Ten Years of Knowledge Harvesting: Lessons and Challenges

Gerhard Weikum[1], Johannes Hoffart[2], Fabian Suchanek[3]

[1] Max Planck Institute for Informatics Saarbrücken, Germany [2] Ambiverse GmbH Saarbrücken, Germany [3] Télécom ParisTech University Paris, France

E-mail: `weikum@mpi-inf.mpg.de`, `johannes@ambiverse.com`, `suchanek@telecom-paristech.fr`

Abstract

This article is a retrospective on the theme of knowledge harvesting: automatically constructing large high-quality knowledge bases from Internet sources. We draw on our experience in the Yago-Naga project over the last decade, but consider other projects as well. The article discusses lessons learned on the architecture of a knowledge harvesting system, and points out open challenges and research opportunities.

1 Large High-Quality Knowledge Bases

Turning Internet content, with its wealth of latent-value but noisy text and data sources, into crisp "machine knowledge" that can power intelligent applications is a long-standing goal of computer science. Over the last ten years, *knowledge harvesting* has made tremendous progress, leveraging advances in scalable information extraction and the availability of curated knowledge-sharing sources such as Wikipedia. Unlike the seminal projects on manually crafted knowledge bases and ontologies, like Cyc [28] and WordNet [15], knowledge harvesting is automated and operates at Web scale.

Automatically constructed knowledge bases – KB's for short – have become a powerful asset for search, analytics, recommendations, and data integration, with intensive use at big industrial stakeholders. Prominent examples are the Google Knowledge Graph, Facebook's Graph Search, Microsoft Satori as well as domain-specific knowledge bases in business, finance, life sciences, and more.

These achievements are rooted in academic research and community projects starting ten years ago, most notably, DBpedia [2], Freebase [6], KnowItAll [14], WikiTaxonomy [35] and Yago [42]. More recent major projects along these lines include BabelNet [32] ConceptNet [41], DeepDive [40], EntityCube (aka. Renlifang) [34], KnowledgeVault [10], Nell [7] Probase [51], Wikidata [48], XLore [49].

The largest of the KB's from these projects contain many millions of entities (i.e., people, places, products etc.) and billions of facts about them (i.e., attribute values and relationships with other entities). Moreover, entities are organized into a taxonomy of semantic classes, sometimes with hundred thousands of fine-grained types. All this is often represented in the form of *subject-predicate-object (SPO) triples*, following the RDF data model, and some of the KB's – most notably DBpedia – are central to the Web of Linked Open Data [19].

For illustration, here are some examples of SPO triples about Steve Jobs:

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

```
Steve_Jobs type entrepreneur           entrepreneur subtypeOf businessperson
Steve_Jobs coFounded Apple             Steve_Jobs hasFriend Steve_Wozniak
Steve_Jobs hasDaughter Lisa_Brennan    Apple_Lisa namedAfter Lisa_Brennan
Steve_Jobs diedOf Pancreatic_Cancer    Steve_Jobs fanOf Bob_Dylan
```

The most obvious use case for this kind of encyclopedic knowledge is to support search engines (for both Internet and enterprise search) on queries about entities. For example, when receiving the query "jobs apple", the system can identify "jobs" as a possible target entity and use the KB to improve its answers (transparently to the user, and in addition to pursuing other interpretations of the user's intent, e.g., looking for job offers at Apple). Other benefits arise from aggregating observations (on queries, clicks, posts, etc.) on a per-entity basis and combining them with semantic types or entity facts for analytics and recommendations.

Once computers have background knowledge on the real world, they have better ways of tapping into vague and noisy contents like natural language texts, social media and Internet data. This is an asset for language understanding, data cleaning and more. Moreover, we can think of this duality of knowledge acquistion and content understanding as a virtuous cycle: it enables the computer to obtain more knowledge, better knowledge, deeper knowledge. Figure 1 illustrates this point.

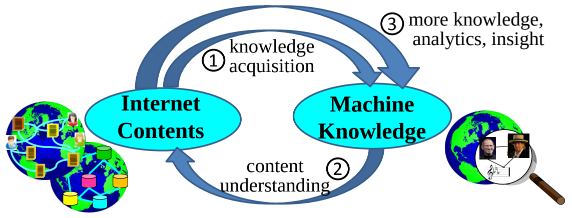

Figure 1: Knowledge Acquision and Content Understanding.

