

Using microtasks to crowdsource DBpedia entity classification: A study in workflow design

Qiong Bu, Elena Simperl, Sergej Zerr and Yunjia Li

School of Electronics and Computer Science, University of Southampton

E-mail: {qb1g13, e.simperl,s.zerr, yl2}@soton.ac.uk

Abstract. DBpedia is at the core of the Linked Open Data Cloud and widely used in research and applications. However, it is far from being perfect. Its content suffers from many flaws, as a result of factual errors inherited from Wikipedia or incomplete mappings from Wikipedia infobox to DBpedia ontology. In this work we focus on one class of such problems, un-typed entities. We propose a hierarchical tree-based approach to categorize DBpedia entities according to the DBpedia ontology using human computation and paid microtasks. We analyse the main dimensions of the crowdsourcing exercise in depth in order to come up with suggestions for workflow design and study three different workflows with automatic and hybrid prediction mechanisms to select possible candidates for the most specific category from the DBpedia ontology. To test our approach, we run experiments on CrowdFlower using a gold standard dataset of 120 previously unclassified entities. In our studies human-computation driven approaches generally achieved higher precision at lower cost when compared to workflows with automatic predictors. However, each of the tested workflows has its merit and none of them seems to perform exceptionally well on the entities that the DBpedia Extraction Framework fails to classify. We discuss these findings and their potential implications for the design of effective crowdsourced entity classification in DBpedia and beyond.

Keywords: task design, workflow design, entity classification, DBpedia, microtask crowdsourcing, CrowdFlower

1. Introduction

DBpedia is a community project, in which structured information from Wikipedia is published as Linked Open Data (LOD) [1]. The resulting dataset consists of an ontological schema and a large number of entities covering, by virtue of its origins, a wide range of topics and domains. With links to many other sources in the LOD Cloud, it acts as a central hub for the development of many algorithms and applications in academia and industry. Still, no matter how popular, DBpedia is far from being perfect. Its many flaws have been subject to extensive studies and inspired researchers to design tools and methodologies to systematically assess and improve its quality [2,3,4].

In this paper, we focus on a particular class of errors, the un-typed entities. According to the statistics¹, the English DBpedia (version 3.9) contains 4.58 million entities, but only 4.22 million among them are classified according to the DBpedia ontology. There are still a large amount of entities unclassified in the latest release of DBpedia (4.3M out of 5.9M are classified in version 2015-04, 5M out of 6.2M are consistently classified in 2015-10 version)². This gap is present due to several factors including incomplete or incorrect mappings from Wikipedia infoboxes to the DBpedia ontology³ or entity ambiguity. Any attempt to fix the problem will most likely require human intervention [5].

¹<http://wiki.dbpedia.org/news/dbpedia-version-2014-released>

²<http://wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets>

³http://mappings.dbpedia.org/server/statistics/*/

DBpedia relies on a community of volunteers and a wiki to define and maintain a collection of mappings in different languages⁴. However, this is a process that is primarily designed for domain experts; it requires a relatively high degree of familiarity with knowledge engineering and the intricacies of both Wikipedia and DBpedia. The participation of the volunteers is primarily intrinsically motivated. While this self-organizing community approach works well in terms of classification precision, it is also acknowledged to be rather slow and challenging to manage and as a consequence, a substantial share of mappings is still missing [6]. Other forms of human input collection could improve on these aspects and the most important in this context are paid microtasks and crowdsourcing.

Paid microtask crowdsourcing employs services such as Amazon’s Mechanical Turk⁵ or CrowdFlower⁶ to undertake work as a series of small tasks that combined together comprise a large unified project to which many people are contributing. It is typically applied to repetitive activities that lend themselves to parallelization and do not require specific expertise beyond the most common human knowledge and cognitive skills. A paid microtasks project is broken down into many units that are self-contained and executed independently by different people for a fee in the range of several cents to a dollar. This approach has already been successfully employed in different areas related to Linked Data and semantic technologies [7,8,9,10]. In particular, [2,4,8] have shown that it can achieve reasonable accuracy in quality repair and entity typing tasks compared to expert crowds at a faster turnaround, however to our best knowledge, none of the works made use of the structural dependencies between the classes as it is the case in Ontologies.

In this work, we consider the problem of systematically selecting the most specific category from a tree of hierarchically organized labels through employing microtasks. We contribute with the in-depth analysis of the main crowdsourcing exercise dimensions in order to come up with suggestions for workflow design, which are grounded in existing literature in cognitive psychology, and elaborate on their implications in terms of precision and cost. To this end, we propose a workflow model consisting of a predictor, able to suggest category candidates for a given entity, as well as a crowd based error detection and correction al-

gorithms. Using three alternative workflows, we compare the performance of an automatic predictor and two crowd based approaches: a naive predictor, where the whole ontology tree is traversed top down by the crowd and a microtask based free text predictor, that is able to make a decision based on human text input. To test our model, we run experiments on CrowdFlower using a dataset of 120 entities previously unclassified by the DBpedia community and compare the answers from the crowd with gold standard created by experts from our research group. Our experiments show that the crowd based approach achieved higher precision at lower cost compared to the workflow with automatic predictor. However, each workflow has its merit and we provide in-depth evaluation of their properties to support experts in selection decisions.

The rest of this paper is structured as follows: we start with an analysis of the state of the art in the areas of entity classification and crowdsourcing task design in Section 2. In Section 3 we present our entity classification model based on machine and human input. Section 4 is dedicated to the design of our experiments where we measure the precision and costs of our model in three alternative workflows. Section 5 discusses the evaluation results and their implications for applying human computation and microtasks to similar scenarios. Finally, we conclude with a summary of our findings and plans for future work in Section 6.

2. Related work

Our work on using the crowd to effectively classify DBpedia entities is closely related to entity classification techniques in general and crowdsourcing based tasks similar to the problem discussed in this paper. We take a closer look at research in these areas and review how existing methods differ from our approach or inspire it in some way.

2.1. Entity annotation and classification

Generally, entity classification can be carried out in different ways, ranging from manual annotation by experts from the corresponding domain, over hybrid approaches where human input and machine algorithms are combined, to the tools for purely automatic classification. As the expert classification is costly and time-consuming task, automatic entity classification traditionally has been part of the NER (Named Entity Recognition) research where the entity is first au-

⁴<http://mappings.dbpedia.org/index.php>

⁵<https://www.mturk.com/mturk/welcome>

⁶<http://www.crowdfLOWER.com/>

tomatically identified and then classified [11,12,13, 14] according to a number of categories. Unfortunately, such automatic algorithms, require predefined concept-based seeds for training and manually defined rules, which complexity depends on the number of involved classes. As a consequence, the classification ability is typically limited to a relatively small number of classes such as "Person", "Location", "Time" and "Organisation". Those approaches are often combined with machine learning algorithms or ensembles trained on large pre-annotated datasets. However, all discussed techniques are working with a certain error margin and hence their output requires to be manually re-examined and further classified. There is a number of existing tools and APIs such as DBpedia spotlight,⁷ Dandelion,⁸ Alchemy API,⁹ Open Calais,¹⁰ GATE,¹¹ and NERD¹² that can be used to classify entities based on a predefined ontology, but they very often fail to produce a type for some of the entities. On the other hand, entity classification by combining human and machine is a promising alternative as discussed in [15,16,17,18,19]. Some studies show that it is possible to combine automatic prediction methods (Bayesian/Generative probabilistic models) with additional input from the crowd to improve output accuracy [16,20,21,22]. Wang et al. [19] have also shown the performance advantage of the hybrid approach when comparing with fully automatic methods. There exist basically two types of hybrid approaches: The first is based on collecting a large amount of annotations from the crowd and use the collected labels to train machine learning algorithms in order to achieve better classification quality. An example for such approach is the ESP game [18] for gamification based images labelling. The other approach is to use a machine to narrow down the possible options and then employ the crowd to validate or chose the best matching one. As an example, the work presented in [23] employs a machine-based algorithm to classify entities along with calculating a confidence score. The authors suggest that the label crowdsourcing is required only for entities with low confidence scores produced by the classifier.

