Towards Fuzzy Query-Relaxation for RDF

Aidan Hogan[1], Marc Mellotte[1], Gavin Powell[2], and Dafni Stampouli[2]

[1] Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, Ireland. *Email:* {aidan.hogan,marc.mellotte}@deri.org
[2] Innovation Works, European Aeronautic Defence and Space Company (EADS), Newport, UK. *Email:* {gavin.powell,dafni.stampouli}@eads.com

Abstract. In this paper, we argue that *query relaxation* over RDF data is an important but largely overlooked research topic: the Semantic Web standards allow for answering crisp queries over crisp RDF data, but what of use-cases that require approximate answers for fuzzy queries over crisp data? We introduce a use-case from an EADS project that aims to aggregate intelligence information for police post-incident analysis. Query relaxation is needed to match incomplete descriptions of entities involved in crimes to structured descriptions thereof. We first discuss the use-case, formalise the problem, and survey current literature for possible approaches. We then present a proof-of-concept framework for enabling relaxation of structured entity-lookup queries, evaluating different distance measures for performing relaxation. We argue that beyond our specific scenario, query relaxation is important to many potential use-cases for Semantic Web technologies, and worthy of more attention.

1 Introduction

RDF is a flexible data format, and is well-suited to data integration scenarios. However, specifying precise queries over integrated, incomplete, heterogeneous data is much more challenging than likewise in closed, homogeneous settings. Writing precise queries requires precise knowledge of the modelling and content of the data. Even if a querying agent knows its exact information needs—and is able to specify those needs in a crisp, structured request—often, the query will not align well with heterogeneous data. Furthermore, a querying agent may not be able to specify the precise "scope" of answers it is interested in, but may instead only be able to specify some "ideal" criteria that would be desirable.

Current Semantic Web standards and tools only go so far towards matching the needs of the querying agent and the content of the dataset. RDFS and OWL only facilitate finding more *crisp answers* to queries—answers matched directly by the data or its entailments—and do not directly support a continuous notion of *distance* (or *similarity*) for resources. For example, a query asking for a `2012 blue sports car` *on sale in* `New York` may also be interested in a `2010 navy roadster` *on sale in* `Newark`. Although the subsumption relationship between a sports car and a roadster could be modelled in RDFS, the distance of resources such as blue/navy (vs. blue/red) and New York/Newark (vs. New York/Los

Angeles) cannot be succinctly axiomatised in RDFS/OWL for interpretation by the query answering system. Instead, we argue that the RDF descriptions of resources can be used to compute an inductive, generic notion of distance.

In this paper, we thus advocate a relaxed form of RDF query-answering where the query should be interpreted as specifying the *ideal* criteria for answers, such that other relevant (but non-crisp) scored answers are returned. This is similar to top-k query-answering for Information Retrieval engines: a paradigm that works well in highly-heterogeneous, incomplete scenarios, including Web search.

We first present an industrial use-case from the European Aeronautic Defence and Space Company (EADS) that requires matching witness observations against crisp knowledge integrated from various law-enforcement and intelligence agencies (§ 2). Next, we provide a survey of literature that relates to the needs of EADS' use-case and to query relaxation (§ 3). We then propose a generic framework for building a relaxed RDF query engine (§ 4); we currently focus on entity-lookup queries using similarities of RDF terms. Subsequently, we discuss a generic technique for extracting distance/similarity scores between resources based on their structured descriptions (§ 5). We then outline an early prototype for relaxing queries—that represent witness observations—against a dataset of vehicle descriptions, testing different similarity measures (§ 6). We conclude that our own results are too preliminary for deployment, but argue that RDF query relaxation (or more generally "fuzzy querying") is an important, timely research topic not only for EADS' use-case, but may unlock various potential and diverse Semantic Web applications involving vague/uncertain user requirements.

2 Use-case overview

Our use-case arises from an on-going research project at the European Aeronautic Defence and Space Company (EADS): a large European aerospace, defence and military contractor. EADS Innovation Works is the corporate research and technology department of EADS that explores areas of mobility, security and environment. One of the team's key interests relates to civilian security, and enabling increased agency collaboration through use of intelligent systems. EADS has been working on the development of systems for intelligence analysis to aid post-crime police investigations [30], where analysts need to process raw information, determine valuable evidence, and identify the entities involved and their relationships. Investigations often rely on human observations, including police reports, or statements from victims, witnesses and informers.

Such data are the result of subjective assessment and often carry inherent vagueness and uncertainty. Human observations provide an *estimate* of the entity observed, described in natural language, and may be imprecise (i.e., stating that a suspect was 1.77 m tall when (s)he was 1.79 m tall) or vague (i.e., "*between 1.7 m – 1.85 m*", or "*average height*", etc.). Previous work [28,29] analysed issues with using human intelligence data (HUMINT), and presented methods to align data in different formats (numeric, textual and ranges). Herein, we view such observations as structured *fuzzy queries* to be executed over crisp data.

Posing complex but potentially vague/approximate queries against a crisp index broaches various issues. Numeric attributes—such as height—can be made "vague" using standard range queries. However, answers will not be scored: those near the "sweet-spot" of the range are not distinguished from those near the borders of *plausibility*. Given multiple attributes, the need for scoring becomes more pronounced. Furthermore, given non-numeric attributes (e.g., a `navy` getaway car, probably a `Toyota`), standard query answering offers no support for vagueness. Ideally, the query engine should return ranked approximate answers by relaxing both numeric and non-numeric values. Furthermore, the query should allow for specifying different levels of relaxation for different attributes; e.g., that `Toyota` is more vague and should be relaxed more than `navy` in the results.

