Finding and Explaining Similarities in Linked Data

Catherine Olsson
Raytheon BBN Technologies and
Massachusetts Institute of Technology
catherio@mit.edu

Plamen Petrov, Jeff Sherman, Andrew Perez-Lopez
Raytheon BBN Technologies
Arlington, VA
{ppetrov,jsherman,aperezlo}@bbn.com

***Abstract*—Today's computer users and system designers face increasingly vast amounts of data, yet lack good tools to find pertinent information within those datasets. Linked data technologies add invaluable structure to data, but challenges remain in helping users understand and exploit that structure. One important question users might ask about their data is "What entities are similar to this one, and why?" or "How similar are these two entities to one another, and why?". Our work focuses on using the semantic content of linked data not only to facilitate the process of finding similar entities, but also to produce automatically-generated and human-understandable explanations of what makes those entities similar. In this paper, we formulate a definition of an "explanation" of similarity, we describe a system that can produce such explanations efficiently, and we present a methodology to allow the user to tailor how "obvious" or "obscure" the provided explanations are.**

I. Introduction

Today's world is a world of data. As technology advances, it becomes easier and easier to collect and store vast amounts of data. Much of this data can be viewed in terms of nodes with properties and relationships, or edges, among those nodes — that is to say, it can be represented as a graph. Once a dataset has been represented in a graph format, such as with Semantic Web [1] or other linked data technologies [2][3], it can easily be combined with data from different sources. In this way, linked data allows already-vast datasets to be readily combined and connected, giving users and programs access to more data than ever before. The challenge, then, is in making sense of this data.

Some of the data analysis questions that are emerging include the following: How does one entity in the linked data relate to another entity, possibly derived from a different source? How does a given entity relate to the rest of the data? What are the similarities between two entities, and why? There are also related data search and retrieval questions to be tackled, such as "Find all entities similar to this entity" and "Find groups of entities that are similar to each other."

To solve these problems, work has been done at Raytheon BBN Technologies (BBN) to devise a similarity measure called the Structural Semantic Distance Measure (SSDM), which leverages both the structural and semantic content of linked data to find similar entities. SSDM is based on SimRank [4], a highly domain-general similarity measure with an efficient approximate calculation. SSDM improves on SimRank by incorporating the semantic content of edge labels, and by achieving greater independence of ontological choices.

Raw numerical similarity scores provide very little insight to users about what those scores mean, so users often want an *explanation* of how a score should be understood and interpreted. In this paper, we formulate a definition of an "explanation" of an SSDM score that ensures that the explanation is both human-understandable and well-grounded in how SSDM scores are calculated. We also describe a system that can produce such explanations efficiently.

Additionally, not all users will desire the same level of detail in their explanations. Therefore, we present a methodology for allowing the user to tailor how "obvious" or "obscure" the provided explanations are. We expect that users who are investigating an unfamiliar domain will prefer "obvious" explanations that refer to common and well-known properties, while expert users will prefer "obscure" explanations that shed light on less well-known relationships and details.

A. Motivation

Our work is motivated by a number of problems in the intelligence and military research communities. Many of those problems are ubiquitous and have direct translation to business intelligence, logistics, and planning. Take, for example, a model of a large organization C with its associates, their interactions, locations they visit, resources they use or produce, and events in which they participate. Given this information, one could explore the stated relationships among the constituents of C, such as "show all transactions that involve person X." Beyond these simple information-retrieval tasks, analysts might want to examine more complex (or less crisply defined) interactions. For example, "show all associates similar to Y" could be a very useful query when trying to learn more about person Y. Finally, given a subset $S = s_1, s_2, ...s_n$ of members in the organization C, which might represent a group that is suspect of participating in nefarious activities, a query like "show all subsets of C similar to S" might be an excellent way to discover other suspicious clusters in the organization.

Note that in the example above we did not have any a priori knowledge of the organization other than its structure (which in general is a directed graph) and the elements for which we were searching. In particular, we did not assume any hierarchy, types of relationships present, or any statistical properties of the graph. It was also important that the queries were phrased in a general way using the word "similar" to indicate a degree of likeness, but not (necessarily) an exact match. Such problems occur every day both in the military, intelligence, and defense communities as well as in the business and civilian worlds.

We have structured our algorithms and methodologies to be applicable to any data expressing entities and relationships between entities. Notably, much of the data encountered in the military and intelligence domains deals with entities and the relationships between them, and can therefore benefit from our contributions.

Throughout this paper, we include examples from the movie industry, drawn from a popular and widely accessible dataset about movies, actors, directors, film genres, and so on [5]. One can easily find direct analogies between this data and the types of data encountered in the intelligence and defense domains.