In contrast to the works mentioned above, our approach can efficiently deal with a large number of

classes and keep the error margin narrow at the same time. Based on the discussed literature, in our workflows, we utilise automatic tools and algorithms as predictors for producing a small set of candidate suggestions from a large set (over 700) of DBpedia classes, hence significantly reducing the search space.

2.2. Crowdsourcing task design

In recent years, researchers have successfully applied human computation and crowdsourcing to a variety of scenarios that use Linked Data and semantic technologies where classification is one of the most popular tasks. The idea is to decompose tasks with higher complexity into smaller sub-tasks [24] such that each of those tasks can be solved by non-expert crowdworkers. Due to the decentralized and diverse nature of possible participants in crowdsourcing work, data validation and quality control have been an essential topic explored by many previous researchers, resulting in challenges in workflow and task design.

One challenge is to design an effective workflow and a mechanism to aggregate the sub-task outputs into a high quality final result. For example, to solve the complex authoring task using crowdsourcing platform, Bernstein et al. [7] introduce the fix-verify pattern where a task is split into multiple generation and review stages. As a popular ontology, DBpedia is a subject to a series of microtask experiments. The authors in [2] introduce a methodology to assess Linked Data quality through the combination of a contest, targeting Linked Data experts [4] and CrowdFlower microtasks. Similar to the works above, in this paper we employ the concept of combining a predicting stage (human, auto, and hybrid) and a verification stage through the CrowdFlower workers subsequently. To improve the crowdsourcing workflow and to ensure high quality answers, various machine learning mechanisms have been recently introduced [25,26,27,28]. Closely to our task, ZenCrowd [8] explore the combination of probabilistic reasoning and crowdsourcing to improve the quality of entity linking. Large scale crowdsourcing can be as well applied in citizen science projects and classification tasks as shown in [29,30]. Machine learning algorithms are often used to support the decision making of adaptive task allocation. In this work, we implement automatic algorithms, similar to that in [19], for a freetext predictor and a naive predictor to facilitate the crowd in locating promising areas within the ontology where the most specific class label for a given entity is likely to be located.

⁷<https://dbpedia-spotlight.github.io/demo/>

⁸<https://dandelion.eu/semantic-text/entity-extraction-demo/>

⁹<http://www.alchemyapi.com/products/demo/alchemylanguage>

¹⁰<http://www.opencalais.com/opencalais-demo/>

¹¹<https://gate.ac.uk/sale/tao/splitch6.html#x9-1360006.7>

¹²<http://nerd.eurecom.fr/>

Another challenge appears in mitigating noisy answers to achieve high quality annotations. Various aspects in microtask design have been studied in the past years. Kittur et al. [31] investigate crowdsourcing micro-task design on Mechanical Turk and suggests that it is crucial to have verifiable questions as part of the task and to design the task in a way that answering the question both properly and randomly would require comparable efforts. The authors in [17] elaborate on the quality of the crowd-generated labels in natural language processing tasks, compared to expert-driven training and gold standard data. Their work suggests that the quality of the results produced by four non-experts can be comparable to the output of a single expert. Some successful citizen science applications such as Galaxy Zoo [32], do not require a high expertise level of the volunteers and the fine granularity of the workflow makes it possible for a task to be carried out through a person without a relevant scientific background. In our work, we leveraged testing questions to eliminate the potential spam user or unqualified user, and designed the task in a way such that only a little user expertise is required. FreeAssociation and Categodzilla/Categorilla [33] tools show a clear empirical evidence that variations, such as less restrictive criteria for user input, can have a positive impact on data quality. In this work we employ a hybrid free-text based predictor where user input for the most specific category is completely unrestricted. The InPhO (Indiana Philosophy Ontology) project [9] use a hybrid approach to dynamically build a concept hierarchy by collecting user feedback on concept relationships from Mechanical Turk and automatically incorporating the feedback into the ontology. The authors in [10] introduce and evaluated the Crowdmap model to convert ontology alignment task into microtasks. This research inspires us to reduce the amount of work for the crowd by using a "predictor" step, providing a shortlist of candidates instead of requiring the worker to explore the full ontology.

This literature is used as a foundational reference for the design of the microtask workflows introduced in this paper, including aspects such as instructions, user interfaces, task configuration, and validation. In contrast to solutions mentioned in the literature, the main novelty of our work lies in the systematic study of a set of alternative workflows for ontology centric entity classification. We believe that such an analysis would be beneficial for a much wider array of Semantic Web and Linked Data scenarios in order to truly understand the complexity of the overall design space and define

more differentiated best practices for the use of crowdsourcing in other real-world projects under different budgetary constraints.

3. Approach

The entity classification problem considered in this paper is a problem of selecting the most specific type from a given class hierarchy for a particular entity. As a class hierarchy can contain thousands of classes, this task is not easy to be solved manually by a few experts maintaining the dataset, especially for large entity batches such as around 1.2M un-typed entities in DBpedia¹³. Automatic and semi-automatic classification is a well-studied area with a variety of probabilistic and discriminative models that can assist in this context. However, whilst there exist a number of possible machine-based classification approaches, they all accept certain error margins of a few dozens of percent and as a result manual correction of their output by an expert remains a heavy overhead. In this section we propose a human computation based model for reducing the amount of the corrections required to be done by an expert.

The problem we are tackling can be formalised as follows: Let O be the given DBpedia ontology and e be a particular entity which has not been classified in DBpedia. Our target is to find the most specific type N_E for entity e within O . For an un-typed entity, we consider this as a tree traversal problem starting from a predicted candidate node in the ontology and continued until the most specific type is found. To this end, as depicted in Figure 1, we break our workflow into (a) *prediction step*, where a list of candidate nodes is identified first, (b) *error detection step* where the output is manually checked and (c) *error correction step* where the error (if detected) is manually corrected. We vary the implementation of each step and measure the performance of our model by two of the most important factors that play into the success of a microtask approach to DBpedia entity typing: precision and costs. Precision refers to the ability of crowd contributors to submit precise answers and our objective is to minimise the costs denoted by the amount of manual work required in a workflow. To be more specific, cost in our framework is the sum of prediction, error detection and error correction effort, measured in terms of the

¹³<http://wiki.dbpedia.org/dbpedia-dataset-version-2015-10>

number of steps to finish the corresponding tasks by crowdworkers.

3.1. The Model

In this section, we formally define our human computation driven model for semi-automatic entity annotation using the hierarchically structured set of labels. This starts with the definition of the terms used in our model, then covers the main tasks where human participation is involved – predictor, the subsequent error detection and error correction process, and our cost model.

Definition 1 (DBpedia Ontology) *The DBpedia ontology \mathcal{O} is a tree structure, where each node corresponds to an entity category and can contain a list of references to the child nodes, each of those corresponds to a more specific sub-category of the parent entity type. The root node **ROOT** ("Thing") is the topmost node in the ontology.*

Definition 2 (Candidate Nodes) *Given an entity e and ontology \mathcal{O} , particular node N is considered as the exact solution node N_E iff N and not any of the children nodes of N is corresponding to the category of e , i.e N represents the most specific category for e in \mathcal{O} . Any ancestor of N_E thereby is considered as a candidate node N_C .*

Definition 3 (Predictor and Predicted Nodes) *Let a predictor P be a (semi-) automatic approach, able to select for given e a ranked set $S_{predicted} = \{N_{p_1}, \dots, N_{p_n}\}$ of nodes from \mathcal{O} as predicted candidates for N_E .*

Our assumption is that the location of an N_P within \mathcal{O} is close to N_E with high probability. In case no prediction can be made by P for a particular e , we consider $S_{predicted} = \{\mathbf{ROOT}\}$, as the **ROOT** is the closest to the optimal solution given the hierarchical structure definition in 1.