We first considered applying ontologies and deductive methods to the use-case, looking to formalise the domain and to use fuzzy/annotated reasoning and querying techniques [20]. We abandoned this approach since (i) no legacy formal models were already in use; (ii) creating a speculative formal model from scratch (and fostering agreement) was deemed infeasible given the idiomatic nature of observations; (iii) inference would be too coarse-grained given the low resolution of formal knowledge available for the scenario. Instead, we decided to pursue an *inductive* approach, touching upon the area of *cooperative answering*.

3 Background and State-of-the-art

Cooperative answering involves the application of Grice's Maxims [13]—which describe how to be helpful in conversation—to information systems. Cooperative answering covers a broader area than our *current* scope, also trying to detect/circumvent user misconception, to provide additional justifications for the answer set, and to include entailed answers (as per deductive reasoning). We refer the reader to a survey of seminal works by Gaasterland et al. [10] and to a more recent survey of cooperative databases by Chu [5]. A pertinent technique in the area of cooperative answering is *query relaxation* or *generalisations*, whereby the criteria specified by the user are "relaxed" to include further, relevant content in the answers [9]. For example, Schumacher and Bergmann [27] propose using similarity measures to perform query relaxation in support of case-based reasoning over databases. Bruno et al. [3] investigate the use of query relaxation for efficient top-k retrieval over numeric, multi-attribute database relations.

Recently, some authors have looked at *deductive* query relaxation mechanisms for RDF [16,17,7,25]. Huang et al. [16], Hurtado et al. [17] and Poulovassilis & Wood [25] all propose using RDFS semantics to perform relaxation, through, e.g., generalising query patterns up class or property hierarchies, etc. Dolog et al. [7] use query rewriting rules to perform logical relaxation based on explicit user-preference models. For our use-case, logical relaxation *alone* is too coarse; e.g., members of the same class are viewed analogously for type relaxation, etc.

Other authors have proposed similarity-based, inductive query relaxation for RDF [19,8], but focus on lexical analyses. Kiefer et al. [19] propose integrating "customised similarity functions" into SPARQL, allowing for string similarity

measures such as Levenstein, Jaccard and TF–IDF to be invoked. More recently, Elbassuoni et al. [8] propose relaxation based primarily on measuring entity similarity as the Jensen–Shannon divergence of the language models of virtual prose documents constructed for each entity. Again, such lexical similarity techniques are too shallow for our use-case (consider `red` vs. `blue` vs. `navy`); they still do not leverage the rich, *structured* descriptions of entities during matching. (Herein, we may use the dual terms *distance* and *similarity* interchangeably.)

Where numerical values are consistently defined for entities (e.g., lat./long. for places, or *L*a*b** triplets for colours, etc.), distances can be computed based on Euclidean spaces where each attribute is a dimension (i.e., $\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$ for a_i, b_i comparable numerical values for entity A and B resp.). However, matching based on categorical attributes—which are prevalent in RDF data—is more difficult [2]. An *overlap measure* coarsely assigns a constant distance d (typically $d = 1$) between entities that do not match for a given categorical attribute, or zero distance otherwise. Otherwise, Boriah et al. [2] survey a number of finer-grained methods for categorical matching; e.g., the Goodall measure [12] assigns a higher similarity between entities sharing a more selective category (i.e., a rare string value). Such measures are *data-driven*, relying on statistics from the data to compute distances: consequently, the similarity of two entities can be influenced by other peers (unlike, e.g., absolute Euclidean distances).

RDF similarity measures are also studied for *instance matching*. Although instance matching focuses on finding `owl:sameAs` alignments—and although many such frameworks rely on deductive methods (e.g., LN2R [26]) or lexical similarity methods (e.g., KnoFuss [22], RDFSim [18]) to generate alignments—a few inductive similarity metrics have been proposed based on overlaps in the descriptions of entities [23,15,14]; one such approach is later used in § 5.

4 Relaxation framework

We now propose a conceptual framework for relaxation of an *entity query*: a list of attribute–value or attribute–variable pairs $Q := \langle (p_1, o_1), \ldots, (p_n, o_n) \rangle$ such that each p_i is a property URI and each o_i is either a variable, a URI or a literal (i.e., $Q \subset \mathbf{U} \times \mathbf{VUL}$).[3] A *crisp* response consists of entities (subjects) with predicate–object edges directly matching the query, as well as bindings for any variables in $o_1, \ldots, o_n$. In SPARQL terms, this query model roughly corresponds to basic graph patterns with a common subject variable; for example:

```
SELECT * WHERE {?s :colour :blue ; :city :NY ; :type :Sport ; year 2010 ; reg ?r .}
```

To relax queries, we define a matcher as a function $M : \mathbf{VUL} \times \mathbf{UL} \to \mathbb{R}_{[0,1]}$ that maps a pair of values into a *relaxation score*: a value in $[0, 1]$ where 1 indicates that the two values are not interchangeable, and 0 indicates perfect interchangeability (e.g., $M(\mathsf{c}, \mathsf{c}) := 0$, $M(?\mathsf{v}, \mathsf{c}) := 0$). Each matcher is a *distance function* between the query and entity values, respectively. The match function

[3] The query is given an ordering for later convenience. We re-use standard RDF notation: **V** denotes variables, **U** URIs and **L** RDF literals. AB denotes $A \cup B$.

may not be symmetric for pairs of values at different levels of specificity: e.g., $M(\texttt{:blue}, \texttt{:navy})$ might be 0.2 suggesting :navy as a good relaxation for generic :blue, whereas $M(\texttt{:navy}, \texttt{:blue})$ might be 0.5 since :navy is more specific.

