A Tutorial on Leveraging Knowledge Graphs for Web Search

Gianluca Demartini(✉)

University of Sheffield, Sheffield, UK
g.demartini@sheffield.ac.uk

Abstract. Knowledge Graphs are large repositories of structured information about entities like persons, locations, and organizations and their relations. Modern Web search engines leverage such background Knowledge Graphs to create rich search engine result pages for entity-centric search queries.

In this document we provide an introduction to Knowledge Graphs and their application to search-related problems. We present techniques to search for entities instead of documents as answer to a search query. Finally we present human computation techniques to build hybrid human-machine systems to solve entity-oriented search tasks making use of Knowledge Graphs.

1 Introduction

Web search engines have evolved beyond just presenting the classic ten blue links in the search engine result page (SERP) as an answer to a keyword query. Modern Web search engines include in the SERP results from verticals such as news, images, videos, etc. More than that, Web search engines present rich search result pages when users query for information about specific entities like actors or movies. Depending on the type of entity users ask for, the structure of the information presented differs. Search engine result pages may include news articles, pictures, factual statements, and related entities. This is due to the fact that users of Web search engines look for specific entities on-line. Indeed, about 50 % of the query workload a commercial search engine receives is related to specific entities [20].

Information presented in such rich search engine result pages is taken at query time from a background Knowledge Graph by matching the user query against all entities stored in the Knowledge Graph. Once the relevant entity has been identified, information is retrieved from the Knowledge Graph and presented to the user.

In this document we provide an overview of the different steps involved in creating such entity-centric user experiences in Web search engines.

In Sect. 2 we present the fundamental definitions of syntax and query language for Knowledge Graphs. We describe the Relational Description Framework (RDF) as well as the SPARQL query language to access data stored in Knowledge Graphs.

P. Braslavski et al. (Eds.): RuSSIR 2015, CCIS 573, pp. 24–37, 2016.
DOI: 10.1007/978-3-319-41718-9_2

In Sect. 3 we describe the standard approaches to extract and uniquely identify entities such as persons, locations, and organizations in textual content such as, for example, news article or general Web pages.

In Sect. 4 we introduce systems that make use of entities to provide search functionalities to Web users. We describe indexing and ranking approaches related to a number of entity-oriented search tasks.

In Sect. 5 we discuss hybrid human-machine systems that make use of crowdsourcing to deal with Knowledge Graph related problems such as entity linking, data integration, and entity search.

Finally, Sect. 6 concludes this document highlighting open direction for future research.

2 Introduction to Knowledge Graphs

Rich SERPs containing entities are possible thanks to what goes under the name of *Web of Data* and Knowledge Graphs. In the case of Google, the Knowledge Graph project launched in May 2012 following the acquisition of Metaweb in July 2010 with its main product Freebase: a freely editable Knowledge Graph available both in human-readable and machine-readable formats.

Similarly, other Knowledge Graphs have been created by the academic community under the label of *Linked Open Data*[1] (LOD). Most popular datasets in LOD include DBpedia[2], Freebase[3], and YAGO[4]. DBpedia is a collection of RDF triples automatically extracted from Wikipedia article leveraging the semi-structured information present in it (e.g., the Wikipedia info-boxes) and manually defined mapping rules that allow to extract structured data out of Wikipedia articles.

In the commercial domain, Knowledge Graphs have been developed and used to power end-user applications such as Web search. Examples include the Google Knowledge Vault [15], the Facebook Entity Graph[5], and Microsoft Satori which is used, among other applications, to power rich search engine result pages.

2.1 Information Extraction and Knowledge Acquisition

The process in which information is extracted from unstructured text to create Knowledge Graphs is called *Information Extraction and Knowledge Acquisition*. Starting from textual data, information extraction techniques will identify structured components like, for example, entities and factual statements. Then, the process of knowledge acquisition connects factual statements generating a Knowledge Graph by applying techniques for fact consistency and data integration.

[1] http://linkeddata.org.
[2] http://dbpedia.org.
[3] https://www.freebase.com/.
[4] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/.
[5] https://www.facebook.com/notes/facebook-engineering/under-the-hood-the-entities-graph/10151490531588920.