3.1.1. Human computation driven error detection and correction model

The error detection and correction process in this context is to traverse \mathcal{O} starting from a set S consisting of top-ranked predicted nodes N_P , until N_E is found and to break the process if no correct solution can be found. We model each traversal step as a microtask for a human assessor, where we ask, whether N_C exists in a list of options. The answer can be *true* (with in-

dication of the corresponding node) or *false*, in case no nodes from the list can be selected. Generally, we assume that if a node is proven to be false, also all of its descendants are proven to be false.

Error detection algorithm: We employ a traversal algorithm 1 with a set S of the top-scored node as a start. In case any of N_P from this set can be identified as N_C , a check is done for each of its child nodes according to the definition 2.

Algorithm 1 Error detection

```

1: procedure CHECKSPECIFICTYPE( $e$ ,
    $S = \{N_{p_1} \dots N_{p_n}\}$ )
2:   if  $N_c \in S$  and  $humanChoice() = N_c$  then
3:     if  $\forall N_i \in children(N_c)$ ,
        $humanChoice() \neq N_i$  then
4:       return TRUE
5:   return FAIL;

```

Error correction algorithm: After the error is detected, it is possible to correct it using human assessors. Similarly to error detection, the microtask based correction algorithm 2 starts from the set S of candidate nodes. In case a N_P from S is identified as N_C , its child nodes are traversed in a breadth-first manner, until the node with the most specific type corresponding to e is found. Otherwise the algorithm continues with the parent node of N_P . Every node on the way is touched only once. The algorithm stops when N_C is found without children corresponding to the type of e . Note, in case no specific node can be found in \mathcal{O} , the algorithm will return **ROOT**.

The overall cost for entity annotation in our framework is constituted as the sum of prediction, error detection and error correction costs, more formally:

$$Cost(annotation) = Cost(prediction) + Cost(detection) + Cost(correction)$$

In which, the costs of a particular algorithm are defined as the number of steps necessary to complete the algorithm run. Cost(detection) and Cost(correction) will be the cost to detect and correct an error, which corresponds to the steps it takes to run the above Algorithm 1 and 2 respectively. The details of the prediction step will be described in the next section.

3.1.2. Predictors

A predictor, in our case, is a module, able to produce a list of candidate classes for a given entity. Currently,

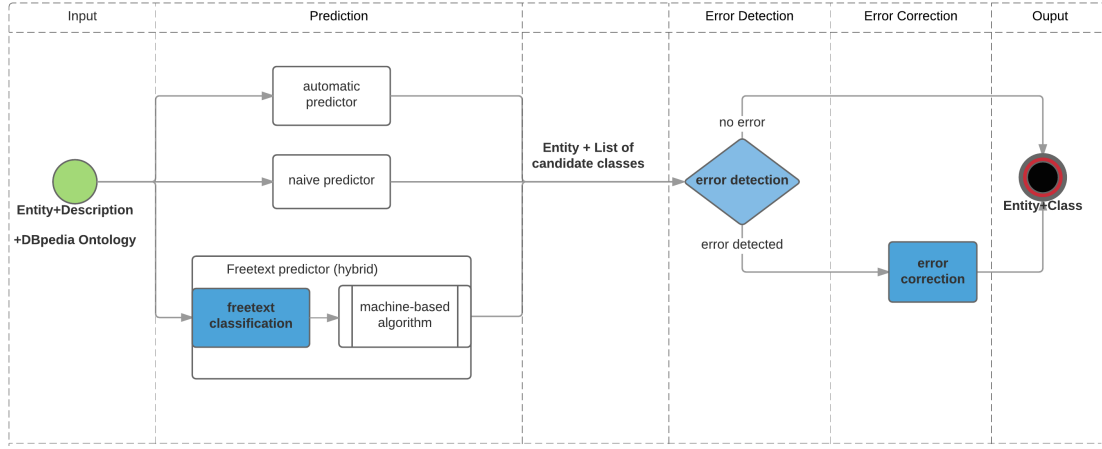


Fig. 1. Representation of three workflows with human participation steps highlighted in blue.

Algorithm 2 Error correction

```

1: procedure FINDSPECIFICTYPE(  $e$ ,
    $S = \{N_{p1} \dots N_{pn}\}$ )
2:   if  $N_c \in S$  and  $humanChoice() = N_c$  then
3:      $C = anyChild(e, N_c)$ 
4:     if  $C = FAIL$  then
5:       return  $N_c$ 
6:     else
7:       return  $C$ 
8:   else
9:      $S_{parents} \leftarrow \{\}$ 
10:    for  $N_i \in S$  do
11:       $S_{parents} \leftarrow S \cup parent(N_i)$ 
12:    return findSpecificType( $e, S_{parents}$ )
13: procedure ANYCHILD( $e, N$ )
14:   for  $N_i \in children(N)$  do
15:     if  $N_i = humanChoice()$  then
16:        $N_{child} = anyChild(e, N_i)$ 
17:       if  $N_{child} = FAIL$  then
18:         return  $N_i$ 
19:       else
20:         return  $N_{child}$ 
21:   return FAIL;

```

all the predictors described in literature are automatic predictors, where given an entity a list of candidates along with the confidence value of the predictions is produced. In this work we employ three approaches, namely an automatic predictor P_{auto} , and two manual predictors P_{naive} and P_{free} as follows:

As an example for P_{auto} , we used DandelionAPI¹⁴ which has an add-on for Google Spreadsheet allowing entities uploaded in a spreadsheet to be analysed and typed with DBpedia types. The entity to be classified is the text to be analysed, and the output we got is the types along with the confidence levels. The types given from Dandelion API always include the parent types if a specific type is identified. For instance, if an entity's specific category is "Building", Dandelion would output Building, ArchitecturalStructure and Place. In this case, we use the open source tool which is free and the prediction cost can be considered as 0.

Additionally, we propose two human computation based prediction approaches. The first P_{naive} is a naive predictor approach that starts from the root of the ontology and traverses the tree by sequentially expanding the children. This approach is very costly, but can be applied also for entities, where any of the other (semi-) automatic approaches fail. Our second human computation based prediction approach P_{free} allows for unconstrained input from the crowd, balancing microtask efficiency and annotation accuracy. The main advantages of this approach are its simplicity and freedom of choice. Classification is one of the most common human cognitive skills and the crowd is not constrained by any unnecessary biases. In addition, restrictions on category input can sometimes increase the difficulty of the task [33] and hence impact the overall accuracy. The outcome depends on how many times the questions are asked and how the answers are aggregated. The more answers are collected, the more reliable is also the classification [34]. Top aggregated an-

¹⁴<https://dandelion.eu/docs/api/>

swers can be voted by the crowd in a second step to identify the most suitable candidate [35], or detect errors and correct the answers, as it is done in our case.

Considering the diversity of vocabulary of the crowd users, direct aggregation of their answers is not effective. To solve this issue, we leverage the freetext input to automatically calculate the closest DBpedia types based on the textual similarity between entity titles and ontology class names. We used the `difflib`¹⁵ `SequenceMatcher` to compute a string match similarity scores between every input collected from the crowd and names off all of existing DBpedia classes. Each DBpedia class is assigned an aggregated score as the indication of how close it is to the user proposed type. We finally retrieve the top scored DBpedia types to form the list of output candidates to be used in error detection and error correction steps.

3.2. Workflows

For the purpose of this study, we identify three types of workflows, following the preliminary considerations presented so far. Each workflow incorporates a predictor, as well as the error detection and correction steps.