Matchers then form the core of the relaxation framework, and can be instantiated in different ways (cf. [24]). For numeric attribute matchers (e.g. :year), normalised distances can be used: letting max_i and min_i denote the max./min. values for a numeric property p_i appearing in the data, qv_i a value in the query and ev_i a value for an entity, we can apply a normalised numeric matcher $M_i : (qo_i, eo_i) \mapsto \left|\frac{qo_i - eo_i}{max_i - min_i}\right|$. For string attributes with functional character strings (e.g., registration plates), lexical matchers can be used; we later use a Levenshtein edit-distance matcher for licence plates such that $M_i : (qo_i, eo_i) \mapsto \frac{Lev(qv_i, ev_i)}{\max(|qo_i|, |eo_i|)}$; other matchers can be used as appropriate. For *categorical* attributes—with URIs or a discrete set of literals as values (e.g., colour, city)—creating a matcher often requires background knowledge about the different values; as per Schumacher and Bergmann [27], we thus propose to use a similarity table for such attributes, computed by a background matching process (discussed later in § 5).

Thus, the relaxation framework may involve multiple matchers: a toolbox of appropriate matchers can be offered to an administrator. Where a suitable matcher is not found for a pair of values, the query engine can resort to returning standard "crisp" answers, allowing for an ad-hoc, incremental relaxation framework. We currently do not consider inference or relaxation of properties etc.; our framework could perhaps be extended as per the literature surveyed in § 3.

For a query $Q = \langle (p_1, o_1) \dots (p_n, o_n) \rangle$ and entity E, the matchers generate a tuple of numeric distances $M_{i \dots n}(Q, E) = \langle d_1, \dots, d_n \rangle$. Considering the query as the origin, the matchers map entities onto points in an n-dimensional Euclidean space with each dimension ranging over $[0, 1]$ (a unit n-cube). Where an entity has multiple values for a given attribute, the closest to the query is used; where an entity is not assigned a query-attribute, the matcher returns 1.[4] Thereafter, entities on the origin are crisp matches. Otherwise, the distance from an entity to the query-origin can be measured straightforwardly as a Euclidean distance (in this case, $\sqrt{\sum_{i=1}^{n} d_i^2}$), or with *root-mean squared deviation* (*RMSD*: $\sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}}$).

The overall distance from the query-origin to each entity gives an *overall relaxation score* that can be used to order presentation of results, or to perform top-k thresholding. Furthermore, users can annotate query attribute–value pairs with a *vagueness* score that allows for controlling the relaxation of individual facets (e.g., to allow more relaxation for :colour than :city). Thus a *vague query* is defined as $Q' := \langle (p_1, o_1, v_1,) \dots, (p_n, o_n, v_n) \rangle$ where $v_1, \dots, v_n \in \mathbb{R}_{[0,1]}$ (i.e., $Q' \subset \mathbf{U} \times \mathbf{VUL} \times \mathbb{R}_{[0,1]}$). A value v_i indicates a threshold for d_i such that the entity will only be considered a result if $d_i \leq v_i$ (e.g., if $v_1 := 0$, then (p_i, o_i) must have a crisp match). Thus, the origin and the coordinate $\langle v_1, \dots, v_n \rangle$ prescribe a region of space (an m-orthotope for m the number of non-crisp query attributes) within which results fall into, allowing to tweak relaxation results.

[4] Intuitively, this is the relaxed form of standard conjunctive query answering.

5 Generating similarity tables for categorical values

The query relaxation framework relies on matchers to define distances between attribute values. We discussed direct matchers for numerical values (based on normalised Euclidean distance/RMSD) and for string values (based on normalised edit distances), but left the generation of similarity tables for categorical values. Such tables can be generated in a number of ways. First, if the set of categorical values is small enough, tables can be generated manually (on a pay-as-you-go basis); however, the number of scores needed is quadratic for the number of values. Otherwise, tables can be (semi-)automatically generated by any form of similarity measure; e.g., certain categorical attributes (like colours or places) are described using numeric attributes, allowing use of Euclidean distances.

Background information about the values can also be used to compute similarity scores. For instance, given a sufficient *unstructured* corpus, distributional semantic relatedness [11] can generate similarities between terms based on their co-occurrences in prose text. Given a sufficient *structured* RDF corpus describing the terms, instance-matching techniques can be used to generate similarity measures. Herein, we investigate the latter approach, using a generic RDF similarity technique which we had previously proposed [14], viz. *concurrence*, which is designed to cope with diverse RDF corpora; the core premises of concurrence have also been extended for the purposes of instance matching by other authors [15]. We now introduce the concurrence algorithm, which we use later (§ 6) to mine make–model similarity scores from an RDF corpus extracted from DBpedia.

Concurrence matches RDF resources based on their shared property-value pairs (generalising in-links and out-links: i.e., considering both *sp* and *po* pairs). Values are considered purely categorical, such that only crisp matches on shared pairs are used. Similarity is influenced more by pairs that are determined to be highly selective (i.e., to be exclusive): if a group of resources share a categorical value, the similarity score between these resources is boosted proportionate to the size of the group. This is similar in principle to Goodall's measures [12,2].

Since RDF assumes an Open World (i.e., incomplete data), the concurrence method is "monotonic": similarity is not punished when two resources have different values for an attribute since attributes can have multiple values, and (without OWL machinery checking cardinalities and ground (in)equalities), it is difficult to ascertain whether or not two RDF resources necessarily *disagree* on an attribute value (as opposed to just the *known* values differing). Instead, each additional shared pair monotonically increases the similarity score.

