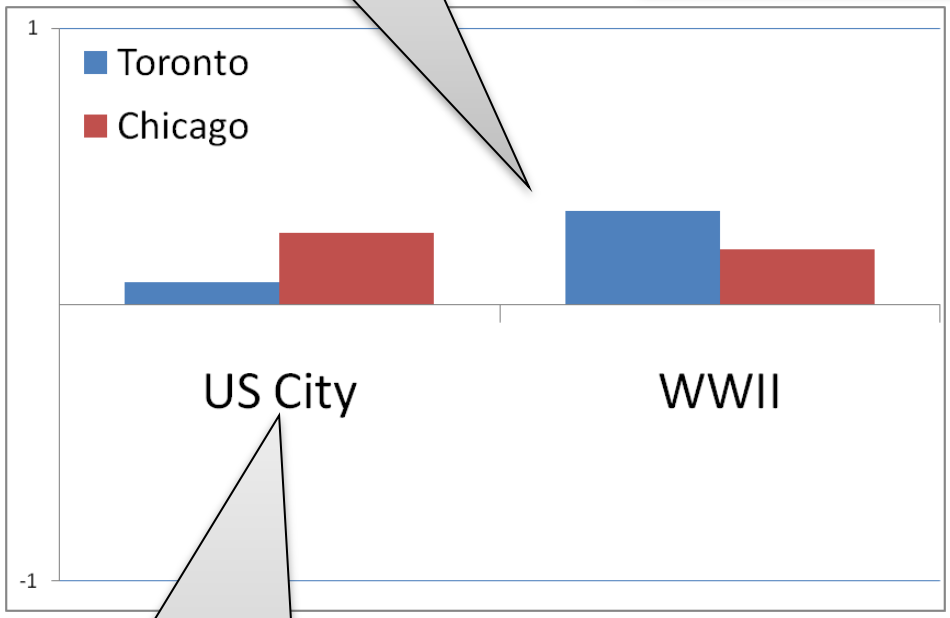
BACKUP

Toronto vs. Chicago

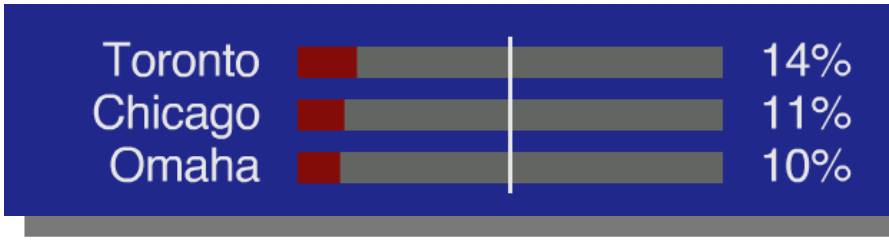
US CITIES

Its largest airport is named for a World War II hero; its second largest, for a World War II battle