Early information extraction solutions were based on manually defined patterns matched against textual documents in order to identify expected occurrences of statements. Early solutions used to construct Knowledge Graphs have been based on manual effort like, for example, Cyc [22] which has collected 250 thousand entities over 20 years. The objective of current work is the combination of information extraction techniques to automatize and scale-up the knowledge acquisition process.

In summary, we can group approaches to construct Knowledge Graphs into manually-supported construction (e.g., Freebase and Wikidata) where human intervention is used to complement automatically extracted data, and automatic methods. Among automatic methods we can find Knowledge Graphs extracted from semi-structured data like DBpedia and YAGO which leverage existing structure in Wikipedia, Knowledge Graphs extracted from text, and Knowledge Graphs resulting from the combination of text extraction and database integration like the Google Knowledge Vault [15].

When creating Knowledge Graphs a number of challenges have to be tackled. These include the choice about which entities to include and which not to include. As an example, Knowledge Graphs based on Wikipedia follow the *notability criteria*[6] which states that only popular entities should be included. Thus, these Knowledge Graphs trade-off high data accuracy for low coverage. Another challenge is about keeping information in the Knowledge Graph up-to-date. For this, two alternatives are usually adopted: either outdated information is modified or factual statements are annotated with a time validity interval (e.g., the fact 'George W Bush is president of the USA' has a time validity between 2001 and 2009).

2.2 RDF Data Format

The W3C standard to define structured information on the Web is RDF[7]. Data in LOD is typically available in RDF format to download or to access over SPARQL endpoints as answer to a query.

RDF, standing for Resource Description Framework, encodes data as triples in the form of *subject*, *predicate*, and *object* which naturally form distributed graphs. This data format can be used to describe semi-structured information and factual statements (e.g., subject:Tom_Cruise predicate:Starring_In object:Top_Gun). Entities appearing as subject or object of RDF statements are represented by Unique Resource Identifiers (URIs) in order to relate all statements about the same entities. Additionally to that, the object of an RDF statement can assume a numerical or literal value instead of an entity (e.g., subject:Tom_Cruise predicate:Born_In object:1962).

Data in RDF format can be encoded in different ways. Traditionally, XML serializations were used to store and exchange RDF data. Recently, the use of

[6] https://en.wikipedia.org/wiki/Wikipedia:Notability.
[7] http://www.w3.org/RDF/.

JSON-LD[8], a variant of the popular JSON format, has become popular to encode such factual statements about entities and their properties.

In order to define data in RDF we need to follow a schema, that is, a common vocabulary used to define attributes of entities (e.g., 'starring_in'). To this aim, RDF Schemas (RDFS) are defined to be re-used. RDFS define properties and classes to be used for a certain domain and have been defined by the corresponding community. Popular examples, of RDFS include *foaf*[9] to define relations in social networks, *Dublin Core*[10] to describe publications, and *Good Relations*[11] used to describe products. RDFS constructs include *classes* (i.e., types of entities) and *properties* of entities. It is also possible to create hierarchies by using *SubClassOf* and *SubPropertyOf* constructs. Finally, it is possible to create constraints on the class of subject and object for a certain predicate by defining the predicate *Domain* and *Range* respectively.

Once data is represented in RDF format and stored in an *RDF store*, structured queries can be run against it to retrieve specific subsets of the data. Similar to SQL for Databases, RDF data can be queried using SPARQL[12]: A declarative query language for RDF and RDFS. SPARQL leverages the fact that RDF is represented as triples and defines *triple patterns* to be matched against the data. Like for SQL, a number of constructs exists for SPARQL including 'order by', 'distinct', 'limit', etc.

3 Named Entity Recognition and Linking to Knowledge Graphs

In this section we describe a classic Natural Language Processing (NLP) pipeline consisting of Named Entity Recognition (NER) followed by Entity Linking (i.e., entity disambiguation). Next, we describe one additional step on top of such NLP pipeline which consists of the selection of entity types from the Knowledge Graph to be displayed to users reading a document where a certain entity appears.