II. Background

The general problem of similarity is twofold: first, to construct a measure of pairwise similarity so that a meaningful similarity score can be calculated for any pair of entities in a dataset; and second, to devise a method for efficiently retrieving the entities that are most similar to a given entity.

In this section, we discuss a similarity measure developed at BBN called the Structural Semantic Distance Measure (SSDM). SSDM is an extension of existing work on calculating similarity over unlabeled, directed graphs. The contribution of SSDM is to incorporate the *semantic* content contained in edge labels, and to achieve a greater independence of ontological choices for edge labeling.

Our work on SSDM builds off the SimRank algorithm by Jeh and Widom [4]. We chose to base our work on SimRank for the following reasons:

- SimRank is domain-independent in that it can be applied to any data representing relationships between entities. This is in contrast with domain-specific similarity algorithms, such as those that can only be used to compare documents [6], ontological categories [7], or some other domain-specific data type.
- SimRank can be computed efficiently in approximation, even over very large datasets, in contrast with measures that rely on Singular Value Decomposition or other computations that scale poorly [8].
- The approximate computation of SimRank can not only determine the similarity between two entities efficiently, but can also generate a list of entities that are most similar to a given entity.
- The computation behind SimRank can be understood on a conceptual level, which makes it possible to explain the similarity score by referring directly to the computation performed. This would not be possible using a similarity measure that relied on more abstract calculations.
- SimRank looks beyond an entity's immediate neighborhood and features when determining similarity, which enables it to incorporate a broader scope of information about the structural context of entities.

All these positive attributes are retained in SSDM, along with several additional improvements.

A. SimRank

SimRank is based on the intuition that "Two entities are similar if they are related to similar entities." While this statement may seem trivial at first, it leads directly to a simple mathematical definition of similarity: the similarity score between two entities is the average pairwise similarity of their neighbors, scaled by a decay factor.

Consider the example in Fig. 1. Intuitively, one would imagine that Movie 1 and Movie 2 should be similar, because they have two actors in common and they are both in the same genre. Additionally, Director 1 and Director 2 should be similar even though they have no immediate connections in common, because they directed similar movies. SimRank captures and formalizes this intuition.

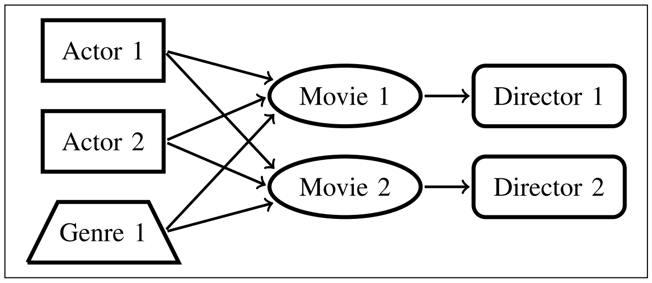

Fig. 1: This figure depicts relationships that exist between entities in a movie dataset. Director 1 and Director 2 have no immediate neighbors in common, but they are similar because they are related to similar movies

Each pair's similarity is dependent on many other pairs, which may seem to be a barrier to computing their scores. Fortunately, this barrier is readily surmountable. On small datasets the system can be solved with an iterative algorithm, and on large datasets it can be solved using an efficient approximate method outlined by Fogaras and Rácz in [9]. Our implementation of the SSDM calculation is based on this efficient approximate method.

The algorithm outlined by Fogaras and Rácz relies on the mathematical notion of a *random walk* through a graph, in which an abstract walker steps from node to node through the graph by following random edges [10]. In the original SimRank paper, Jeh and Widom observed that the SimRank score of two nodes can be approximated from the *expected meeting time* of two random walkers starting at those two nodes; a higher expected meeting time corresponds with a lower SimRank score. Fogaras and Rácz used this observation to develop an efficient and scalable algorithm for calculating similarity scores.

In the algorithm proposed in [9], one random walker is initialized per node in the graph, and each walker moves along one edge per time step. To reduce the amount of computation required, walkers are allowed to *converge* at their meeting point, and are thenceforth treated as a single walker without loss of correctness in the approximate calculation of expected meeting times. Fig. 2 demonstrates how walkers converge. Once the maximum number of steps has elapsed, the run is halted. Repeated runs are performed and the data are aggregated.