- (W_{naive}) **Naive**: ask the crowd to choose the DBpedia category by traversing from the root top-down until a specific category is selected. This is the workflow that employs P_{naive} predictor.
- (W_{free}) **Freetext**: accepts unconstrained input from the crowd to label an entity. This workflow incorporates the P_{free} predictor where collected freetext annotations are processed to identify the top candidates in DBpedia types which then can be explored and corrected by the crowd in a follow-up verification task.
- (W_{auto}) **Automatic**: uses an entity typing tool to generate candidate list, then ask the crowd to detect and correct errors. This workflow is based on the P_{auto} predictor.

Figure 1 not only shows an overview of these three workflows, but also highlights the steps requiring human participation in blue background. In the remainder of this section we explain the workflows and their translation into microtasks in more detail.

For the (W_{naive}) **Naive** workflow, the idea is to traverse the DBpedia class hierarchy from top-down. The particular worker choice from a list depends on

her level of expertise and on the given situation, as the theory teaches. Rosch et al. proved in experiments that experts and newbies make very different classification decisions - people with little insight into a domain tend to feel comfortable with categories that are neither too abstract, not too specific, whereas experts are much more nuanced [36]. The same effect was observed by [37] or in games with a purpose [38]. As we are working with microtask platforms, we have to assume that the behaviour of the crowd contributors will resemble newbies in the Rosch et al. experiments and casual gamers who interacted with GWAPs. Thus, we cannot always expect from P_{naive} to identify the most specific class in the DBpedia ontology, which matches the input entity. This is also why in each list of options we presented to the crowd, we need to include a `NoneOfAbove` option. In this Naive workflow, the crowd is firstly presented with a list of candidate types (first level of DBpedia classes under the `ROOT`, along with `NoneOfAbove`) . As there are 49 first level classes in DBpedia, using the naive approach, the classes at each level are randomly put into groups and one such group is presented at a time. The details on decision on the number of options to be put in each group are available in task design section 3.4). If `NoneOfAbove` is chosen, another group of options is presented to the worker. If any given level 1 class is chosen and the class has children, a group candidate types from level 2 class will be displayed for the worker instead. The process keeps going until a class which has no children is chosen, or a chosen class does not contain children that are suitable specific types for the given entity. The result of this process is considered as final.

The (W_{free}) **Freetext** workflow starts with a task where an entity and its description are presented to the worker and free text annotation from this worker are collected. Once we collected annotations for all entities, for each entity, we consider all the annotations and calculate their textual similarity (many available APIs^{16,17,18}) with titles of all existing DBpedia classes(vocabularies) and add-up a similarity score for each DBpedia classes. The higher the similarity score of a DBpedia class has, the higher is the chance that this class is the candidate type of the corresponding entity. Then we pick the top classes for each entity and add a "NoneOfAbove" option to form a shortlist as the predicted result from P_{free} . Then we start the error de-

¹⁵<https://docs.python.org/2/library/difflib.html>

¹⁶<https://pypi.python.org/pypi/Distance/>

¹⁷<https://docs.python.org/2.7/library/difflib.html>

¹⁸<http://www.nltk.org/howto/wordnet.html>

tection and correction process. Similar to (W_{naive}), we traverse the DBpedia class hierarchy until a class which has no children is chosen, or a class has been chosen, but none of its children are suitable specific types for the given entity. In the case when none of predictor predicted classes is chosen as a suitable type for the given entity, we will traverse up to the parent level. Depending on whether a suitable class is chosen from the parent level or not, further child or parent classes will be presented respectively. The process keeps going until it reaches ROOT or a class is chosen and that class either has no children or none of its children are suitable types for the given entity.

For the (W_{auto}) **Automatic** workflow, the idea is to use one of existing APIs^{19,20,21} to produce a list of candidate types for the given entities, and only the top types with higher confidence level are picked to be presented along with the *NoneOfAbove* option to the crowd. Based on worker's selection, the similar traverse process is followed until it reaches ROOT or a class is chosen and that class either has no children or none of its children classes is chosen.

3.3. Microtask Design

In general, we distinguish between two types of microtasks based on their output: **T1** where the workers produce free text output; and **T2** where the worker can choose from a list of classes. In both cases, we generate descriptions of the input entities in the form of labels or text summaries. Either way, the effort to generate the first type of tasks (**T1**) is comparatively lower, as there is no need to compute a list of potential candidates. However, this is compensated by overhead in sense making of the output and means to aggregate free text inputs into meaningful class suggestions, while in **T2** the answer domain is well-defined. Crowdsourcing literature recommends to use iterative tasks to deal with **T1** scenarios: in a first step the crowd is generating suggestions, while in the second it is asked to vote on the most promising ones [39,40,35]. Accordingly, **T2** can be used as the voting step for **T1** output to get more useful suggestions in the condition **T1** output has been post-processed to produce a ranked list, such as using the aggregation we proposed in 3.1.2 for P_{free} .

The **T2** variant requires the strategy to generate the list of candidates. It can be achieved through auto-

matic tools addressing a similar task (as listed in section 2.1), however, they impose clear bounds on the accuracy of the crowd experiments, as not all input can be processed by the automatic tools (recall) and their output can only be as precise as the task input allows (precision). Additionally, the choice of an appropriate threshold to accept or reject produced results is also a subject of debate in related literature [41,42]. Another option is to provide the crowd all possible choices in a finite domain - in our case all classes of the DBpedia ontology. The challenge then is to find a meaningful way for the crowd to explore these choices. While classification is indeed one of our most common human skills, cognitive psychology established that people have trouble when too many choices are possible. This means both, too many classes to consider [43,42] and too many relevant criteria to decide whether an item belongs to a class or not [44,45]. Miller's research suggests that in terms of the capacity limit for people to process information, 7 (plus or minus 2) options are a suitable benchmark [46]. We hence would ideally use between 5 and 10 classes to choose from. This situation is given by the predictor P_{auto} , as the confidence of automatic entity typing algorithms decreases rapidly and only the top 10 are likely to be representative. However, in the P_{naive} condition, we start from the DBpedia ontology, which has over 700 classes to choose from²². The problem persists even when browsing the ontology level by level, as some classes have tens of subclasses (e.g., there are 21 different forms of organization, 50 child categories of person and 43 specific types of athlete). In our experiments, therefore, we split the list into subsets of 7 items or less to display per microtask.

3.4. Implementation

As shown in section 3.1, all three workflows take the same input. As a result, for each workflow we first queried the DBpedia endpoint via SPARQL to obtain the name, description, and a link to Wikipedia of each entity. W_{naive} and W_{auto} workflows only involve **T1** task type, W_{free} include both **T1** and **T2** in sequence. The Figure 2 and 3 depict the user interfaces for **T1** and **T2** types of tasks. For **T2** we have always limited the number of options shown to a user in the list to maximum of 7 (6 class candidates and an *NoneOfAbove* option). In case there are more candi-

¹⁹<https://dandelion.eu/docs/api/>

²⁰<http://www.alchemyapi.com/api/entity/types>

²¹<http://www.opencalais.com/opencalais-api/>

²²<http://mappings.dbpedia.org/server/ontology/classes/>
(accessed on Jan 14, 2016)

dates, we split the list as discussed earlier. We created the gold standard data (see Section 4), set the required CrowdFlower parameters, launched the jobs, and monitored their progress.

3.4.1. CrowdFlower Parameters

Task: Following the advice from the literature [2], we used 5 units/rows for each task/page for each of the three workflows. The worker completes a task by categorising five entities.

Judgment: Snow et al. [17] claim that answers from an average of four non-experts could achieve a level of accuracy parallel to NLP experts on Mechanical Turk. We hence asked for 5 judgments per experimental data throughout our experiments. In W_{free} , we asked for 11 free text suggestions. This choice is in line with experiments from the literature [2,10]. We also did an empirical simulation with our previous experiments on entities that have been classified to find out the effective number of judgement to use. Figure 4 shows the number of times the entity has been annotated and the corresponding matched answers (based on 120 entities).