The first step to process unstructured documents to create a connection to an existing Knowledge Graph is NER, also known as Entity Extraction. The step of NER consists of identifying entity mentions in textual documents. Thus, the result of NER will be the indication of which n-gram in the text represents an entity (e.g., a person, a location, or an organization). Next, the step of entity resolution or entity linking consists of uniquely identifying the extracted entity by creating a link to an entity described in an external Knowledge Graph. In this way, we 'resolve' the extracted n-gram to one specific entity and we are able to uniquely identify it. This requires also the step of entity classification, that is, distinguishing among different types of entity.

[8] http://json-ld.org/.
[9] http://www.foaf-project.org/.
[10] http://dublincore.org/.
[11] http://www.heppnetz.de/projects/goodrelations/.
[12] http://www.w3.org/TR/rdf-sparql-query/.

3.1 Named Entity Recognition

In order, the steps of classic NLP pipelines are: tokenization, sentence splitting, part-of-speech (POS) tagging, NER, entity-linking, co-reference resolution, and relation extraction. NER approaches make use of tokenization and POS tags and can be classified in the following categories: dictionary-based, pattern-based, and supervised learning models. While the simpler approaches based on dictionaries and patterns work well in certain domains like, for example, for geographical entities where it is easy to obtain a comprehensive list of existing entities, most advanced techniques make use of supervised machine learning models. In detail, NER can be formulated as a classification task where the goal is to assign to each token the tag B (i.e., beginning of an entity), I (i.e., continuation of an entity), or O (i.e., word outside an entity). Next, a classification problem can be defined to attach a type to an entity (e.g., person, organization, etc.). Common machine learning models used for these problems are Decision Trees and Support Vector Machines. Finally, the most popular NER approaches use probabilistic sequence models where Hidden Markov Models and Conditional Random Fields are the most widely used. In this type of approaches each token in a sequence is assigned to a label which is dependent on the labels of tokens in its proximity. Popular features used by supervised models used for NER include gazetteers, orthographic features (e.g., capital letters), word types, POS tags, context, and trigger words (e.g., 'Mr', 'Miss'). Software libraries that implement state of the art NER approaches have been developed. Popular libraries include the Stanford Named Entity Recognizer[13] for Java and NLTK[14] for Python.

3.2 Entity Linking

Once named entities have been identified, the next step is to use the entity mention to gather candidate entities from the Knowledge Graph. Then, by producing a ranked list of candidate matching entities, we can select the best match for the entity mentioned in the document. More than text similarity measures, it is possible to leverage the context in which entities appear to obtain better disambiguation. The intuition is that entities that co-occur in the same textual context tend to be more related each other. To leverage this intuition we need to express such relatedness in a numerical way by, for example, measuring statistical co-occurence, similarity of entity descriptions in the Knowledge Graph, or measuring the distance of the entities following relations in the Knowledge Graph. As an example of relatedness measured by co-occurence, we can imagine a document mentioning both the entity 'FC Barcelona' and the entity 'Bayern'. Based on textual match, the entity 'Bayern' could match to either the entity 'Bavaria' (i.e., the region in Germany) or the entity 'Bayern München' (i.e., the football team) in the Knowledge Graph. As from other documents in the collection we can observe that the entity 'FC Barcelona' (i.e., a football team) often co-occurs with the entity 'Bayern München', we select this option as the most likely match for the mention extracted from the document.

[13] http://nlp.stanford.edu/software/CRF-NER.shtml.
[14] http://www.nltk.org.

More than just general text documents on the Web like, for example, news articles, entities may appear in other textual content. Recent research focus has been put on identifying entities appearing in structured tabels on the Web [16], micro-blogs [19], and Web search queries [20,24] where adapted approaches have been designed.