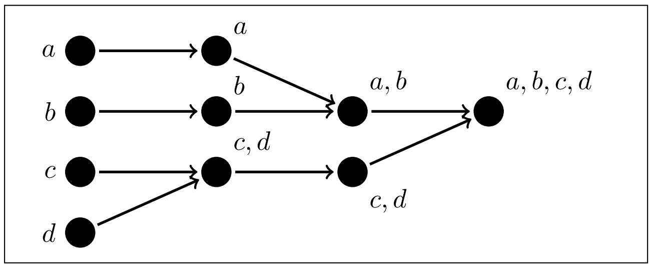

Fig. 2: Walkers a, b, c, and d begin as independent walkers. As they walk (shown progressing from left to right), they meet one another. c and d meet at the first time step and converge. a and b meet next. Finally, on the far right of the diagram, all walkers have converged.

Additionally, to repair some deficiencies with the original formulation of SimRank, walkers in Fogaras and Rácz's algorithm are incentivized to converge if they are near one another. This is accomplished by randomly permuting all vertices in the graph at the start of each time step, with each walker stepping to the neighbor with the smallest index in the permutation.

The end result of one run of Fogaras and Rácz's scalable SimRank algorithm is a *fingerprint graph* which encodes the first meeting times for each pair of walkers. Several runs are conducted, and the fingerprint graphs are compiled into a larger *fingerprint database*. The fingerprint database is precomputed and can be efficiently queried thereafter, either to retrieve a similarity score between any two nodes, or to retrieve the set of nodes with a similarity score greater than some threshold with any given node.

The implementation we devised for calculating the SSDM retains the basic structure of the computation described above, including converging random walkers and permutation-based convergence incentivization.

B. SSDM — The Structural Semantic Distance Measure

The Structural Semantic Distance Measure developed at BBN is closely related to SimRank as described above, with two key enhancements: SSDM incorporates the semantic content of edge labels, and SSDM is independent of ontological choices — namely, which edge directionality each proposition should have.

Whereas SimRank is a measure over unlabeled directed graphs, SSDM incorporates edge labels. This makes it well-suited to any data in subject-predicate-object format, such as RDF or other linked data; subjects and objects are equivalent to nodes in the graph, and predicates are equivalent to labeled edges. The most important use of edge labels is in the way expected meeting time is calculated. The original SimRank computation defines "meeting time" as the time step when two walkers step to the same node, at which point they converge. The SSDM computation has a stricter condition on convergence: namely, when two walkers meet, it only counts as a convergence if they arrive at the same node on the same step *and* they have traversed *identical sequence of predicates* to that node. The reasoning behind this modification is that the semantic meaning of edge labels is critical to the similarity calculation. For example, two entities `A` and `B` may both be related to a third entity `C`, but they are certainly not similar if the relations in question are `A isA C` and `B isNever C`.

Additionally, SimRank only allows similarity to propagate along in-edges, which means that the original computation of SimRank only allows walkers to step backwards (that is, from objects to subjects). This makes SimRank highly dependent on ontological choices, because it is an arbitrary choice in a directed graph whether each label should be phrased in the forward or reverse direction. For example, the relation `A isComponentOf B` could be equally expressed as `B hasComponent A`; the choice of which direction is used is arbitrary and can vary from dataset to dataset. Choosing and enforcing consistent edge directionality is a difficult issue in ontologies in general, so we did not want SSDM to be heavily dependent on arbitrary edge direction choices. As a result, SSDM allows walkers to walk both directions.

Note that allowing walks in both directions requires us to distinguish a walker traversing `A isComponentOf B` from a walker traversing `B isComponentOf A`, as these two steps have very different semantic meanings. Therefore, in SSDM, it is not enough for walkers to simply have traversed identical predicates in order to converge; they must have traversed those predicates *in the same direction* (in or out).

To illustrate the conditions on convergence required by SSDM, consider the example of calculating the similarity between two movies, *War of The Worlds* and *Gladiator*, as shown in Fig. 3. Suppose Walker 1, starting from *War of The Worlds*, traverses the predicates $directed_{in}$, $directed_{out}$, $hasActor_{out}$ to reach Harrison Ford. If Walker 2 follows the same predicates in the same order to reach Harrison Ford, as is shown in Path A, then the two walkers will converge with a meeting time of three steps. If Walker 2 instead follows a different sequence of predicates, such as $hasActor_{out}$, $hasActor_{in}$, $hasActor_{out}$ as shown in Path B, it will not converge with Walker 1 because the two walkers did not follow identical predicates to get there.

SSDM was designed as a domain-independent similarity measure that would easily account for the semantics of labeled graph data without being dependent on ontological choices. This ability to incorporate semantic information is especially important in domains with rich semantic context, and allows SSDM to capture semantic nuances that are missed by less sophisticated similarity measures. The significance of this extra information in the measure is as-yet unassessed, but we expect that SSDM should perform better than SimRank for semantic graphs. In short, SSDM is an efficient, semantically-grounded, and ontology-independent algorithm for discovering similar entities in a linked dataset.