Formally, for an RDF dataset, let $\mathsf{card}(p, v)$ denote the number of resources that share a given property–value pair (abstracting subject/object directionality), indicating how exclusive that pair is in the data. Next, let $\overline{\mathsf{card}}(p)$ denote the mean cardinality of p across all such values v in the data, indicating how exclusive the property is (in general). The base similarity score given to all resources sharing the pair (p, v) is then defined as $\mathsf{concur}(p, v) := \frac{1}{\overline{\mathsf{card}} \times \mathsf{card}(p,v)}$, which returns a value in $[0, 0.5)$ if the pair (p, v) is shared at least once.

Now, two resources sharing multiple pairs are given a multiset of concur scores $C := \{c_1, \ldots, c_n\}$. The concurrence method combines these using a *probabilistic*

sum, such that $\perp_{\text{sum}}(c_a, c_b) := c_a + c_b - c_a \times c_b$. Since this function is associative and commutative, it is well-defined for the multiset C (i.e., by summing pairs in whichever order).[5] The function is also monotonic and outputs a value in $[0, 1]$ (assuming $\forall c_i \in C(0 \leq c_i \leq 1)$). The intuition behind this aggregation function is that each additional match score reduces the current distance between the two resources by a product of itself; for example $\perp_{\text{sum}}(0.5, 0.5) = 0.75$, or $\perp_{\text{sum}}(0.8, 0.2) = 0.8 + (1 - 0.8) \times 0.2 = 0.84$. As per the examples, the probabilistic sum gives stronger weight to individual high values than an arithmetic sum.

The baseline concurrence method is also adjusted to account for some cases of dependence between property-value pairs. To reduce the effect of groups of (e.g., sub-)properties being given the same value—i.e., pairs $\{(p_1, v), \ldots, (p_n, v)\}$—only the most exclusive such pair (for which $\mathsf{concur}(p_i, v)$ gives the highest score) will be used, and the rest discarded. To reduce the effect of repeated shared values for a given property—i.e., pairs $\{(p, v_1), \ldots, (p, v_n)\}$—all shared pairs for each property p are (probabilistically) summed up separately to a maximum value of $\overline{\mathsf{card}}(p)$, with these scores then summed to an overall total.

Finally, generating quadratic pair-wise similarity scores may not be feasible for large datasets. Thus, a threshold t (s.t. $t > 2$) is specified for large-scale scenarios where pairs $\mathsf{card}(p, v) > t$ are ignored: the number of matches generated are quadratic wrt. $\mathsf{card}(p, v)$ but the respective scores are inverse-proportional. For example, if $\mathsf{card}(\texttt{type}, \texttt{vehicle}) = 50,000$, this shared pair would require generating $\frac{50{,}000^2 - 50{,}000}{2}$ atomic matches with a score of $< \frac{1}{50{,}000}$. The threshold thus allows for pruning potentially massive volumes of low-scoring matches.

Concurrence is implemented using batch processing techniques (sorts/scans), and have been demonstrated on a corpus of one billion quadruples of openly-crawled Linked Data (with $t := 38$). Further details are available in [14].

6 Proof of concept

In this section, we investigate proof-of-concept for the original use-case.

6.1 Vehicles scenario and data

Relating to the crime observation use-case, we focus on vehicle descriptions: automobile observations are often integral to police investigations, and it would be easier to find rich, publically available, representative, structured datasets for the vehicles use-case than, e.g., for weapons or criminals.

The case study was then to match structured, partial, possibly vague and imprecise queries against a crisp set of car instances. This dataset would emulate information provided by the UK Driver and Vehicle Licensing Agency (DVLA), or a Home Office database of recently stolen cars, etc. An observation by a witness would be something like "*a blue getaway car that looked like a Toyota with*

[5] The probabilistic sum (aka. algebraic sum) is the dual t-conorm of the product t-norm; it also refers to the probability of a disjunction of independent events.

an 'LD' licence plate". The relaxation framework is used to derive a ranked list of cars from the dataset in order of their relevance to the observation. In this respect, the observation acts like a query which should be executed against the car instances. Results should include not only those cars which directly match the characteristics given in the observation, but also similar cars. Different characteristics of the observation can be annotated with different vagueness values.

For demonstration purposes, we decided that the chosen dataset should contain information about a significant number of car instances, with attributes for (at least) make, model, colour and body-type, covering common facets of vehicle observations. We thus took an existing structured dataset describing 50,000 car instances based on a popular Irish website advertising used cars. Each car instance is described using the following six properties: vehicle make (48 unique values; e.g., `Toyota`), make–model (491 values; e.g., `Toyota Corolla`), body style (8 values; e.g., `Saloon`), fuel type (5 values; e.g., `Diesel`), colour (13 values after normalisation; e.g., `navy`), and registration (i.e., unique licence plate; 50,000 values). Taking the raw data, colours were normalised into a set of thirteen defined values, a new set of UK-style licence plates was randomly generated, and the data were modelled in RDF using Hepp's Vehicle Sales Ontology (VSO).[6]

Notably, all vehicle attributes except licence-plates are categorical, and thus require tables that encode similarity/distance scores. To relax licence-plate values, we allow wildcard characters in the query and use the normalised Levenshtein measure mentioned earlier. For colour, the thirteen values were mapped to an *L*a*b** three-dimensional colour space, where *Delta-E* was used to compute (Euclidean) distances between the colours and generate a matrix for relaxation.[7] An open, non-trivial challenge was then posed by the other properties. For fuel-type, it was particularly unclear what kind of relaxation behaviour should be expected for the use-case; a set of regular distance scores were manually defined. Of more interest were the make–model, model and body-style attributes, for which further background information was needed.