Recent work has shown that about 70 % of Web search queries contain a named entity, that about 50 % of queries have an entity focus, and that 10 % of queries are looking for a certain entity type [20]. In these cases, the query intent is typically expressed by words additionally to the mentioned entity used to disambiguate it, for example, by expressing its type (e.g., 'tom cruise actor'). NER approaches developed to work best on search queries look at certain keyword matches [3] and at looking up entity names in the Knowledge Graph [5]. As a demonstration of the interest of the Information Retrieval (IR) community to NER and Entity Linking, at SIGIR 2014 the Entity Recognition and Disambiguation Challenge was run to disambiguate entities appearing in Web search queries and Web pages to entities in the Freebase Knowledge Graph.

3.3 Ranking Entity Types

Once entities appearing in documents have been extracted and disambiguated by creating links to entities in the Knowledge Graph, we can start to create end-user applications that make use of the information available in the Knowledge Graph. As an example application, we now describe a system to select the most relevant type of an entity mentioned in a document to be showed to users reading the document as an explanation of the entity they are reading about [29].

Once an entity mention in a document is disambiguated to one entity appearing in a Knowledge Graph, a number of factual statements about that entity are available. One piece of information is the *type* of the entity. For example, for the entity 'Tom Cruise', DBpedia shows more than 40 different types attached to it including 'Person', 'Artist', 'Actor', and 'Producer'. The task is to select one of these types as the most informative for a user to see when reading a document mentioning Tom Cruise. While it is clear that the type Person may, in most of the cases, be not so informative as it is too general, the selection between Actor and Producer may depend on the textual context where the entity appears.

The NLP pipeline developed for this [29] looks similar to the classic NLP pipeline up to the task of entity linking. After this, a supervised machine learning model is used to produce a ranked list of entity types obtained from the Knowledge Graph. Among the diverse set of features used to rank entity types, the most effective ones are based on the type hierarchy (i.e., information about the fact that Actor is a sub-type of Person) and on the types of other entities co-occuring with the entity for which we are ranking types.

When applying such NLP pipelines more than just accurate results we would like the approaches to work at Web scale. In [29] the proposed pipeline (which is available at https://github.com/MEM0R1ES/TRank) has been deployed over an Hadoop cluster using a Map/Reduce implementation. By building inverted indices on top of the Knowledge Graph to store entity description used for entity linking and to store the entity type hierarchy used to select entity types, it is

possible to distribute such data structures on each node of the cluster and run the pipeline in parallel. The specific implementation could run the pipeline at a rate of 100 documents per node per second. Interestingly, the split of the overall execution time among the different steps in the pipeline indicates that the most time consuming step is NER (with 36 %) while the least time consuming step is type ranking with 6 % of the overall time spent to process a document.

4 Searching for Entities

As mentioned in Sect. 3 searching for entities is becoming more and more popular over time for Web search engine users. In this section we describe different approaches for entity search. We start from the historically first entity type for which search systems have been developed, that is, persons, by describing how expert finding systems work. Then, we discuss general entity ranking on top of a Knowledge Graph. Next, we talk about the popular ad-hoc object retrieval task (i.e., given a keyword query describing an entity, retrieve the matching entity from a Knowledge Graph).

4.1 Expert Finding

Expert finding is defined as the search task of returning a list of candidate experts ranked by expertise on the topic described by the user query [1]. Expert finding systems are most valuable in a large enterprise setting where skills and competencies are spread across departments. In a scenario where executives needs to create teams to work on new projects or simply to find the right person to solve a problem, expert finding systems can produce a ranked list of company employees on a certain topic described by the user query. Such ranked list is generated based on the digital content available in the enterprise after being appropriately indexed.

We distinguish two basic expert finding approaches: *document-centric* and *candidate-centric* approaches.

In document-centric approaches an inverted index is created on top of the documents available within the enterprise. Then, given a user query a ranked list of documents matching the user query is generated using classic IR ranking models. Finally, candidate experts are extracted from top ranked documents to generate a list of experts to be returned as answer to the query.

In candidate-centric approaches, names of candidates are extracted from all documents first. Then, candidate profiles are created by aggregating information from all documents where the candidate name appears. Finally, an index is created over candidate profiles and, given a user query, a ranked list of candidate profiles is generated as answer.