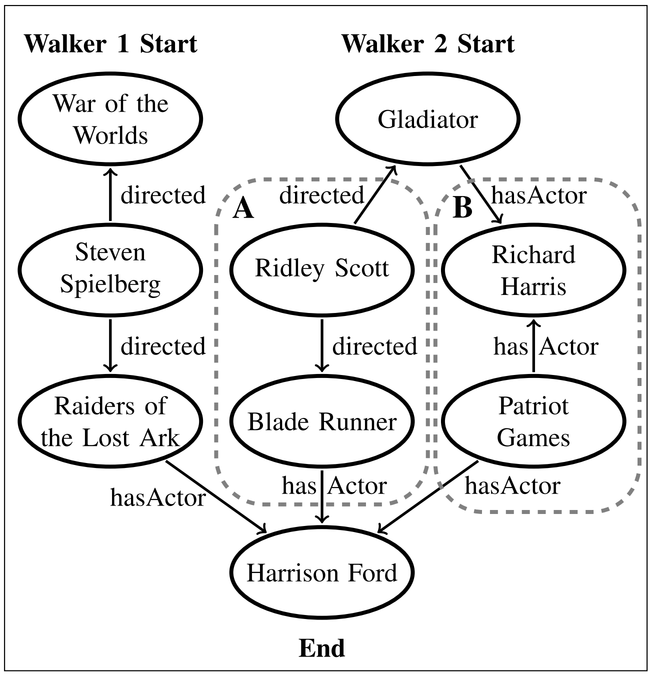

Fig. 3: This diagram depicts two possible ways — path **A** and path **B** — by which Walker 2 could meet Walker 1 at the Harrison Ford node. If Walker 2 takes Path **A**, they will converge. If Walker 2 takes Path **B**, they will not.

III. Explaining Similarities

A similarity measure is useful for ranking items or pairs of items, but a numerical score alone gives little insight into *why* two entities are similar. The user can easily retrieve the similarity score for two entities but is then left wondering: *what about* those entities and their relations caused them to receive a high or low similarity score? What is the nature of their similarity? In addition, users may want more or less depth in the explanations provided.

In order to enable users to answer this question, we sought to build a system that could provide *explanations* for similarity scores. Our three main contributions to the area of similarity explanations are as follows:

1) We formulated a definition of an "explanation" for a similarity score that is human-understandable, as well as appropriately grounded in the way that the similarity score was originally calculated.
2) We wrote a program to efficiently produce such explanations.
3) We further developed a methodology for biasing explanations towards either more "obvious" or more "obscure" facts.

IV. Definition of an Explanation

A good explanation of a similarity score must be both human-understandable and grounded in the original calculation of the similarity measure. An explanation that is not human-understandable is hardly an explanation at all, and an explanation that is generated by a computation which is entirely unlike the original calculation of the similarity score can hardly be considered an explanation of why that score was produced.

Recall that the SSDM computation calculates similarity scores based on repeated runs of converging random walkers. The faster two random walkers tend to converge, the higher the similarity score of their starting nodes will be.

It follows that for an explanation of a score to be well-grounded in how SSDM scores are calculated, it must somehow elucidate *where* and *how* walkers from the nodes in question tend to converge, and whether these convergences tend to happen rapidly or whether the walks are long. In order for this information to also be human-understandable, it must be relatively concise.

For these reasons, we decided that an explanation should consist of a brief *list of common convergence points*, along with a handful of concise *chains of statements* per convergence point describing the "best" relationships linking each starting node to that point. The "best" relationships may be the shortest chains of statements, or the ones that tend to be traversed most frequently, or (as explained later) relationships that are appropriately obvious or obscure. Figure 4 shows an example of an explanation of this form.

```
John Williams
  {A New Hope, hasMusicContributor, John Williams}
  {The Empire Strikes Back, hasMusicContributor,
    John Williams}

Harrison Ford
  {A New Hope, hasActor, Harrison Ford}
  {The Empire Strikes Back, hasActor, Harrison Ford}

Star Wars (Film Collection)
  {A New Hope, inCollection, Star Wars}
  {The Empire Strikes Back, inCollection, Star Wars}

  {Revenge of the Sith, hasSequel, A New Hope}
    {Revenge of the Sith, inCollection, Star Wars}
  {A New Hope, hasSequel, The Empire Strikes Back}
    {A New Hope, inCollection, Star Wars}
...
```

Fig. 4: An excerpt from an explanation for the similarity between *Star Wars Episode IV: A New Hope* and *Star Wars Episode V: The Empire Strikes Back*, showing both one-relationship chains and multi-relationship chains

By defining an "explanation" this way, we ensure that users are presented with a coherent explanation of *where* and *how* walkers from the nodes in question tend to converge. Additionally, such explanations are also readily understandable as explanations of what commonalities the nodes have, and how they are related to each commonality.