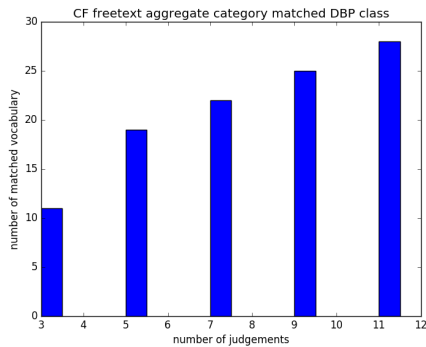


Fig. 4. Freetext: Judgement Number vs. Matched Answers

Payment: We paid 6 cents for each task consisting of 5 units. This setting took into account existing surveys [47], as well as related experiments which have similar complexity level. Tasks like reading an article and then asking the crowd to rate the article based on given criteria as well as providing a suggestion for the areas to be improved were paid 5 cents [31]. Tasks that have smaller granularity such as validating the given linked page display relevant image to the subject were paid 4 cents per 5 units [2]. In a similar vein, the complexity of our classification task is somewhere in between considering the time and knowledge it requires to complete the task.

Quality control: We created test questions for all jobs and required contributors to pass a minimum accuracy rate (we use 50%) before working on the units.

Contributors: CrowdFlower distinguishes between three levels of contributors based on their previous performance. The higher level of contributors required, the longer it takes to finish the task, but might be with higher quality. In our experiment, we choose the default Level1 which allow all levels of contributors to participate in the classification task. We used this level for all three workflows.

Aggregation: For the **T2** tasks we used the default option (aggregation='agg')²³, as the task is to choose from a set of pre-defined options. For **T1**, we looked at the first three answers (aggregation='agg_3') based on 11 judgments. We also used aggregation='all' to collect the additional categories when the crowd felt the best match was not listed among the suggestions.

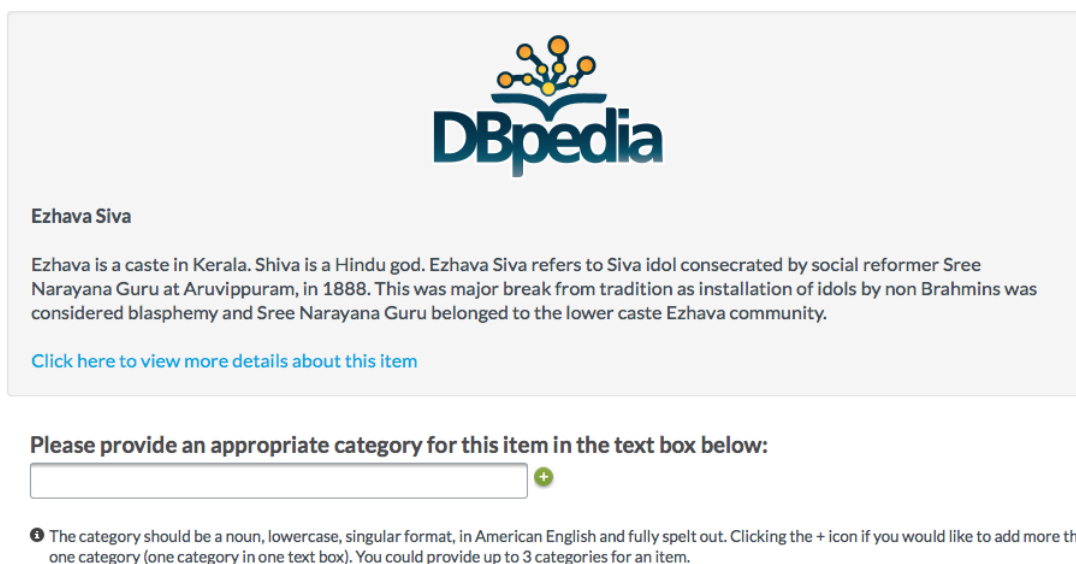
3.5. Using the Crowdsourced Data


The validated and aggregated results may be leveraged in several ways in the context of DBpedia. Freetext suggestions signal potential extensions of the DBpedia ontology (concepts and labels) and of the DBpedia Extraction Framework (mappings). Applying the freetext workflow gives insights into the limitations of entity typing technology, while the way people interact with the naive workflow is interesting not only because it provides hints about the quality imbalances within the DBpedia ontology, but also for research on Semantic Web user interfaces and possible definition of new mapping rules for entity typing.

4. Evaluation

In this section, we will evaluate the performance of proposed workflows as depicted in Figure 1. Overall, we obtained three different workflows, namely: W_{naive} , W_{auto} and W_{free} , respectively based on predictors P_{naive} , P_{auto} and P_{free} . We evaluate the accuracy of the predictors and compare the overall precision and costs of different workflows.

²³<https://success.crowdfLOWER.com/hc/en-us/articles/203527635-CML-Attribute-Aggregation>






Ezhava Siva

Ezhava is a caste in Kerala. Shiva is a Hindu god. Ezhava Siva refers to Siva idol consecrated by social reformer Sree Narayana Guru at Aruvippuram, in 1888. This was major break from tradition as installation of idols by non Brahmins was considered blasphemy and Sree Narayana Guru belonged to the lower caste Ezhava community.

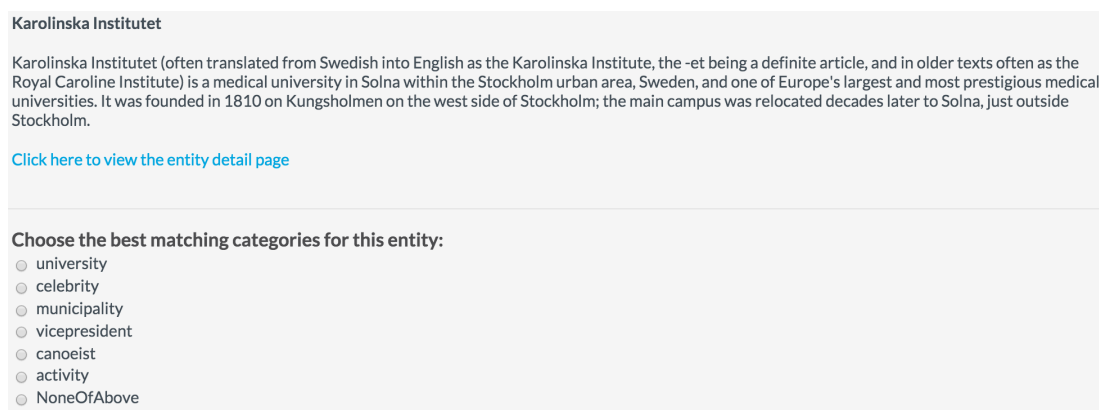
[Click here to view more details about this item](#)

Please provide an appropriate category for this item in the text box below:



i The category should be a noun, lowercase, singular format, in American English and fully spelt out. Clicking the + icon if you would like to add more than one category (one category in one text box). You could provide up to 3 categories for an item.

Fig. 2. The user interface for **T1** tasks based on free text input



Karolinska Institutet

Karolinska Institutet (often translated from Swedish into English as the Karolinska Institute, the -et being a definite article, and in older texts often as the Royal Caroline Institute) is a medical university in Solna within the Stockholm urban area, Sweden, and one of Europe's largest and most prestigious medical universities. It was founded in 1810 on Kungholmen on the west side of Stockholm; the main campus was relocated decades later to Solna, just outside Stockholm.