Other models for expert finding exist. One example include voting models where, by means of data fusion techniques, ranked documents are seen as votes to candidate expertise. Votes are than aggregated by taking into account different features like, for example, document rank, score, or the number of different documents voting for the same candidate [21]. Another example of expert finding

are user-oriented models where additional real-world constrains are taken into account. Possible dimensions are the user previous knowledge on the topic she is looking experts for (i.e., retrieved experts should have wider knowledge than the user) and contact time (i.e., taking into account geographical location of the user and the candidate experts or their distance in the organizational hierarchy) [27].

4.2 Entity Ranking

Generalizing expert finding techniques to any entity type, we talk about entity ranking systems. Thus, we aim at building systems which can, at the same time, answer queries about, for example, 'Impressionist art museums in Holland' as well as about 'German car manufacturers'. Because of the availability of multiple entity types, Wikipedia has been used as main background collection for entity ranking systems. Over time, effective approaches have strongly leveraged the category structure of Wikipedia to identify relevant entity types, looked at query extension techniques by means of synonyms and other related words, and built on top of the link structure connecting Wikipedia pages to identify alternative entity labels in the anchor text of hyperlinks [10].

More than searching for entities over a static collections like Wikipedia, entity ranking approaches are useful when ranking entities in a collection evolving over time. The most relevant example is that of news articles that are published over time about the same story. In the case of news events that evolve over time entities may appear only later in the story or be present at the beginning but than become not relevant anymore. Thus, in such cases we want entity ranking systems that leverage past information to generated better rankings [11].

4.3 Ad-hoc Object Retrieval

A different entity search task is Ad-hoc Object Retrieval (AOR) [24] which is highly related to the NLP task of Entity Linking. In this task, a user keyword query describing one specific entity is used to generated a ranked list of entity identifiers retrieved from a Knowledge Graph. The goal is to identify which entity the user is looking for in order to display him with some structured information about the entity as retrieved from the background Knowledge Graph. This is typically done as a way to support end users who cannot express their information need using SPARQL and by giving up query expressivity to scale over large amounts of data. Moreover, users are not aware of the RDFS schema used to store and describe the data in the Knowledge Graph and thus there is the need to match any user keyword query against the available data.

In order to efficiently answer such queries without need to scan the entire collection, we must construct an index over the data stored in the Knowledge Graph. Given the semi-structured nature of Knowledge Graph data (i.e., it contains the structure relating entities as well as textual descriptions of entities), we can leverage IR indexing techniques for unstructured content still maintaining the possibility to search over the structure present in the data. Two alternative techniques have been proposed [4]: Horizontal index structure and Vertical

index structure. Horizontal indexing requires two indices: one for terms and one for properties in the Knowledge Graph. Thus, for each term we store on the same index position the properties where it appears. In this way we need to store just two indices which grow horizontally as more data is added to the index. The dictionary size is the number of unique terms plus the number of properties in the Knowledge Graph. Vertical indexing requires to store one index per property. Each index will store all the terms appearing for that property. Thus, this type of index grows vertically in the number of properties. While the dictionary in this case is smaller (i.e., the number of unique terms), a dataset with many distinct properties could become inefficient as many merging operations across indices may need to take place to answer a query. To answer a query over these types of indexing techniques, functions that rank over multiple indices can be used (e.g., BM25F [25]).

An alternative indexing solution is the use of IR inverted indices in combination with graph-based indices over which SPARQL queries can be run [30]. In this system, given a query, a ranked list of entities is generated first using standard approaches. Then, as a refinement step, top-k results are looked up in the graph in order to leverage the graph structure to re-rank results and obtain a more effective AOR system. While graph traversal operations generate some overhead in terms of computational complexity, the efficiency of the system remains acceptable to answer user queries in real-time. Recent work on AOR has looked at approaches that account for term dependencies in the case of multi-field entity descriptions like in the case of the AOR task [31].

5 Crowdsourcing for Knowledge Graphs

In this section we present the concept of *crowdsourcing* and explain how it can be used to improve the quality of systems that make use of Knowledge Graphs. We will look at crowd-based entity linking system as well as at the use of crowdsourcing to understand complex Web search queries.