Our own endeavor on knowledge harvesting has its roots in research on semantic search starting in 2004. Later it became the *Yago-Naga project*, and led to the first release of the Yago KB (`yago-knowledge.org`) in February 2007. The salient strength of Yago is its rich type system with hundred thousands of fine-grained classes. When IBM Watson won the Jeopardy quiz show, it harnessed Yago's taxonomy for semantic type checking [26]. Later Yago releases added temporal and spatial knowledge [22], multilingual dimensions [38] and commonsense properties [17, 45]. Yago is now a joint project of the Max Planck Institute for Informatics and the Télécom ParisTech University. It is the only publicly available KB with statistical quality assurance: at least 95% accuracy (i.e., correct triples) based on sampling and Wilson confidence intervals.

This article intends to review the knowledge harvesting work of the past decade, pointing out lessons learned as well as open challenges and research opportunities. We draw on our Yago-Naga project as a primary source of experience, but aim to reflect the general research avenue.

2 Lessons Learned

2.1 Extraction Sources and Methods: Low-Hanging Fruit First

There is a wide spectrum of *information extraction (IE)* methods that knowledge harvesting can be based on, including regular expression matching, probabilistic graphical models, constraint reasoning and more. These can be applied to a wide spectrum of potential input sources, ranging from specialized databases (e.g., on movies or music) and web tables all the way to news articles and social media. From a purely scientific perspective, it may seem desirable to tap into every possible source with a single unified method and as little supervision as possible. Our experience is that this does not work - not if the goal is to build a high-quality knowledge base, with precision close to what a team of human curators would achieve.

Choice of Sources: We have followed – and strongly advocate – a layered approach where we pick low-hanging fruit first: high-quality input sources with limited noise in content structure, and robust methods for high-quality output. Specifically, we first tapped into semi-structured elements of Wikipedia: category names, infoboxes, lists, headings, etc. For specific kinds of knowledge, we integrated the best available curated sources: WordNet for taxonomic relations among semantic classes, and GeoNames for spatial entities. If we had been keen to increase Yago's coverage of movies and songs, we would have tapped into sources like IMDB (or LinkedMDB), Musicbrainz, etc. Generally, entities of specific types can often be harvested from dedicated sources or via specific identifier systems [44].

Once we obtained a KB core, we were able to harness this to distantly supervise extractions from other sources. In doing so, we could still "cherrypick" the more suitable sources: there is no point in obtaining poor extractions from super-noisy inputs like social media if the same knowledge can be distilled from easier inputs such as biographies. Also, we can leverage redundancy and statistics: seeing the same facts many times in different sources. This is why we called our approach *knowledge harvesting*, as opposed to running IE on each and every, arbitrarily difficult, input.

Extraction Methods: Similar engineering principles apply to the extraction methods. We started with simple, robust methods like regular expression matching (applied to token sequences in Wikipedia) combined with simple linguistic analysis, most notably, noun phrase parsing. Then we leveraged semantic type checking, building on the rich taxonomic knowledge distilled from Wikipedia categories and WordNet classes. This way we could keep precision at near-human quality, and were able to feed the obtained knowledge as seeds into more advanced distantly supervised methods: *reasoning with consistency constraints* over evidence-weighted candidate assertions [31, 43].

Our constraint reasoning is based on approximately solving Weighted MaxSat problems, which is equivalent to MAP inference for probabilistic graphical models. We believe that explicit constraint reasoning can be tuned more easily, though. A key aspect is the manual crafting of consistency constraints: functional dependencies, inclusion dependencies, temporal constraints, and more. For example, we can specify that a person (such as `Lisa_Brennan`) can have only one father, and that a person can found a company or compose a song only while being alive (i.e., between birth and death dates). Although this approach requires a modest amount of human supervision, we never encountered this to be a bottleneck.

Other papers with similar lessons include [8, 9, 30, 54].

2.2 Data Representation: Triples, Triploids, Quads, and Beyond

Like other KB projects, we decided to represent facts in the form of SPO triples, following the RDF data model.