[Click here to view the entity detail page](#)

Choose the best matching categories for this entity:

- university
- celebrity
- municipality
- vicepresident
- canoelist
- activity
- NoneOfAbove

Fig. 3. The user interface for **T2** tasks based on a list of candidate options

4.1. Data

For our experiments we uniformly at random choose 120 entities which were not classified so far by the DBpedia. The authors of the paper annotated these entities manually to obtain a gold standard. The annotators worked independently and achieved an agreement of 0.72 measured using Cohen's kappa. According to one of the most commonly used interpretation by Landis and Koch(1977), kappa value in the range of 0.6-0.8 corresponds to a substantial agreement. Noted we are able to get 106 out of 120 entity typings agreed between two annotators. To achieve consensus, the annotators then collaboratively defined a set of rules to

categorize the entities whose classes did not match and involved a third annotator for majority voting calculation. For example, an entity such as "List of Buffy the Vampire Slayer novels" was eventually classified as List instead of Novel, while the "Haute Aboujagane, New Brunswick", which describes a specific community, was defined as Community instead of GovernmentAdministrativeRegion.

Table 1 provides the overview on the composition of the resulting gold standard corpus with respect to general categories. "Place" and "Organisation" are the most present categories with respectively 20 and 16 entities in the dataset, corresponding to one of their child categories. For 18 entities that no appropriate category

#Categories	20	18	16	10	9	7	5	4	2	1
Categories	Place	owl:Thing	Organisation	Work Agent	TopicalConcept	Group	List	EthnicGroup	Company PersonFunction Device UnitOfWork Food Species ChemicalSubstance	Name Activity MeanOfTransportation Medicine EducationalInstitution Event Language

Table 1
Overview of the EI Corpus

in the ontology could be found by the experts, those were labelled as "owl:Thing".

4.2. Experiments

For each workflow we assess their properties in our experiments with respect to quality, effectiveness and costs. The accuracy of the predictor matching our gold standard will provide information about the predictor effectiveness for real applications. The accuracy of the human based error correction with respect to our gold standard will provide insights into what quality can be expected from the crowd in general.

4.2.1. Prediction Quality

In our quality concerned experiments, we employ precision which measures the portion of correctly classified entities among all entities classified by the predictor. Our quality measures are the precision-recall curves as well as the precision-recall break-even points for these curves. The precision-recall graph provides insights in how much precision it is possible to gain in case only a certain amount of recall is required. Typically, the head of the distribution achieves better precision due to the fact that the values correspond to the higher confidence of the predictor. The break-even point (BEP) is the precision/recall value at the point where precision equals recall, which is equal to the F1 measure, the harmonic mean of precision and recall, in that case. The results of the experiments for predictors described in Section 3.1.2 are shown in Figure 5. The main observations are:

The precision of the human computation based methods was generally higher when compared to the automatic method. Also the automatic method did not produce any results for around 80% of the entities, whilst human computation based predictor provided suggestions for all entities with certain quality. P_{naive} was showing the best precision over the test set of BEP

	W_{auto}	W_{naive}	W_{free}
Prediction costs	0	4.1	1
Error detection costs	2.9	0	1.6
Error correction costs	3.2	0	1.83
Sum	6.1	4.1	4.43

Table 2
Cost overview for different workflows

0.49, followed by P_{free} with BEP 0.47. Surprisingly, the precision of P_{naive} was lower than expected for a completely human computation based solution and may indicate that the classification task requires high expertise and cannot rely on crowd alone. For about 10% of recall all methods provide very good results, especially the precision for P_{auto} and P_{free} was very high, making further steps in error detection and correction possibly unnecessary.

We standardized the confidence level of each predictor to the range [0.0-1.0] and plotted the output quality at different confidence levels in terms of precision in Figure 6. As expected from previous results, high confidence levels above 0.9 for P_{auto} and P_{free} indicate high quality results. For confidence levels below, the output cannot be trusted and need error correction. To improve the prediction, quality research needs to be done to develop better predictors, or improve the quality existing ones.

4.2.2. Prediction costs

In comparison to P_{auto} , the predictors P_{free} and P_{naive} required human input and therefore added additional costs. Whilst in P_{free} the user had to execute exactly one free text task per entity, to obtain the results using P_{naive} , the crowd worker had to complete 4.1 tasks on average.

4.2.3. Error Detection Quality and Costs

Having received the output of a predictor, we tested whether the crowd was able to identify prediction er-

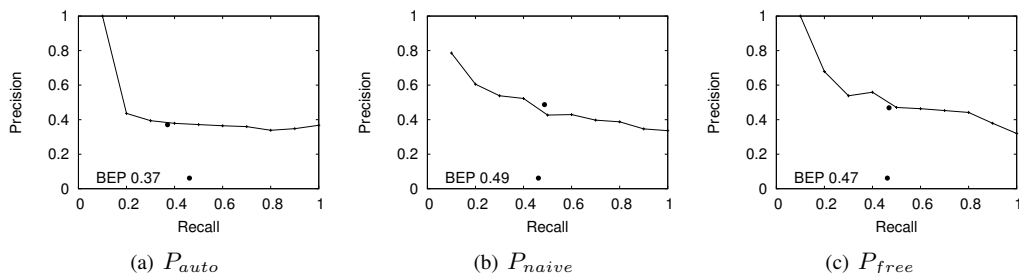


Fig. 5. Precision-recall curves for the output of the different predictors with respect to our gold standard.

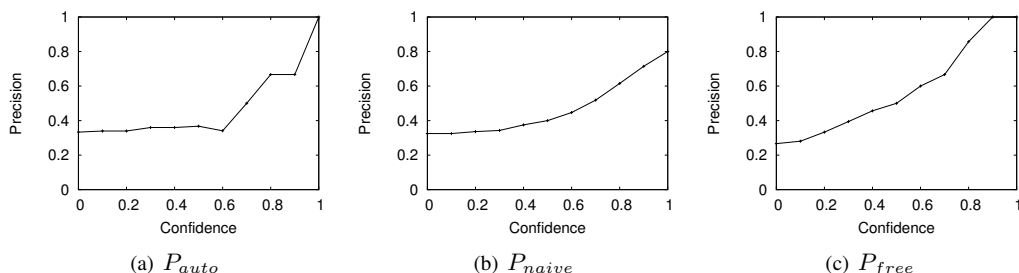


Fig. 6. Precision at confidence level curves for the output of the different predictors with respect to our gold standard.

rors and empirically determined the costs for this detection in terms of the number of questions needed to be answered on average. The graphs provide insights into the quality of the predictor outputs with respect to crowd decisions and can provide decision support to include or exclude the error detection step in real applications. We do not test error detection for P_{naive} as its output is already based on crowd and cannot be expected to improve with the error detection or correction step. As defined in our crowdsourcing model, at each task we show 7 candidate options to the user. On average, error detection took 2.9 detection steps with W_{auto} and 1.6 steps with W_{free} as depicted in Table 2.

4.2.4. Error Correction Quality and Costs

In the last step, we apply our algorithm to correct the errors produced by the predictors. Similar to the previous section, we measure the costs for the correction as the number of questions needed to be answered on average. The Table 2 provides an overview of the costs. W_{auto} required on average 3.2 steps to correct the prediction due to the fact that the predictor did not produce any results for the most entities and the whole tree had to be traversed to find the answer in such cases. 1.83 steps were required on average for W_{free} , indicating that in general the predictor pointed the user to the right area within the ontology. In summary, as depicted in Table 2, W_{auto} appeared the most costly

with 6.1 steps on average and the other two workflows showed comparable results.

4.2.5. The Quality of Human Output

Finally, we measure the quality of the workflows as a whole to estimate the effort to be invested by experts in post-processing. The Figure 7 shows the precision-recall curves of the human based result correction for W_{auto} and W_{free} . We observe that in both workflows the result improved when compared to the prediction step alone. Our proposed workflow W_{free} reached a BEP of 0.53 - the highest result among all experiments.