5.1 Crowdsourcing

Crowdsourcing is defined as the use of human intelligence at scale to solve problems that are simple for humans to complete but still difficult for machine-based algorithms (e.g., image labelling, text summarization, translation, etc.). When applied to Knowledge Graphs, crowdsourcing makes use of micro-tasks that take from few seconds to a couple of minutes for an individual to complete. The execution of tasks happens in so called *crowdsourcing platforms* like, for example, Amazon MTurk[15] where the *workers* (i.e., the crowd) and the *requesters* (i.e., those who publish tasks to be completed) meet. Workers complete Human Intelligence Tasks (HITs) which are grouped in batches of similar tasks in exchange of a small monetary reward. The access to the platform from the requester point of view can typically be done by means of a website or programmatically over APIs.

[15] http://mturk.com.

At the moment, the most popular crowdsourcing platform is Amazon MTurk which was launched in 2005. This platform works as a marketplace where requesters publish batches of HITs to be completed and workers are free to pick among the available tasks those they wish to complete in order to gain the reward assigned by requesters. On top of the assigned reward the platform takes a transaction fee proportional to the reward and to the number of tasks published. In [13] an analysis of the evolution of this crowdsourcing platform over time is presented.

Crowdsourcing has been applied in a variety of areas in computer science. For example, the IR community has leveraged crowdsourcing as a means to obtain relevance judgements at scale. This is possible as crowd workers are cheaper and faster than the traditional assessors that produce relevance judgments. The Semantic Web community has used crowdsourcing for classic problems like ontology mapping where micro-tasks ask crowd workers to verify or identify mappings (e.g., 'is A a type of B'?) [26]. Crowdsourcing has also been used for ontology engineering in the biomedical domain showing that crowd workers, when provided with the relevant background knowledge, can perform as effectively as domain experts in this task [23].

5.2 Knowledge Graph Applications

When applied to Knowledge Graph problems, crowdsourcing has been used for entity linking [8], search query understanding [12], search result extraction [2], and Knowledge Graph enrichment [18].

ZenCrowd [8] combines both algorithmic and manual entity linking by dynamically assessing the quality of human work and aggregating crowd answers with algorithmic results based on a probabilistic reasoning framework. Input documents are first processed over a classic NLP pipeline (see Sect. 3) where entities are extracted by means of NER techniques and a ranked list of candidate entities from the Knowledge Graph is generated. At this point, a decision engine decides, based on entity linking result confidence, which entity to crowdsource. Once crowd answers are available, they are sent together with the original algorithmic results to a probabilistic network for the final linking decision. The probabilistic network combines prior probabilities form the entity linking method and crowd worker confidence scores and, by weighting in these signals, decides which links are to be selected for a certain entity. Experimental results have shown that (1) the use of crowdsourcing improves the quality of links generated by algorithmic approaches and that (2) the probabilistic network combination of human and machine answers improve the quality of the results over the plain crowd answers. This is possible thanks to the identification over multiple entities of trustworthy workers in the crowd and by giving an higher weight to their answer. This shows the importance of being able to identify the best workers in the crowd for a certain task which is an active area of research [6,14].

The idea of using a probabilistic network to combine crowd and algorithmic results has been used also for data integration where the same entity appearing

in multiple datasets has to be identified [9]. In this case a three-way blocking approach has been defines where crowdsourcing is seen as the most expensive similarity measure.

A final example of crowdsourcing applied to entity-oriented search is CrowdQ [12]: A system that uses crowdsourcing to understand the intended meaning of complex Web search queries by building a structured query template and answering it over a Knowledge Graph.

6 Open Research Problems

6.1 Knowledge Graphs

In the area of Knowledge Graphs there is a number of open research questions. One of these is about Knowledge Graph growth: As discussed in Sect. 2, Knowledge Graphs are often incomplete (e.g., because of the notability criteria of Wikipedia). This requires techniques to add relations to the graph (i.e., link prediction), to connect different graphs (i.e., ontology matching), and to add new entities to the Knowledge Graph. More than that, existing information may not be correct. Errors may be due, for example, to non-perfect entity resolution which causes the existence of duplicate entities in the Knowledge Graph. Another example of open research question is about how to best let users access data in the Knowledge Graph as we cannot expect the average Web user to type a SPARQL query. In this direction of research there is the need to work on *semantic parsing* to better interpret user queries and on *question answering* systems built on top of Knowledge Graphs.