Canonicalization: An important design decision, not shared by all of the major KB's, is to aim for database-style rigor in capturing the S, P and O roles of triples. We wanted S to be only *canonicalized entities* (or classes when P is `subtypeOf`), P to be only explicitly specified relations (with type signatures), and O to be only entities or literals such as dates or numbers (or classes for `type` or `subtypeOf`). Here, canonicalization means that we can uniquely identify each entity, and that all facts that refer to the same entity are attached to the same S value, regardless of the surface names under which the entity is discovered. For example, we insist that the two sentences *"Jobs is one of the co-founders of Apple."* and *"Steve founded Apple, together with his friend Woz."* result in the same fact `Steve_Jobs coFounded Apple`. Conversely, we have to carefully distinguish occurrences of "Woz" meaning Steve Wozniak against occurrences that abbreviate the game "Wizard of Oz". This issue calls for *named entity disambiguation* (aka. entity linking) [39], which we initially integrated into constraint reasoning [43] while later developing a general-purpose stand-alone solution [21]. Occurrences of "Steve" that our methods cannot map to a unique entity with high confidence would be disregarded.

Triples vs. Triploids: Our design emphasizes *precision* at the expense of *recall*. Several other KB's show facts with different S values even if they refer to the same entity, such as `Jobs founded Apple`, `Steve coFounded Apple`, `SteveJobs coFounded Apple`, `Woz coFounderOf Apple`, and so on. Here, the P values are not properly canonicalized either. Unlike some KB projects that see this as an advantage, resulting in a larger KB, we believe that noisy redundancy and ambiguity is the recipe for inconsistency and degraded quality in downstream applications.

In retrospect, we still stand by this design choice, but we would be open to *additionally* harvesting non-canonicalized assertions, using so-called Open IE methods [29]. We would then treat S, P and O as textual phrases, to explicitly distinguish them from canonicalized triples. As this is no longer within the RDF standard, we refer to this case as *triploids*. Here are some examples, including mixed cases with partial canonicalization:

```
"Steve" "revolutionized" "music industry"    "Steve" invented iPod
Steve_Jobs "dated" Joan_Baez                 Steve_Jobs "admired" "Dylan"
```

The rationale for tolerating the co-existence of triples and triploids is to increase the coverage of the KB and support more use cases. Search applications could work well even with triploids, whereas other cases require rigorous reasoning and could thus be restricted to proper triples. We call the hybrid representation an *Extended Knowledge Graph (XKG)* [53], essentially a richly labeled graph with canonicalized or textual labels for nodes and edges. An XKG could be incrementally turned into a KG in a pay-as-you-go manner, using methods for post-hoc canonicalization of entity names and predicate phrases [16].

Beyond Triples: It is good practice to associate data with its provenance. To this end, we attached to each SPO triple metadata about the extraction source, method and confidence. As there could be multiple sources, we refer to this aspect as "knowledge witnesses". How do we represent this? The Semantic Web community advocates so-called *quads* for this purpose: adding a fourth dimension to each triple which encodes its provenance. However, this is insufficient as we need to capture multiple witnesses, extraction dates, confidence scores, etc. So we used a technique akin to reification: each SPO triple is given an identifier, and these identifiers can in turn be used as S values in additional (metadata) triples.

Following the idea of the virtuous cycle in Figure 1, we also harvested *spatial* and *temporal knowledge* about (certain kinds of) SPO triples [22]. For example, for a triple like `Steve_Jobs isCEOof Apple`, we distill the respective time intervals for the validity of this fact from additional sources [18, 50]. Likewise, events can be positioned in space and time, e.g., by capturing that `Steve_Jobs announced iPhone` happened on January 9, 2007 in San Francisco. This is represented in the KB as follows (with identifiers prefixing the triples):

```
id1:  ( Steve_Jobs isCEOof Apple )
id2:  ( id1 validDuring [1997-07-09, 2011-08-24] )
id3:  ( Steve_Jobs announced iPhone )
id4:  ( id3 happenedOn 2007-01-09 )     id5:  ( id3 happenedIn San_Francisco )
```

While this representation may appear elegant, it is unwieldy for querying. Simple queries such as asking for iPhone-related events in 2007 become fairly complex. We even invented extensions to the RDF query language Sparql to express search conditions with fact identifiers:

```
Select ?x, ?y Where {
     ?id:  ?x ?y iPhone .  ?id happendOn 2007-##-## .  }
```

SPOTLX Tuples: These considerations made us rethink our choice for RDF, and eventually led to the model of *SPOTLX tuples* for the Yago2 release in 2011 [22]. Each fact was expressed as a six-tuple with Subject, Predicate, Object, Time, Location and teXt (or conteXt), plus additional metadata attributes. The X component allows textual witnesses for facts, which could be queried jointly with the facts in the KB. For example, to find iPhone-related events in 2007 in the Bay Area which involved "standing ovations", we could use SQL over SPOTLX tuples, including abstract data types for time, space and text. The RDF representation has still been kept in parallel, for easy data exchange and interoperability in the Linked Open Data world.