4.2.5.1 Crowdsourcing Tasks

We decided to limit the number of top-options shown to the user to 7 as recommended in the literature. Longer lists may contain the correct result with higher probability, however would also require more interaction and effort in complexity for a crowd worker. To show the possible influence of the option number on the prediction quality, we plot the correspondence in Figure 8 where we vary the list size on the logarithmic scale from top 1 to the maximum of 740 possible categories and measure the predictor BEPs' on the basis of the gold standards. As we can observe, the precision improves only slightly with the growing list size, indicating no meaningful advantage for lists with more than 7-10 options.

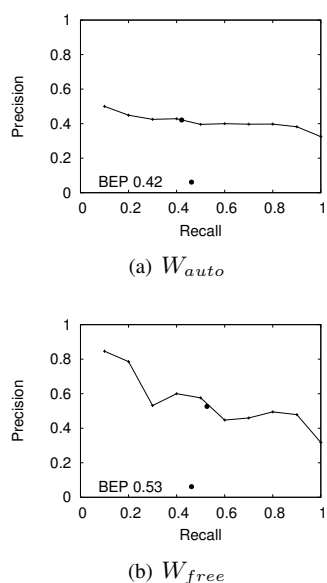


Fig. 7. The overall output quality of the workflows.

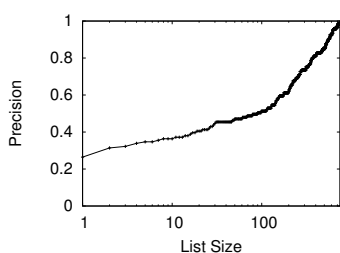


Fig. 8. Precision at different Options List Size with the W_{free} Predictor

5. Discussion and Lessons Learnt

5.1. Are unclassified entities unclassifiable?

As noted earlier, there were significant differences in the performance scores achieved in the experiments using different workflows. Especially notable is that existing NLP tool could only classify 45% of the randomly selected entities (54 out of 120) with a precision of only 0.37. Human-based approach can achieve relatively higher 0.49 to 0.47 precision, but still a large portion of the entities are not classified to the correct specific type. To some extent, this indicates that untyped entities have certain characteristics which make them difficult to be classified.

- Firstly, we observed that their types are quite diverse and not within the most popular classes²⁴. For instance, "Standard", "SystemOfLaw", or "TopicalConcept" are not categories non-expert could easily distinguish.
- Secondly, the imbalanced structure of DBpedia ontology also makes the classification of untyped entities whose boundary between subtle categories are not well defined. For example, "Hochosterwitz Castle" is a "Castle" which would be the most specific category for this entity, however, Castle is a child category of Building, which is a child type of ArchitecturalStructure that has many child types such as Arena, Venue and Pyramid, leading the user to choose none of the children of "ArchitecturalStructure" as they did not see any fits. Similarly, among all 18 entities that are instances of "Place", only 5 of them are correctly classified to the most specific category because of the unclear and over-defined sub-categories. "Place" and "Area" are both immediate first level types under "owl:Thing", which create a lot confusion in the first place as we observed from the crowd contributed classification. Also categories such as "Region", "Locality" and "Settlement" are difficult to be differentiated.
- Lastly, ambiguous entities unsurprisingly caused disagreement [48]. This was the case with "List" and specific types such as the "1993 in Film",²⁵ which is an List (not a film), and the "1976 NBA Finals",²⁶ which is rather a Tournament (child of "Event", "SocietalEvent" and "SportsEvent", but not a "List"). In general, entities like these contain a context which sometimes makes the entity itself ambiguous. In the similar way, "Provinces of the Dominican Republic"²⁷ is a list (not a place) while "Luxembourg at the Olympics" is a sports team. In another case, an entity with context is just difficult to fit in any existing DBpedia types. For instance, "Higher education in Hong Kong" and "Petroleum industry in Nigeria".

5.2. The outputs are only as good as the inputs

Taking naive workflow where we present maximum of 7 types (including a "NoneOfAbove" option) in one

²⁴<http://wiki.dbpedia.org/services-resources/datasets/data-set-39/data-set-statistics>

²⁵https://en.wikipedia.org/wiki/1993_in_film

²⁶https://en.wikipedia.org/wiki/1976_NBA_Finals

²⁷https://en.wikipedia.org/wiki/Provinces_of_the_Dominican_Republic

step as an example, the aggregated outcome shows that 33 entities are categorised as "other" after traversing the DBpedia class tree top-down from "owl:Thing", with none of the DBpedia categories being chosen. This also contributes to the ongoing debate in the crowdsourcing community regarding the use of miscellaneous categories [48,49]. In our case, using this option elicited a fair amount of information, even if it was used just to identify a problematic case. [49] discuss the use of instructions as a means to help people complete an entity typing task for microblogs. In our case, however, we believe that performance enhancements would be best achieved by studying the nature of unclassified entities in more depth and looking for alternative workflows that do not involve automatic tools in cases which we assume they will not be able to solve. A low hanging fruit is the case of containers such as lists, which can be identified easily. For the other cases, one possible way to move forward would be to compile a list of entity types in the DBpedia ontology, which are notoriously difficult, and ask the crowd to comment upon that shortlist instead of the one more or less 'guessed' by a computer program. Another option would be to look at workflows that involve different types of crowds. However, it is worth mentioning that for the 120 randomly chosen untyped entities from DBpedia, 18 of them don't fit in any DBpedia types based on our gold standard which indicate there is a need to enhance the ontology itself.

5.3. Popular classes are not enough

As noted earlier, entities which do not lend themselves easily to any form of automatic classification seem to be difficult to handle by humans as well. This is worrying, especially if we recall that this is precisely what people would expect crowd computing to excel at, enhancing the results of technology. However, we should also consider that a substantial share of microtask crowdsourcing applications addresses slightly different scenarios: (i) the crowd is either asked to perform the task on their own, in the absence of any algorithmically generated suggestions; or (ii) it is asked to create training data or to validate the results of an algorithm, under the assumption that those results will be meaningful to a large extent. The situation we are dealing with here is fundamentally new because the machine part of the process does not work very well and distorts the wisdom of the crowds. These effects did not occur when we used free annotations. An en-

tity such as "Brunswick County North Carolina"²⁸ is an obviously a "County" and a child type of "Place". Freetext approach actually proposes this category, although that category does not exist in DBpedia yet. This result is consistent with [33].

It became evident that in case the predicted categories are labelled in domain-specific or expert terminology, people tend not to select them. While under the unbound condition they are comfortable using differentiated categories, the vocabulary has to remain accessible. For example, Animal (sub-category of Eukaryote) is used more than Eukaryote. In all three workflows, if the more general category is not listed, participants were inclined towards the more specialized option rather than higher-level themes such as Person, Place, and Location. This could be observed best in the freetext workflow. Such aspects could inform recent discussions in the DBpedia community towards a revision of the core ontology.

5.4. Spam prevention

It has been observed that crowdsourcing microtasks sometimes generate noisy data which either is submitted deliberately from lazy workers or from the crowd whose knowledge of the task area is not sufficient enough to meet certain accuracy criteria. Test question is a good way to help minimize the problems caused by both cases such that only the honest worker with basic understanding are involved in the tasks. In our experiment, we did not specially use control question to prevent spam, instead we use test question to recruit qualified workers. Although the test question approach requires about 10% additional judgments to be collected, it does give good inputs in which the definite spam is rare and negligible.

6. Conclusion and Future Work

In this paper, we are the first to propose a crowdsourcing based error detection and correction workflows for (semi-) automatic entity typing in DBpedia with selection of the most specific category. Though our framework employs DBpedia ontology, in principle it can also be applied to other classification problems where the labels are structured as a tree. In our second contribution we propose a new microtask based

²⁸https://en.wikipedia.org/wiki/Brunswick_County,_North_Carolina

free text approach for the prediction of entity type. This predictor provides good results even for entities where automatic machine learning based approach fails and naive full ontology scans are necessary. We empirically evaluated the quality of the proposed predictors as well as the crowd performance and costs for each of the workflows using 120 DBpedia entities that are chosen uniformly at random from the set of entities, not yet annotated by the DBpedia community.