6.2 Entity Search

Open research questions in the area of entity search go beyond the work on improving system effectiveness and efficiency for existing tasks. The heavy use of NLP pipelines requires a better understanding on how errors propagate over this pipelines and which effect they have on the final result presented to the end user. More than that, there is a need for work looking at the user experience and on novel entity-centric user interfaces that would allow user to access information in a richer way than just by using keywords to express an information need and consuming results as a ranked list. In this line of work, exploratory search of collections can benefit from entity-centric approaches allowing users to navigate hybrid graphs of documents and entities interconnected together.

Novel entity-centric search tasks can be looked at as well. Examples include searching for relations, attributes, or, in general, for more complex entity queries involving joins of datasets and complex requests (e.g., 'birthdate of the mayor of the capital city of Italy'). For such complex information needs the use of crowdsourcing has been proposed to solve the problem of query parsing and understanding [12,28].

6.3 Crowdsourcing

In Sect. 5 we have seen examples of so called hybrid human-machine systems that leverage machines to scale over large amounts of data as well as human intelligence by means of crowdsourcing to keep the quality of the results high.

While these systems generally provide more accurate results, a number of challenges have to be taken into account when designing such systems. First, the use of financial incentives attracts malicious workers who aim at obtaining the monetary reward attached to tasks without caring about the accurate completion of the task [17]. Moreover, the efficiency of hybrid systems is highly unpredictable because of the human-in-the-loop component which is difficult to schedule. A number of problems around effectiveness and efficiency of crowdsourcing platforms have still to be solved [7].

References

1. Balog, K., Fang, Y., de Rijke, M., Serdyukov, P., Si, L.: Expertise retrieval. Found. Trends Inf. Retrieval **6**(2–3), 127–256 (2012)
2. Bernstein, M.S., Teevan, J., Dumais, S., Liebling, D., Horvitz, E.: Direct answers for search queries in the long tail. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2012, pp. 237–246. ACM, New York (2012)
3. Blanco, R., Cambazoglu, B.B., Mika, P., Torzec, N.: Entity recommendations in web search. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part II. LNCS, vol. 8219, pp. 33–48. Springer, Heidelberg (2013)
4. Blanco, R., Mika, P., Vigna, S.: Effective and efficient entity search in RDF data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 83–97. Springer, Heidelberg (2011)
5. Blanco, R., Ottaviano, G., Meij, E.: Fast and space-efficient entity linking for queries. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM, Shanghai, China, 2–6 February 2015, pp. 179–188 (2015)
6. Bozzon, A., Brambilla, M., Ceri, S., Silvestri, M., Vesci, G.: Choosing the right crowd: expert finding in social networks. In: Proceedings of the 16th International Conference on Extending Database Technology, EDBT 2013, pp. 637–648. ACM, New York (2013)
7. Demartini, G.: Hybrid human-machine information systems: challenges and opportunities. Comput. Netw. **90**, 5–13 (2015)
8. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: ZenCrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 469–478. ACM, New York (2012)
9. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: Large-scale linked data integration using probabilistic reasoning and crowdsourcing. VLDB J. **22**(5), 665–687 (2013)
10. Demartini, G., Firan, C.S., Iofciu, T., Krestel, R., Nejdl, W.: Why finding entities in wikipedia is difficult, sometimes. Inf. Retr. **13**(5), 534–567 (2010)