2.3 Data Storage and Query Processing: Join, Relax, Rank and Scale

Since we initially focused on SPO triples, we desired efficient support for the Sparql query language. Because of the fine-grained nature of RDF data, this calls for extensive join processing, a typical example being:

```
Select ?x Where {
     ?x bornIn ?t .  ?t inCountry ?c .  ?c locatedIn Europe .
     ?x performed ?s .  ?s type song .  ?s composedBy Bob_Dylan .  }
```

This query finds European artists who covered Bob Dylan. It consists of 6 triple patterns: a 6-way self-join over the schema-free SPO table. As there was no high-performance RDF engine at that time, we built our own: the RDF-3X system with emphasis on join optimization [33].

This served us well for some time, but the transition from triples to quads and SPOTLX tuples (see above) led us to move to a standard relational engine, namely, PostgreSQL. However, as our KB grows and we pursue new KB-driven applications, we reconsider this decision. Our recent endeavor to support entity-centric large-scale text analytics [24], employs a cloud-based platform using Spark with Cassandra for storing the KB. This key-value storage solution gives full flexibility to represent knowledge tuples, decent query processing performance, and an easy way to scale out. However, this platform is far from ideal for the performance of many-way joins and SPOTLX queries. So we may have to keep revisiting this design choice.

Ranking of Query Answers: In addition to query performance, an important concern for us has been the need to rank query answers and to support query relaxation. These requirements arise as users who explore the KB are not familiar with its structure, terminology and content. Broad queries return many answers; so we need ranking to identify the most informative ones. An example is the query about iPhone-related events given above. With KB's being part of the highly heterogeneous Linked-Data ecosystem, this issue becomes even more demanding. Therefore, we developed IR-style statistical ranking models for query answers from triple patterns and complex Sparql queries [12, 27].

Query Relaxation: Even with perfect answer ranking, querying a KB still poses a high burden even for skilled users like analysts or journalists, due to the potential mismatch between the user's and the KB's vocabulary and structure. Reconsider the query about European artists covering Bob Dylan. Albeit perfectly formulated, it may still return very few answers or none at all, simply because the predicates in the query – `inCountry`, `performed`, `composedBy` – could be sparsely populated. Perhaps, cities are more often in the `locatedIn` relation which should better be transitively applied. Songs may better be found by using two predicates `hasAlbum` between artists and albums, and `hasTracks` between albums and songs. Instead of the `composedBy` predicate, the inverse relation `composed` between artists and songs may be the preferred way when populating the KB. Or perhaps neither of the two predicates is much in use; the KB could instead have triploids with P phrases like "composer of", "created", "his masterpiece".

This suggests alternative query formulations, which we call *query relaxations*. A good search interface should automatically generate one or more relaxations as needed. However, as we deviate from the user's original formulation, answers may be treated with lower confidence, and we have to merge results from multiple relaxations – another case for answer ranking.

Over the last ten years, our understanding for these issues in KB search and exploration has gradually improved. The form-based interface of our recent Trinity system [53] supports such relaxations and rankings. [4] gives a comprehensive survey on semantic search.

2.4 Data Evolution: The Knowledge Awakens

Hardly anything lasts forever. In a KB, attribute values (e.g., city populations) and relationships of entities (e.g., the CEO of a company or the spouse of a person) change over time. Even the set of entities under consideration is not fixed: new entities are being created all the time (e.g., new songs, sports matches, newborn children of celebrities) and need to be added to the KB. Also, existing entities could be irrelevant for a KB, but become prominent at a certain point. Examples are when an unknown "garage band" or "garage company" starts having success. If Wikipedia had existed in April 1976, it would probably not have included Apple for insufficient notability.