6.2 Mining DBpedia for background information

To acquire a background, structured corpus on the make and make–model values appearing in the data, we enriched the baseline RDF data with selected DBpedia exports [1] from Wikipedia: we noted that DBpedia exported a rich set of data about many of the vehicle make–models, including width, height, categories, engine, transmission, etc. The resulting RDF data would then serve as input for similarity techniques to generate scores for subsequent query relaxation.

The first step was to map string values in the data (e.g., `Toyota Corolla`) to DBpedia URIs (e.g., `http://dbpedia.org/resource/Ford_Mondeo`). First attempts were made using the reconciliation function of the "RDF Extension for Google Refine"[8], which allows for mapping strings to resources in a SPARQL

[6] Available at `http://www.heppnetz.de/ontologies/vso/ns`; retr. 2011/12/12

[7] Distances in the L*a*b* colour space correspond more closely with discrepancies in human perception than RGB/CMYK models [32].

[8] See `http://lab.linkeddata.deri.ie/2010/grefine-rdf-extension/`; retr. 2011/12/12/

№	Top Overall Matches		№	Top Matches Across Makes	
1	Honda_Accord	Honda_Odyssey	8	Audi_80	Volkswagen Passat
2	Mercedes-Benz_S-Class	Mercedes-Benz_SL-Class	11	Audi A3	Škoda_Octavia
3	Suzuki_Alto	Suzuki_Wagon_R	13	Audi_TT	SEAT_León
4	Ford_Tourneo_Connect	Ford_Transit_Connect	25	Citroën_C1	Peugeot_107
5	Mitsubishi_Colt	Mitsubishi_Mirage	27	Hyundai_i10	Kia_Picanto
6	Volvo_S60	Volvo_S80	28	Audi_A3	Seat_Léon
7	Nissan_Maxima	Nissan_Murano	30	Daewoo_Winstorm	Opel_Antara
8	Audi_80	Volkswagen Passat	38	Hyundai_Tuscan	Kia_Sportage
9	Merdeces-Benz_S-Class	Merdeces-Benz_W220	39	Citroën_C3	Hyundai_Getz
10	SEAT_Córdoba	SEAT_Ibiza	40	SEAT_Toledo	Volkswagen_Jetta

Table 1: Top ten make–model matches overall (left) and for distinct makes (right)

endpoint and refining these mappings in a second phase [21]. Although numerous matches were found, many make–models were not reconciled to DBpedia URIs.

Instead, we adopted a manual approach by appending make–model strings onto the DBpedia namespace URI, replacing space with underscore. However, this approach also encountered problems. First, of the 491 string values, 68 models (14%) did not have a corresponding reference in DBpedia (`404 Not Found`): some of the unmatched models were colloquial UK/Irish names (e.g., the make–model `Citroen Distpatch` is known elsewhere as `Citroën Jumpy`), some were misspelt, and some had encoding issues. These values were manually mapped based on suggestions from Wikipedia search. Second, some of the matches that were found returned little data. Of these, some were redirect stub resources (e.g., `Citroen C3` redirects to `Citroën C3`), where we "forwarded" the mapping through the redirect. Others still were disambiguation pages (e.g., `Ford Focus` disambiguates to `Ford Focus International`, `Ford Focus America` and `Ford Focus BEV`). Furthermore, some resources redirected to disambiguation pages (e.g., `Ford Focus ST` redirect to the `Ford Focus` disambiguation page). Here, we mapped strings to multiple resources, where `Ford Focus` was mapped to the set of DBpedia resources for { `Ford Focus`, `Ford Focus America`, `Ford Focus International` and `Ford Focus BEV` }. In total, 90 strings (18.3%) had to be manually mapped to DBpedia. Given the mappings, we retrieved 53k triples of RDF data from DBpedia, following redirects and disambiguation links.

We applied concurrence over the dataset with a threshold $t := 200$ (each shared pair with $\mathsf{card}(p, v) = 200$ would generate >20,000 raw concur scores at least below 0.005). For the 491 original make–model values, concurrence found non-zero similarity scores for 184k (76%) of the 241k total car-model pairs possible, with an average absolute match score of 0.08 across all models.

The top ten overall results are presented in Table 1, where we also present the top ten matches for models with different makes; note that matches are symmetric and that all matches presented in the table had a similarity score ex-

ceeding 0.99 indicating highly-relaxable values. Similarity scores are convertible to distance scores for relaxation using $dist = 1 - sim$.[9]

The results were of varying quality for our use-case. For the top make–model match—`Honda Accord`/`Honda Odyssey`—the former is a saloon and the latter is a minivan, and as such, the two models are physically diverse; however both models share specific/exclusive components (e.g., the same engine) which causes the high concurrence score. Another such example is given by `Nissan Maxima`/`Nissan Murano`; also the `Seat Cordoba` is the saloon version of the `SEAT Ibiza` hatchback. This raises a more general issue with the subjective nature of relaxation: although these pairs may be dissimilar for a witness-observation use-case, they are similar from a sales or manufacturing perspective. Still, we see some potentially interesting relaxation for the witness use-case. Many of the highest-scoring cars are physically similar; besides those of the same make, many of the matches with different makes are (or have been) based on the same platform: i.e., are fundamentally the same car with different skins.

Besides manual inspection of top-ranked results, in the absence of a gold standard, evaluating the large set of concurrence results from DBpedia is difficult. Although inspection yielded promising examples of relaxation, conversely, unintuitive/undesirable results were also discovered. For example, the `Westfield SE` (a rare model of "kit" sports-car) had very little information in DBpedia, and hence had no notable concurrence matches; similar problems were encountered for other rare models.