11. Demartini, G., Missen, M.M.S., Blanco, R., Zaragoza, H.: TAER.: time-aware entity retrieval-exploiting the past to find relevant entities in news articles. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010, pp. 1517–1520. ACM, New York (2010)
12. Demartini, G., Trushkowsky, B., Kraska, T., Franklin, M.J.: CrowdQ: crowdsourced query understanding. In: CIDR, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 6–9 January 2013, Online Proceedings (2013)
13. Difallah, D.E., Catasta, M., Demartini, G., Ipeirotis, P.G., Cudré-Mauroux, P.: The dynamics of micro-task crowdsourcing: the case of Amazon MTurk. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, pp. 238–247. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2015)
14. Difallah, D.E., Demartini, G., Cudré-Mauroux, P.: Pick-a-crowd: tell me what you like, and i'll tell you what to do. In: Proceedings of the 22nd International Conference on World Wide Web, WWW 2013, pp. 367–374. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2013)
15. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, pp. 601–610. ACM, New York (2014)
16. Elmeleegy, H., Madhavan, J., Halevy, A.Y.: Harvesting relational tables from lists on the web. VLDB J. **20**(2), 209–226 (2011)
17. Gadiraju, U., Kawase, R., Dietze, S., Demartini, G.: Understanding malicious behavior in crowdsourcing platforms: the case of online surveys. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, pp. 1631–1640. ACM, New York (2015)
18. Ipeirotis, P.G., Gabrilovich, E.: Quizz: targeted crowdsourcing with a billion (potential) users. In: Proceedings of the 23rd International Conference on World Wide Web, WWW 2014, pp. 143–154. ACM, New York (2014)
19. Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A., Lee, B.-S.: TwiNER: named entity recognition in targeted Twitter stream. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2012, pp. 721–730. ACM, New York (2012)
20. Lin, T., Pantel, P., Gamon, M., Kannan, A., Fuxman, A.: Active objects: actions for entity-centric search. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 589–598. ACM, New York (2012)
21. Macdonald, C., Ounis, I.: Voting for candidates: adapting data fusion techniques for an expert search task. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM 2006, pp. 387–396. ACM, New York (2006)
22. Matuszek, C., Cabral, J., Witbrock, M.J., DeOliveira, J.: An introduction to the syntax, content of Cyc. In: AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, pp. 44–49 (2006)
23. Mortensen, J., Musen, M.A., Noy, N.F.: Crowdsourcing the verification of relationships in biomedical ontologies. In: AMIA, American Medical Informatics Association Annual Symposium, Washington, DC, USA, 16–20 November 2013 (2013)

24. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 771–780. ACM, New York (2010)
25. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retrieval **3**(4), 333–389 (2009)
26. Sarasua, C., Simperl, E., Noy, N.F.: CrowdMap: crowdsourcing ontology alignment with microtasks. In: Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E., Cudré-Mauroux, P. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 525–541. Springer, Heidelberg (2012)
27. Smirnova, E., Balog, K.: A user-oriented model for expert finding. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 580–592. Springer, Heidelberg (2011)
28. Teevan, J., Collins-Thompson, K., White, R.W., Dumais, S.: Slow search. Commun. ACM **57**(8), 36–38 (2014)
29. Tonon, A., Catasta, M., Demartini, G., Cudré-Mauroux, P., Aberer, K.: *TRank*: ranking entity types using the web of data. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 640–656. Springer, Heidelberg (2013)
30. Tonon, A., Demartini, G., Cudré-Mauroux, P.: Combining inverted indices and structured search for Ad-hoc object retrieval. In: The 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2012, Portland, OR, USA, 12–16 August 2012, pp. 125–134 (2012)
31. Zhiltsov, N., Kotov, A., Nikolaev, F.: Fielded sequential dependence model for Ad-hoc entity retrieval in the web of data. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2015, pp. 253–262. ACM, New York (2015)

Springer

http://www.springer.com/978-3-319-41717-2

Information Retrieval
9th Russian Summer School, RuSSIR 2015, Saint Petersburg, Russia, August 24-28, 2015, Revised Selected Papers

Braslavski, P.; Markov, I.; Pardalos, P.M.; Volkovich, Y.; Ignatov, D.I.; Koltsov, S.; Koltsova, O. (Eds.)

2016, XX, 187 p. 40 illus., Softcover
ISBN: 978-3-319-41717-2