So a KB should be continuously updated, for example, by subscribing to change feeds from Wikipedia and other data streams. DBpedia tried this [20], but abandoned it for its complexity. Yago instead used an approach with periodic rebuilding of the entire KB. Freebase and Wikidata have update processes in place, but seem to critically rely on human curation.

One difficulty that prevents a straightforward solution is that sometimes new facts can be simply added to the KB whereas others need to invalidate and overwrite previously included facts. Comprehensive versioning of all triples or tuples would alleviate this issue, but comes at the cost of making querying and exploration more complex. The general data evolution problem for KB's still appears to be widely open.

Active Knowledge: For highly dynamic and specialized knowledge, the Yago-Naga project explored an approach for linking KB items with external databases and web services. For example, the chart positions of a song and the box office counts of a movie change so rapidly that it is hardly meaningful to materialize these values in the KB. Instead, one should have automatic linking across knowledge repositories and to web service calls that return up-to-date values on demand. We developed techniques towards this form of *active knowledge* [36]; more work is needed along these lines.

Emerging Entities: One aspect of knowledge evolution for which we have a reasonable success story is *emerging entities* [23]. When identifying entity names in input sources (text, web tables, etc.), we attempt to disambiguate them onto the already known entities in the KB. However, we always consider an additional virtual candidate: none of the known entities. When the evidence and our methods suggest that we observe an *out-of-KB entity*, we capture it under its surface name along with its context. After some time, we obtain a repository of such emerging entities. Then we run clustering techniques to group them, and involve users to confirm this canonicalization. Finally, the emerging entities with sufficient support and confirmation can be added to the KB as first-class citizens. To alleviate humans from labor-intensive curation, it is important to present new entity candidates with informative context [25].

A specific case for out-of-KB entities is constructing ad-hoc KB's on the fly. Consider a new corpus of documents becoming available, for example, the Panama Papers or a batch of articles on specific health issues such as Zika infections. The goal is to automatically build a domain-specific KB that helps journalists and analysts to obtain an overview and drill down into finer issues. Our startup `ambiverse.com` pursues applications along these lines [13, 1].

3 Challenges and Opportunities

3.1 Knowledge Base Coverage

Some of the existing KB's are huge, but no KB will ever be complete. In fact, one can observe all kinds of knowledge gaps.

Locally incomplete knowledge is when certain O values are missing for a given S and P value – for example, when we know some movies of a director but not all of them. A variant of this is when for a given P value, we have O values for some S but not all of them – for example, knowing spouses of some people but missing out on many other married people. The difficulty here is not just to fill these gaps, but to realize when and where gaps exist. In other words, when can we assume a locally closed world, and how can we find evidence for an open world that can provide additional facts? [37] offers further discussion along these lines.

A second challenge arises from **long-tail entities** and **long-tail classes**. There are many lesser known musicians, regional politicians and good but not exactly famous scientists. How can we identify these in the Internet, and harvest facts about them? Long-tail classes pose a similar problem: despite some KB's having hundred thousands of classes, one could always add more interesting ones. For example, what if we got interested in a class of `GratefulDeadFans` (i.e., fans of the rock band, which would have Steve Jobs as a member) or `HippieRetroConcerts`? Where in the class taxonomy do we fit these in, and how can we find their members (which is difficult even among the entities in the KB)? [11] offers further discussion along these lines.

Third and last, there is a big deficit in terms of **missing salient facts** about entities. KB's have been built in an opportunistic manner, mostly relying on Wikipedia. If Wikipedia does not have the information or if a fact is stated only in sophisticated form in the article's text, all the KB's miss out on it. For example, what is notable about Johnny Cash, the late singer? One key fact is that he recorded a live album on a free concert in a US state prison – in 1968 when this was a sensation if not a scandal. KB's contain the name of the album, but not the special circumstances. What is notable about the Nick Cave album "Abbatoir Blues"? Many KB's contain this album, listing its individual songs. No KB points out, though, that the song "Let the Bells Ring" is about Johnny Cash and that the song "O Children" is used in one of the Harry Potter movies. Part of the problem is the poor coverage of predicate types: KB's are missing predicates like `songIsAbout`. But this alone cannot fix the issue with the Folsom Prison concert. There is a great research opportunity here. Similar points can be made for *spatial knowledge* and *temporal knowledge*.