A major disadvantage of concurrence is that distances are not geometric in nature: considering resources A, B, C as points, then the concurrence "distance-vectors" $\overrightarrow{AB}$, $\overrightarrow{BC}$ and $\overrightarrow{AC}$ cannot be considered as true spatial vectors since the additive property thereof does not hold: $\overrightarrow{AB} + \overrightarrow{BC} \neq \overrightarrow{AC}$. Thus, for example, if concurrence gives a score of $\sim$1 indicating near-perfect similarity between B and C (distance of $\sim$0), this does not imply that B and C have similar scores to other cars ($|\overrightarrow{BC}| \approx 0 \not\Rightarrow |\overrightarrow{BA}| \approx |\overrightarrow{CA}|$). Taking an example from our evaluation, the `Audi RS4` and `Audi A4` were matched closely with 0.97; the `Audi RS4` and `Audi A8` models were matched with a score of 0.8; but the `Audi A4` and `Audi A8` models were matched with a low score of 0.29 despite being very closely matched to a common model: the first two pairs gained their concurrence scores through a largely distinct of (incomplete) attributes. This results in unintuitive, incomparable distance scores when looking beyond just pairs of resources.

Furthermore, the results of the concurrence method is heavily dependent on the underlying data. DBpedia does not distinguish between different editions of make–models down through the years, where for example, the RDF data for six diverse editions of `Ford Fiesta`—spanning forty years—are presented together under one resource since they are described in one Wikipedia article (albeit with separate info-boxes). Due to these, and other issues relating to data quality, we decided to pursue tests over a smaller, cleaner dataset.

[9] Alternatively, the similarities could be normalised into a non-parametric, rank-based distance, where a relaxation value of 0.5 includes the top-half similar models.

6.3 Comparing concurrence vs. numerical matching

For evaluating the concurrence method, we wanted to compare its results against standard Euclidean distances over numerical attributes. We attempted to extract numerical values from the DBpedia make–model data, but the presence of multiple values per attribute and the incompleteness of the data precluded any meaningful numerical analysis. Hence, a manual evaluation corpus was gathered directly from Wikipedia for which concurrence and standard numerical approaches could be compared. The corpus consisted of tabular data describing 200 make–model–edition values in terms of numerical values and ranges (years produced, doors, engine capacity, wheelbase, length, width, height and curb weight) as well as categorical values (make, model, edition, body-style); the dataset was, however, incomplete as not all values could be found for all editions surveyed.

Since this evaluation dataset focuses primarily on numeric values, we modified the discrete concurrence algorithm to consider continuous values. Given a set of property-value pairs $\{(p, v_1), \ldots, (p, v_n)\}$ with each $v_i \in \mathbb{R}$ a value for a specific car-edition, and given two resources (editions) with pairs (p, v_a) and (p, v_b) respectively (where $v_a \leq v_b$), we define the numeric cardinality between the two values as $\mathsf{ncard}(p, v_a, v_b) := |\{v_i : \exists (p, v_i) \text{ and } v_a \leq v_i \leq v_b\}|$, denoting the number of resources that fall in the inclusive range of those two values for that property $[v_a, v_b]$, assuming single-valued attributes for brevity. The concurrence score for a crisp categorical value becomes $\mathsf{concur}(p, v) := \frac{1}{\mathsf{card}(p,v)}$ and for a numeric value becomes $\mathsf{concur}(p, v_a, v_b) := \frac{1}{\mathsf{ncard}(p,v_a,v_b)}$. Furthermore, we turned off concurrence's thresholding and dependence filters, which were not required for the clean, small dataset at hand (reverting to a simple probabilistic sum).

We then measured the correlation between results for the modified concurrence algorithm, and for non-normalised RMSD (distances not divided by the $max - min$ denominator) and normalised RMSD (distances for each attribute pre-normalised into $[0, 1]$) computed over numerical attributes. To quantify the correlation, we used *Kendall's* τ, which measures correlation between two orderings: the τ value is in the range $[-1, 1]$ where -1 indicates a perfect negative correlation (the orderings are "reversed"), 0 indicates independence of orderings, and 1 indicates a perfect correlation (the same ordering). The correlation between the concurrence and non-normalised RMSD was positive but weak ($\sim$0.1): without the pre-normalisation of values, attributes with larger units (such as years in the thousands) tended to have high influence. However, between concurrence and normalised RMSD values, the correlation was higher at $\sim$0.54: the main difference was attributable to the monotonic nature of concurrence, which did not punish mismatches between single values to the same extent as the normalised RMSD measure. RMSD had the more favourable results in this regard due in part to the clean nature of the data. Conversely, concurrence gave better matches for incomplete data, where RMSD gave very high scores.

Finally, we compiled the 40,000 make–model–edition similarity scores for each approach into similarity scores for make–models, makes and body-styles by taking the average across all pairs in the generalised groups. For example, to generate a similarity score between `saloon` and `hatchback` body styles (say the

sets S and H resp.), we took the average of all edition-similarities between both groups (i.e., the arithmetic mean of scores for all edition-pairs in $S \times H$).