Low because of weak evidence in content



Overall confidence was below threshold for both answers

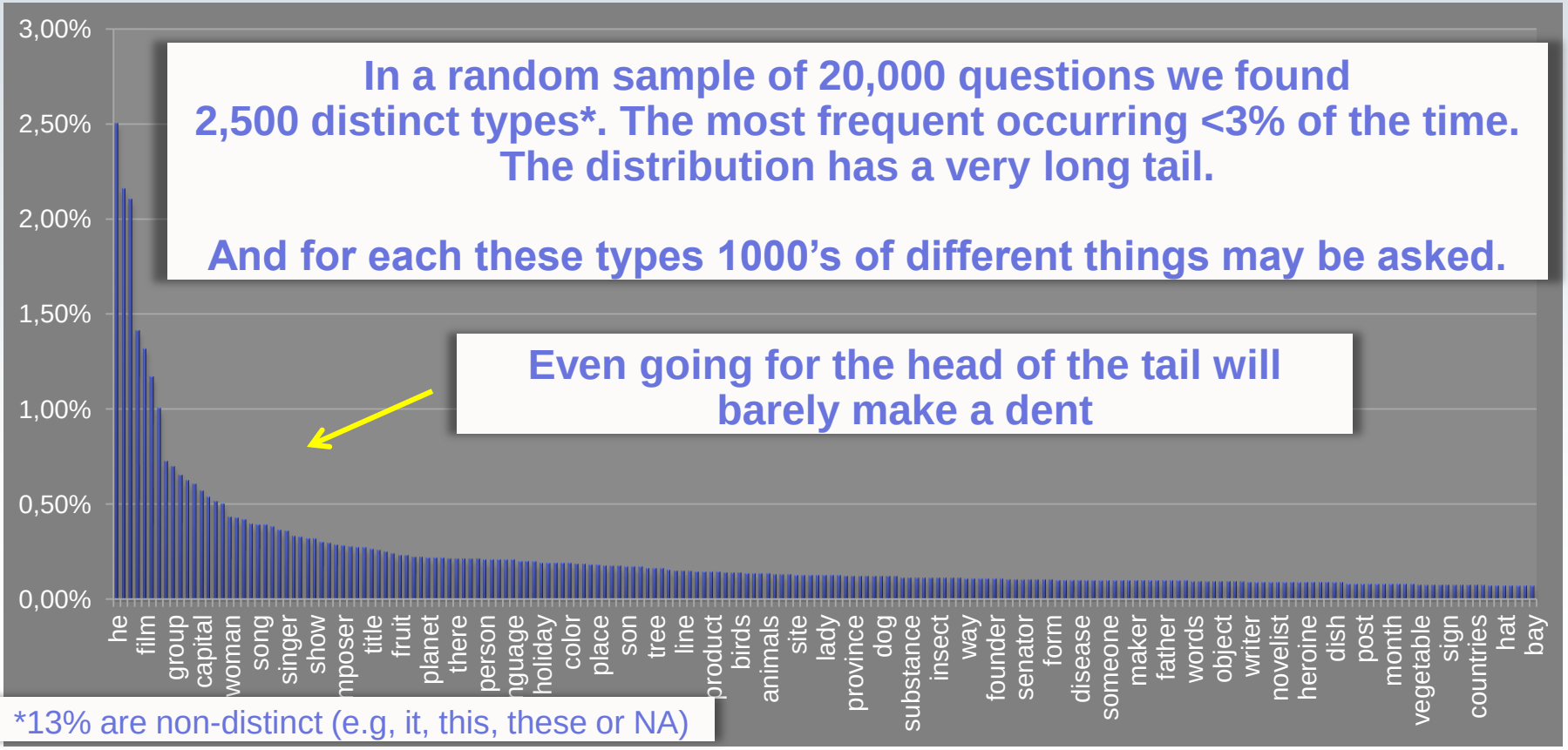


Low because being a **US City** is not a strong requirement simply based on Jeopardy! category

Lexical Answer Type (LAT) Distribution

We do NOT attempt to anticipate all questions and build databases.

We do NOT try to build a formal model of the world



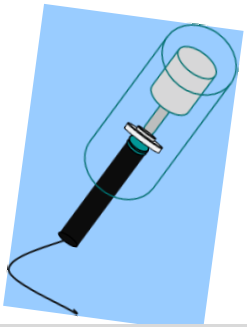
*13% are non-distinct (e.g, it, this, these or NA)

Our Focus is on reusable NLP technology for analyzing vast volumes of *as-is* text.
Structured sources (DBs and KBs) provide background knowledge for interpreting the text.

Acquiring Structured Data in Watson

- Obtain web-based (semi) structured resources
  - E.g. DBpedia, Yago, Wikipedia Categories, Redirects, Lists
- Process Raw Structured Data:
  - **Filter Noise**
    - Discard noisy Wikipedia Redirects
      - e.g. *Eliza Doolittle (character) -> Pygmalion (play)*
  - **Normalize Data**
    - Standardize temporal expressions
      - *"20th Jan 1950" -> "01-20-1950", "13th Century" -> "XX-XX-12XX"*
    - Normalize relation names
      - *{georss#lat, #latitude #geo-lat} - Latitude*
  - **Extend Ontologies**
    - Add disjoints – e.g. *Disjoint(Country, Person)* - Useful to rule out candidates with incompatible answer type

Watson's Buzz



As soon as the clue is read an **enable signal** does 3 things simultaneously

- ✓Activates the hand-held buzzers
- ✓Illuminates a visible light strip
- ✓Signals Watson

Equal Footing: Both Humans and Watson

- ✓Learn about the enable at the same time
- ✓Have to physically push down identical buzzers

Advantage Humans

By listening and anticipating the enable signal, humans can buzz in <5 ms
Watson is not hearing the host and cannot anticipate the enable signal

Advantage Watson

Watson, although not the fastest, is consistently fast
Assuming Watson can compute an answer and confidence in time (not always quick enough)
Watson does not risk the ¼ sec pre-buzz penalty – waits for enable

Watson uses a confidence-weighted buzzer scheme and will hesitate on less confidence answers to avoid "tipping and losing" to better players

Overall TyCor Impact

(Experiment done in Aug 2009)