One may argue that Open IE could fill these gaps, and one should add more textual descriptions to an Extended Knowledge Graph. This would be a remedy, but only in a superficial sense as humans would then have to read and interpret a large amount of noisy text to look up facts. The fundamental trade-off between precision and recall cannot be eliminated this way.

3.2 Commonsense, Rules and Socio-Cultural Knowledge

Automatically constructed KB's have largely focused on harvesting encyclopedic fact knowledge. However, to power semantic search and other intelligent applications (e.g., smart conversational bots in social media), computers need a much broader understanding of the world: properties of everyday objects, human activities, plausibility invariants and more. This ambitious goal suggests several research directions.

We need to distill **commonsense** from Internet sources. This is about properties of objects like size, color, shape, parts or substance of which an object is made of, etc., and knowledge on which objects are used for which activities as well as when and where certain activities typically happen. For example, a rock concert involves musicians, instruments – almost always including drums and guitars, speakers, a microphone for the singer; the typical location is a stage – often but not necessarily in a large hall, and so on. This background knowledge could improve the interpretation of (spoken) user questions, and also image and video search when user queries include abstractions or emotions that cannot be directly matched by captions, tags or other text. Today's search engines return poor results on queries such as "exhausted band at hippie concert" (where the user hopes to find footage of performances by the Grateful Dead or the Doors). Prior work on acquiring commonsense includes ConceptNet [41] and our recent project WebChild [45, 46]. However, there is still a long way to go for computers to learn what every child knows.

Rules that capture invariants over certain kinds of facts is another key element for advancing the intelligent behavior of computer systems. For example, a rule about scientists and their advisors could be that the advisor is a professor at the scientist's alma mater – as of the time when the scientist graduated. Such rules may have exceptions, but could be a great asset to answer more queries and to infer additional facts for "KB completion". Even soft rules can be useful: an example could be that folk guitarists are usually also singers. To acquire this intensional knowledge, rule mining methods have been developed and applied to large KB's – one notable work being our AMIE project [17]. However, the state of the art exhibits major limitations. Rules are restricted in their logical form to Horn clauses or at least clauses. This does not allow rules with existential quantifiers or with disjunctions in the rule head, for example, stating that every human person has a mother and that every human is male or female. The seminal Cyc project [28] had its focus on commonsense rules, even including higher-order logics, but exclusively relied on human experts to manually specify them. Another major challenge with automatic rule mining arises from the open world assumption that underlies KB's and the bias in observations from Internet sources. For example, if the KB has no entrepreneur who founded more than 3 companies, this should not imply a cardinality constraint. Likewise, if the facts in the KB suggest a rule that every founder of an IT company has become a billionair, this may be caused by the bias in the KB construction (e.g., by harvesting only successful entrepreneurs from Wikipedia) and does not entail that the rule is valid in the open world.

Yet another dimension where today's KB's fall short is the **socio-cultural context** of facts or rules. Even seemingly objective statements on discoveries and inventions often depend on the background and viewpoint of a certain group of people. For example, in the US most people would state that the computer was invented by Eckert and Mauchley, whereas a German would give the credit to Konrad Zuse and a British would insist on Alan Turing (or perhaps Charles Babbage). This is not just geographical context; teenagers, for example, may largely think of Steve Jobs as the (re-) inventor of the (mobile) computer. For commonsense knowledge, it is even more critical to capture socio-cultural contexts. This also requires more thought on appropriate representations (certainly beyond SPO triples).

3.3 Interactive Search and Exploration

KB's have become so large and heterogeneous, in terms of structure and terminology, that users struggle with formulating queries – even when supported by a form-based or faceted UI (e.g., [3]). Even worse, a major use case of KB's is to serve as background reference when data scientists or business analysts combine and explore other datasets or online media. For example, a life scientist or political scientist may rapidly collect tens of interesting datasets for a specific study, but would then drastically lose her productivity when trying to join different data items and search for patterns, trends and insight.