6.4 Experimenting with query relaxation

With various matcher mechanisms and tables in hand, we turned to testing query relaxation against the 50k vehicles dataset. We developed a simple prototype to take a vague query, perform a full scan of the dataset computing relaxation scores for each entity based on the matchers, and return a ranked list of answers. Although the query algorithm is linear, we acknowledge that sub-linear (and sub-second) query times might be required for deployment. There are various avenues to enable sub-linear performance (for entity search): (i) if crisp facets are given, these can be looked up directly where relaxation is then used to filter initial results (similar in principle to SPARQL `FILTER`s); (ii) given a matcher based on a table of similarities or on numeric distance, the corresponding vagueness score for the query facet can be used to compute a range query that can be executed as a table lookup (assuming an index with the correct sort order is available); (iii) special relaxation indices can be built, for example, using Locality Sensitive Hashing to index Euclidean points and enable efficient neighbourhood searches [4]. Different optimisations are feasible for different types of matchers. We leave further investigations of optimisations for related work: our current aim is to validate the proposed techniques and offer proof-of-concept.

Table 2 presents the top-5 results for three example witness observations, which were modelled as structured vague queries and run against the vehicles dataset. Vagueness scores are manually chosen for proof-of-concept: mapping textual vagueness to numeric vagueness is out of scope. For the matchers, we used a normalised Levenshtein edit-distance for licence plates; the L*a*b*-based similarity table for colour; and for make, model, edition and body-style, we used three configurations with similarities computed from the 200 vehicle-editions dataset (for which we could compute three sets of results) using (i) concurrence, (ii) normalised Euclidean and (iii) absolute Euclidean distance measures. The scores are based on RMSD from the query origin (subtracted from one).

Observation A gives an example of relaxation for colours; however, note that the original query also requests relaxation for models, but where colour distances are typically much shorter. No difference occurs between the different matcher configurations for car-make (the first relaxed car-make appears in position 10–11). This is a weakness of the framework: different matchers may produce incomparable distances, creating an "imbalance" in the relaxation across different attributes; a possible solution is the use of rank-based distances. From Observation B, we see `maroon` being returned as a crisp match for `red` and see example relaxation of models; from the scoring, we also note that different matchers may vary in terms of inclusiveness. Finally, Observation C uses the Levenshtein edit-distance matcher for licence-plates in combination with colour relaxation, where the `black` result is questionable.

In the absence of a gold standard, we could not evaluate precisely the effectiveness of the relaxation framework for generating relevant, approximate,

Observation A:	"*A greenish car, maybe a Peugeot*".						
Relaxed query:	{(colour, green, 0.2), (make, Peugeot, 0.8)}						
№	**All Approaches (same results)**						
	result						*score*
1	Peugeot			green			1.00
2	Peugeot			yellow			0.96
3	Peugeot			brown			0.95
4	Peugeot			teal			0.95
5	Peugeot			aqua			0.93
Observation B:	"*A red SUV, looked like a Land Rover Freelander*".						
Relaxed query:	{(colour, red, 0), (body, SUV, 0), (model, LR.-Freelander, 0.8)}						
№	**Concurrence**			**Norm. Euclidean**			**Abs. Euclidean**		
	result		*score*	*result*		*score*	*result*		*score*
1	LR. Freelander	red	1.00	LR. Freelander	red	1.00	LR. Freelander	red	1.00
2	LR. Freelander	maroon	1.00	LR. Freelander	maroon	1.00	LR. Freelander	maroon	1.00
3	Hyundai Trajet	red	0.86	Hyundai Tuscon	red	0.93	Hyundai Tuscon	red	0.84
4	Kia Sorento	red	0.86	Hyundai Tuscon	maroon	0.93	Hyundai Tuscon	maroon	0.84
5	Kia Sorento	maroon	0.86	Kia Sorento	red	0.92	Renault Scenic	red	0.84
Observation C:	"*A light Audi A3 8L, 2006 UK reg. starts with SW and ends with M* ".						
Relaxed query:	{(colour, white, 0.4), (edition, Audi-A-8l, 0.1), (reg, SW?6??M, 0.4)}						
№	**All Approaches (same results)**						
	result						*score*
1	Audi A3 8L			SW06RWM			yellow		0.92
2	Audi A3 8L			SF56GCN			white		0.91
3	Audi A3 8L			BW06LJN			gray		0.90
4	Audi A3 8L			SW04TVH			black		0.85
5	Audi A3 8L			AE56MWM			maroon		0.83

Table 2: Example observations, relaxed queries and results

well-graded answers. Generating high-quality distance scores based on categorical values is much more challenging than for numeric attributes [2], but a crucial part of inductive query relaxation for RDF. Table 2 provides some preliminary results towards query relaxation for RDF, but based on the outlined problems, the results were deemed currently unsuitable for deployment in the use-case. However, we believe that with further investigation, such methods can be improved and adapted for use in other less critical applications, such as relaxing query results from public SPARQL endpoints.

7 Conclusion

In this paper, we introduced a use-case from EADS involving matching witness observations against structured entity descriptions. We proposed query-relaxation as a framework within which to tackle the problem. We discussed how matchers can be used to enable query relaxation, how different matchers can be combined and used for different attributes, and how RDF similarity techniques can be used to compile similarity scores for categorical values. We presented the results of various proof-of-concept experiments with the goal of performing query-relaxation over 50k car descriptions. We discussed using DBpedia to

mine background information for make–model similarity scores, computed using our proposed *concurrence* method. We subsequently compared the correlation between the results of a modified version of our concurrence method and that of standard Euclidean distance measures. Finally, we presented some example query-relaxation results based on vague observations of vehicles as per the use-case. Unfortunately, the results were not deemed reliable enough for deployment.

As such, lots more work is left to do and many challenges are left unaddressed. Beyond our use-case, we argue that cooperative answering and query relaxation is an important, timely topic for Semantic Web researchers to pursue: RDF stores often index diverse datasets with complex schemata, against which writing precise queries is extremely challenging. Query relaxation would then find application in various areas, including Web search and recommender systems [3], e-commerce [6], case-based reasoning [27], reconciliation [21], etc. As discussed, current instance matching techniques can be repurposed for relaxation.