While upper levels of the DBpedia ontology seem to match well the basic level of abstraction coined by Rosch and colleagues more than 30 years ago [36], contributors used more specific categories as well, especially when not being constrained in their choices to the (largely unbalanced) DBpedia ontology. The experiments also call for more research into what is different about those entities which make the hard cases and in our discussion we gave some suggestions for improvement.

In our future work we plan to investigate how the quality of semi-automatic predictors can be further improved to reduce the costs for correction to a minimum. More work can be done in the design of the microtasks especially in terms of the option choice displayed to the user. Finally, there is a lot to be done in terms of support for microtask crowdsourcing projects. The effort invested in our experiments could be greatly reduced if existing platforms would offer more flexible and richer services for quality assurance and aggregation (e.g., using different similarity functions).

References

- [1] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2014.
- [2] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. *The Semantic Web—ISWC 2013*, pages 260–276, 2013.
- [3] P Kreis. Design of a quality assessment framework for the dbpedia knowledge base. *Master’s thesis, Freie Universität Berlin*, 2011.
- [4] Amrapali Zaveri, Dimitris Kontokostas, Mohamed A Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 97–104. ACM, 2013.
- [5] Katharina Siorpaes and Elena Simperl. Human intelligence in the process of semantic content creation. *World Wide Web*, 13(1-2):33–59, 2010.
- [6] Dimitris Kontokostas. Making sense out of the wikipedia categories (gsoc2013). [wiki.dbpedia.org](http://wiki.dbpedia.org/blog/making-sense-out-wikipedia-categories-gsoc2013), 2013. URL <http://wiki.dbpedia.org/blog/making-sense-out-wikipedia-categories-gsoc2013>. [Accessed: 15 May 2016].
- [7] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soyent: a word processor with a crowd inside. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 313–322. ACM, 2010.
- [8] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*, pages 469–478. ACM, 2012.
- [9] Kai Eckert, Mathias Niepert, Christof Niemann, Cameron Buckner, Colin Allen, and Heiner Stuckenschmidt. Crowdsourcing the assembly of concept hierarchies. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 139–148. ACM, 2010.
- [10] Cristina Sarasua, Elena Simperl, and Natalya F Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *The Semantic Web—ISWC 2012*, pages 525–541. Springer, 2012.
- [11] Cheng Niu, Wei Li, Jihong Ding, and Rohini K Srihari. A bootstrapping approach to named entity classification using successive learners. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 335–342. Association for Computational Linguistics, 2003.
- [12] Yu-Chieh Wu, Teng-Kai Fan, Yue-Shi Lee, and Show-Jane Yen. Extracting named entities using support vector machines. In *Knowledge Discovery in Life Science Literature*, pages 91–103. Springer, 2006. ISBN 3540328092.
- [13] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIG-DAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110. Citeseer, 1999.
- [14] Jae-Ho Kim, In-Ho Kang, and Key-Sun Choi. Unsupervised named entity classification models and their ensembles. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.
- [15] Joana Costa, Catarina Silva, Mário Antunes, and Bernardete Ribeiro. On using crowdsourcing and active learning to improve classification performance. *International Conference on Intelligent Systems Design and Applications, ISDA*, pages 469–474, 2011. ISSN 21647143. .
- [16] Edwin Simpson, Stephen Roberts, Ioannis Psorakis, and Arfon Smith. Dynamic bayesian combination of multiple imperfect classifiers. *Studies in Computational Intelligence*, 474:1–35, 2013. ISSN 1860949X. .
- [17] Rion Snow, Brendan O’Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.
- [18] Luis Von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006. ISSN 0018-9162.

- [19] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012. ISSN 2150-8097.
- [20] Babak Loni, Jonathon Hare, Mihai Georgescu, Michael Riegler, Xiaofei Zhu, Mohamed Morchid, Richard Dufour, and Martha Larson. Getting by with a little help from the crowd: Practical approaches to social image labeling. *Proceedings of the 2014 International ACM Workshop on Crowdsourcing for Multimedia*, pages 69–74, 2014. .
- [21] Jonathon S Hare, Maribel Acosta, Anna Weston, Elena Simperl, Sina Samangooei, David Dupplaw, and Paul H Lewis. An Investigation of Techniques that Aim to Improve the Quality of Labels provided by the Crowd. In *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop, Barcelona, Spain, October 18-19, 2013.*, volume 1043 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.
- [22] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622, Las Vegas, Nevada, USA, 2008.
- [23] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Large-scale linked data integration using probabilistic reasoning and crowdsourcing. *The VLDB Journal*, 22(5):665–687, 2013. ISSN 1066-8888. .
- [24] Dafna Shahaf and Eric Horvitz. Generalized Task Markets for Human and Machine Computation. In *AAAI*, 2010.
- [25] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP '10*, page 64, 2010. ISSN 145030222X. .
- [26] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G. Ipeirotis, and Philippe Cudré-Mauroux. The Dynamics of Micro-Task Crowdsourcing: The Case of Amazon MTurk. pages 238–247, may 2015.
- [27] Stephanie L. Rosenthal and Anind K. Dey. Towards maximizing the accuracy of human-labeled sensor data. *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10*, page 259, 2010. .
- [28] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. *Advances in Neural Information Processing Systems*, 22(1):1–9, 2009.
- [29] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.
- [30] Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proceedings of the VLDB Endowment*, 7(13):1529–1540, 2014. ISSN 2150-8097.
- [31] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.
- [32] Galaxy Zoo. Galaxy zoo. <http://zoo1.galaxyzoo.org/>, 2007.
- [33] David Vickrey, Aaron Bronzan, William Choi, Aman Kumar, Jason Turner-Maier, Arthur Wang, and Daphne Koller. Online word games for semantic data collection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 533–542. Association for Computational Linguistics, 2008.
- [34] Chris J Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, et al. Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.
- [35] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. TurkIt: human computation algorithms on mechanical turk. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 57–66. ACM, 2010.
- [36] Eleanor Rosch, Carolyn B Mervis, Wayne D Gray, David M Johnson, and Penny Boyes-Braem. Basic objects in natural categories. *Cognitive psychology*, 8(3):382–439, 1976.
- [37] James W Tanaka and Marjorie Taylor. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive psychology*, 23(3):457–482, 1991.
- [38] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.
- [39] Andrew W Brown and David B Allison. Using crowdsourcing to evaluate published scientific literature: methods and example. *PLoS one*, 9(7):e100647, 2014.
- [40] Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1999–2008. ACM, 2013.
- [41] Benjamin Scheibehenne, Rainer Greifeneder, and Peter M Todd. What moderates the too-much-choice effect? *Psychology & Marketing*, 26(3):229–253, 2009.
- [42] Barry Schwartz. The paradox of choice. Ecco, 2004.
- [43] Sheena S Iyengar and Mark R Lepper. When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology*, 79(6):995, 2000.
- [44] Leola A Alfonso-Reese, F Gregory Ashby, and David H Brainard. What makes a categorization task difficult? *Perception & Psychophysics*, 64(4):570–583, 2002.
- [45] Jianping Hua, Zixiang Xiong, James Lowey, Edward Suh, and Edward R Dougherty. Optimal number of features as a function of sample size for various classification rules. *Bioinformatics*, 21(8):1509–1515, 2005.
- [46] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
- [47] Panagiotis G Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.
- [48] Lora Aroyo and Chris Welty. Crowd truth: Harnessing disagreement in crowdsourcing a relation extraction gold standard. *Web Science '13*, 2013.
- [49] O Feyisetan, E Simperl, M Luczak-Roesch, R Tinati, and N.Shadbolt. Towards hybrid NER: a study of content and crowdsourcing-related performance factors. In *Proceedings of the ESWC2015 (to appear)*, 2015.