This calls for new modes of interactive search and exploration of KB's and associated datasets. We believe the most effective way of relieving the user from the necessity to cope with the complex structure of the data, is by means of **natural language** for question answering and other interactions. User inputs such as "Which European singers covered Bob Dylan?" can be translated into structured Sparql queries. There has been substantial work on this task recently (e.g., [5, 47, 52]). However, this is still restricted to simple questions that return a single fact or a list of entities. Questions with multi-relation hops (i.e., chain joins in DB jargon) or asking for composite answers is beyond scope. Even more demanding is coping with colloquial inputs, where the user first describes her information need and context and then poses an underspecified question. An example would be: "Steve was a big fan of Bob Dylan. Didn't he date a folk singer to get closer to him? Who was that?"

Even if these issues were solved, merely translating each question into a single one-shot query would not be sufficient. The generated query may not capture the user intent, it may return unexpected, unwanted or just too few answers, and users may be utterly puzzled on how to formulate the very first question in order to start their knowledge exploration. So we need to extend question answering to interactive dialogs, with the system explaining answers and guiding the user towards better questions.

3.4 Knowledge Base Life-Cycle

It has been a major endeavor to build a comprehensive KB. However, it is an even more daunting task to maintain, grow and improve it over a long time horizon. Discovering and adding emerging entities is just the tip of the iceberg. Especially in combination with expanding the scope towards commonsense, rules and socio-cultural context, coping with new knowledge, obsolete knowledge and context-specific knowledge poses enormous challenges. The crux will be to minimize the cost of human curation, while keeping quality assurance a key priority.

References

[1] Ambiverse: Text to Knowledge. `https://www.ambiverse.com`

[2] S. Auer et al.: DBpedia: A Nucleus for a Web of Open Data. ISWC 2007

[3] H. Bast, F. Bäurle, B. Buchhold, E. Haussmann: Semantic full-text search with Broccoli. SIGIR 2014

[4] H. Bast, B. Buchhold, E. Haussmann: *Semantic Search on Text and Knowledge Bases*. Foundations and Trends in Information Retrieval 10(2-3):119-271

[5] J. Berant, A. Chou, R. Frostig, P. Liang: Semantic Parsing on Freebase from Question-Answer Pairs. EMNLP 2013

[6] K.D. Bollacker et al.: Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. SIGMOD 2008

[7] A.J. Carlson et al.: Toward an Architecture for Never-Ending Language Learning. AAAI 2010

[8] L. Chiticariu, Y. Li, F.R. Reiss: Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! EMNLP 2013

[9] O. Deshpande et al.: Building, Maintaining, and Using Knowledge Bases: a Report from the Trenches. SIGMOD 2013

[10] X. Dong et al.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. KDD 2014

[11] X. Dong: How Far Are We From Collecting the Knowledge of the World. Keynote at WebDB 2016, `http://lunadong.com/talks/tailKnowledge.pdf`

[12] S. Elbassuoni, M. Ramanath, R. Schenkel, G. Weikum: Searching RDF Graphs with SPARQL and Keywords. IEEE Data Eng. Bull. 33(1): 16-24 (2010)

[13] P. Ernst et al.: DeepLife: An Entity-Aware Search, Analytics and Exploration Platform for Health and Life Sciences. ACL 2016

[14] O. Etzioni et al.: Unsupervised Named-Entity Extraction from the Web: an Experimental Study. Artificial Intelligence 165(1): 91-134

[15] C. Fellbaum, G. Miller (Editors): *WordNet: An Electronic Lexical Database*. MIT Press, 1998

[16] L. Galárraga, G. Heitz, K. Murphy, F.M. Suchanek: Canonicalizing Open Knowledge Bases. CIKM 2014

[17] L. Galárraga, C. Teflioudi, K. Hose, F.M. Suchanek: Fast rule mining in ontological knowledge bases with AMIE+. VLDB J. 24(6): 707-730 (2015)

[18] T. Ge, Y. Wang, G. de Melo, H. Li, B. Chen: Visualizing and Curating Knowledge Graphs over Time and Space. ACL 2016

[19] T. Heath, C. Bizer: *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011

[20] S. Hellmann, C. Stadler, J. Lehmann, S. Auer: DBpedia Live Extraction. OTM Conferences 2009

[21] J. Hoffart et al.: Robust Disambiguation of Named Entities in Text. EMNLP 2011

[22] J. Hoffart, F.M. Suchanek, K. Berberich, G. Weikum (2013): YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Artificial Intelligence 194: 28-61

[23] J. Hoffart, Y. Altun, G. Weikum: Discovering emerging entities with ambiguous names. WWW 2014