In summary, we would hope to see further proposals towards "cooperative SPARQL engines" which intelligently aid users—using a mixture of deductive and inductive techniques—in the daunting task of answering potentially vague queries over diverse RDF datasets. We have taken tentative steps in this direction, looking at query relaxation for entity queries. Further focus on inductive techniques—like those proposed by Stuckenschmidt [31] or Hu et al. [15]—will also better position the Semantic Web community to support applications needing "intelligence" beyond *just* crisp semantics and logics.

Acknowledgements: *This paper was funded in part by by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). We thank Nuno Lopes, Axel Polleres, Ali Hasnain and Maciej Dabrowski for their contributions.*

References

1. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the Web of Data. *J. Web Sem.*, 7(3):154–165, 2009.
2. S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In *SDM*, pages 243–254, 2008.
3. N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM Trans. DB Syst.*, 27(2):153–187, 2002.
4. S. Chaudhuri, M. Datar, and V. R. Narasayya. Index selection for databases: A hardness study and a principled heuristic solution. *IEEE Trans. Knowl. Data Eng.*, 16(11):1313–1323, 2004.
5. W. W. Chu. Cooperative database systems. In *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc., 2008.
6. M. Dabrowski and T. Acton. Modelling preference relaxation in e-commerce. In *FUZZ-IEEE*, pages 1–8, 2010.
7. P. Dolog, H. Stuckenschmidt, H. Wache, and J. Diederich. Relaxing RDF queries based on user and domain preferences. *J. Intell. Inf. Syst.*, 33(3):239–260, 2009.
8. S. Elbassuoni, M. Ramanath, and G. Weikum. Query relaxation for entity-relationship search. In *ESWC (2)*, pages 62–76, 2011.

9. T. Gaasterland. Cooperative answering through controlled query relaxation. *IEEE Expert*, 12(5):48–59, 1997.
10. T. Gaasterland, P. Godfrey, and J. Minker. An overview of cooperative answering. *J. Intell. Inf. Syst.*, 1(2):123–157, 1992.
11. E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007.
12. D. W. Goodall. A new similarity index based on probability. *Biometrics*, 22(4), 1966.
13. P. Grice. Logic and conversation. *Syntax and semantics*, 3, 1975.
14. A. Hogan, A. Zimmermann, J. Umbrich, A. Polleres, and S. Decker. Scalable and distributed methods for entity matching, consolidation and disambiguation over Linked Data corpora. *J. Web Sem.*, 10:76–110, 2012.
15. W. Hu, J. Chen, and Y. Qu. A self-training approach for resolving object coreference on the semantic web. In *WWW*, pages 87–96, 2011.
16. H. Huang, C. Liu, and X. Zhou. Computing relaxed answers on RDF databases. In *WISE*, pages 163–175, 2008.
17. C. A. Hurtado, A. Poulovassilis, and P. T. Wood. Query relaxation in RDF. *J. Data Semantics*, 10:31–61, 2008.
18. E. Ioannou, O. Papapetrou, D. Skoutas, and W. Nejdl. Efficient semantic-aware detection of near duplicate resources. In *ESWC (2)*, pages 136–150, 2010.
19. C. Kiefer, A. Bernstein, and M. Stocker. The fundamentals of isparql: A virtual triple approach for similarity-based semantic web tasks. In *ISWC/ASWC*, pages 295–309, 2007.
20. N. Lopes, A. Polleres, U. Straccia, and A. Zimmermann. AnQL: SPARQLing up annotated RDFS. In *ISWC*, pages 518–533, 2010.
21. F. Maali, R. Cyganiak, and V. Peristeras. Re-using cool URIs: Entity reconciliation against LOD hubs. In *LDOW*, 2011.
22. A. Nikolov, V. S. Uren, E. Motta, and A. N. D. Roeck. Integration of semantically annotated data by the KnoFuss architecture. In *EKAW*, pages 265–274, 2008.
23. J. Noessner, M. Niepert, C. Meilicke, and H. Stuckenschmidt. Leveraging terminological structure for object reconciliation. In *ESWC (2)*, pages 334–348, 2010.
24. R. Oldakowski and C. Bizer. SemMF: A framework for calculating semantic similarity of objects represented as RDF graphs. In *ISWC (Poster Proc.)*, 2005.
25. A. Poulovassilis and P. T. Wood. Combining approximation and relaxation in semantic web path queries. In *ISWC (1)*, pages 631–646, 2010.
26. F. Saïs, N. Pernelle, and M.-C. Rousset. Combining a logical and a numerical method for data reconciliation. *J. Data Semantics*, 12:66–94, 2009.
27. J. Schumacher and R. Bergmann. An efficient approach to similarity-based retrieval on top of relational databases. In *EWCBR*, pages 273–284, 2000.
28. D. Stampouli, M. Brown, and G. Powell. Fusion of soft information using TBM. In *13th Int. Conf. on Information Fusion*, 2010.
29. D. Stampouli, M. Roberts, and G. Powell. Who dunnit? An appraisal of two people matching techniques. In *14th Int. Conf. on Information Fusion*, 2011.
30. D. Stampouli, D. Vincen, and G. Powell. Situation assessment for a centralised intelligence fusion framework for emergency services. In *12th Int. Conf. on Information Fusion*, 2009.
31. H. Stuckenschmidt. A semantic similarity measure for ontology-based information. In *FQAS*, pages 406–417, 2009.
32. M. Tkalčič and J. F. Tasič. Colour spaces: perceptual, historical and applicational background. In *IEEE EUROCON*, pages 304–308, 2003.