[24] J. Hoffart, D. Milchevski, G. Weikum: STICS: searching with strings, things, and cats. SIGIR 2014

[25] J. Hoffart et al.: The Knowledge Awakens: Keeping Knowledge Bases Fresh with Emerging Entities. WWW 2016

[26] IBM Journal of Research and Development 56(3/4), Special Issue on "This is Watson" (2012)

[27] G. Kasneci et al.: NAGA: Searching and Ranking Knowledge. ICDE 2008

[28] D.B. Lenat: CYC: A Large-Scale Investment in Knowledge Infrastructure. Comm. ACM 38(11): 32-38

[29] Mausam et al.: Open Language Learning for Information Extraction. EMNLP-CoNLL 2012

[30] T. Mitchell et al.: Never-Ending Learning. AAAI 2015

[31] N. Nakashole, M. Theobald, G. Weikum: Scalable knowledge harvesting with high precision and high recall. WSDM 2011

[32] R. Navigli, S.P. Ponzetto: BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif. Intell. 193: 217-250 (2012)

[33] T. Neumann, G. Weikum: RDF-3X: a RISC-style engine for RDF. PVLDB 1(1): 647-659 (2008)

[34] Z. Nie, J.-R. Wen, W.-Y. Ma: Statistical Entity Extraction From the Web. Proceedings of the IEEE 100(9): 2675-2687 (2012)

[35] S.P. Ponzetto, M. Strube: Deriving a Large-Scale Taxonomy from Wikipedia. AAAI 2007

[36] N. Preda et al.: Active knowledge: dynamically enriching RDF knowledge bases by web services. SIGMOD 2010

[37] S. Razniewski, F.M. Suchanek, W. Nutt: But Do We Actually Know? AKBC 2016

[38] T. Rebele et al.: YAGO: a multilingual knowledge base from Wikipedia, Wordnet, and Geonames. ISWC 2016

[39] W. Shen, J. Wang, J. Han: Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. IEEE Trans. Knowl. Data Eng. 27(2): 443-460 (2015)

[40] J. Shin et al.: Incremental Knowledge Base Construction Using DeepDive. PVLDB 8(11): 1310-1321 (2015)

[41] R. Speer, C. Havasi: Representing General Relational Knowledge in ConceptNet 5. LREC 2012

[42] F.M. Suchanek, G. Kasneci, G. Weikum: YAGO: a Core of Semantic Knowledge. WWW 2007

[43] F.M. Suchanek, M. Sozio, G. Weikum: SOFIE: a self-organizing framework for information extraction. WWW 2009

[44] A. Talaika, J. Biega, A. Amarilli, F.M. Suchanek: IBEX: Harvesting Entities from the Web Using Unique Identifiers. WebDB 2015

[45] N. Tandon, G. de Melo, F.M. Suchanek, G. Weikum: WebChild: harvesting and organizing commonsense knowledge from the web. WSDM 2014

[46] N. Tandon, G. de Melo, A. De, G. Weikum: Knowlywood: Mining Activity Knowledge From Hollywood Narratives. CIKM 2015

[47] C. Unger, A. Freitas, P. Cimiano: An Introduction to Question Answering over Linked Data. Reasoning Web 2014

[48] D. Vrandecic, M. Krötzsch: Wikidata: a free collaborative knowledgebase. Commun. ACM 57(10): 78-85 (2014)

[49] Z. Wang et al.: Xlore: A large-scale english-chinese bilingual knowledge graph. ISWC 2013

[50] Y. Wang, M. Dylla, M. Spaniol, G. Weikum: Coupling Label Propagation and Constraints for Temporal Fact Extraction. ACL 2012

[51] W. Wu, H. Li, H. Wang, K.Q. Zhu: Probase: a Probabilistic Taxonomy for Text Understanding. SIGMOD 2012

[52] M. Yahya et al.: Natural Language Questions for the Web of Data. EMNLP-CoNLL 2012

[53] M. Yahya, D. Barbosa, K. Berberich, Q. Wang, G. Weikum: Relationship Queries on Extended Knowledge Graphs. WSDM 2016

[54] C. Zhang, J. Shin, C. Ré, M.J. Cafarella, F. Niu: Extracting Databases from Dark Data with DeepDive. SIGMOD 2016