V. Our Approach

The most obvious way to extract an explanation for a computation seems to be to inspect the path that the computation followed to obtain its result. Unfortunately, in the case of our

SSDM calculation, such a strategy is inadequate. We have established that we would like to present chains of relations to the user. However, the fingerprint graphs produced by the SSDM computation record only when and where the walkers converged, discarding all information about the path taken by the walkers; furthermore, discarding this information is essential to the calculation as a whole to maintain acceptable space and performance characteristics. While simply listing convergence points does provide the user with *some* intelligible information, it does not provide as rich an explanation as we would like.

Therefore, our approach to explanation generation is to re-run the SSDM calculation on a smaller scale at query-time, and explicitly store the relations traversed rather than condensing the results into a terse fingerprint graph. Two alterations were required to make the modified SSDM calculation efficient enough to provide an acceptable user experience at query time. Both performance improvements were achievable because the modified calculation uses just two walkers rather than starting one walker at every node in the graph. Recall that the similarity of two entities is derived from the expected meeting times of walkers starting at those two entities. If we know in advance which two entities we will be comparing, there is no need to start walkers at any other entities. Because it is so efficient to generate a pair of walks compared to a whole graph's worth of walks, we can afford to run the computation many times per explanation request, and then choose from among the possible explanations to display relevant results to users.

The first performance improvement strategy relates to the permutation-based convergence strategy described in Section II-A. Each step of the ordinary SSDM calculation begins by randomly shuffling all edges in the graph. In the modified calculation, only the two walkers' immediate out-edges need to be shuffled, which is almost always a vastly smaller number of edges, and takes a negligible amount of time.

The second performance improvement strategy relates to the strict edge-label requirements for convergence. In the ordinary SSDM calculation, walkers frequently meet at the same node but do not actually converge because they did not follow the same predicates. In the modified calculation, we instead *require* the two walkers to follow the same predicates as one another. So, in the example given in Fig. 3, path B would never be generated; instead, either Walker 1 and Walker 2 would both follow a `directed` in-edge, or both would follow a `hasActor` out-edge, or both would follow some other shared edge not shown. If the two walkers were ever unable to follow the same predicate in the same direction, then that run of the computation would end. Coupling the edge options of the two walkers greatly reduces the number of trials that fail to converge, leading to a much more efficient calculation.

As a proof-of-concept for this methodology, we implemented an explanation-generating component of Parliament, an open-source triple store developed and maintained by BBN [11]. In Parliament, users are able to browse and view entities in the knowledge base, explore other triples containing an entity, and view a list of similar entities and their SSDM scores. As part of our work on similarity explanations, we added a "why" button for each score, which users can click to produce an on-demand explanation of that score. The software behind the "why" button interacts with the underlying Jena[1] model of the data in Parliament to walk the graph and produce an explanation using the methodology described in this section. Even for datasets with millions of triples, such as the movie dataset, preliminary findings show that accurate explanations could be produced and displayed to the user within seconds.

In summary, explanations for SSDM scores can be efficiently computed on-demand at query time by re-running a modified version of the SSDM random walker calculation.

VI. Using Salience to Bias Explanations

The final contribution we describe in this paper is a method for incorporating *salience* into explanation generation. Salience is a measure of how rare a fact is in a dataset. Our goal was to produce the most "useful" explanations, which we believed would be the explanations with the *most salient* facts, because salient facts are rare and therefore highly descriptive.

We instead discovered that high-salience explanations often come across as obscure because they can contain extremely rare facts. Similarly, low-salience explanations often come across as obvious because they contain extremely common facts. Nonetheless, just as high-salience explanations often have the upside of being very descriptive, low-salience explanations often have the upside of revealing the broadest and most general similarities rather than obscure trivia. Which flavor of explanation is more "useful" likely depends on the goals of the user and requires more research in an application domain.

In this section we present the definition of salience as applied to facts and explain how we constructed a salience-weighted edge generator so that the explanations generated would contain more or fewer high-salience facts. Note that while the following descriptions will focus on biasing explanations by salience, it can be used to favor facts based on any numerical property of those facts.

A. Fact Salience

Salience is a measure of how rare a fact is in a dataset. A *fact* in this case refers not to a whole statement, but to a statement missing its subject or object; that is to say, structures of the form `subject predicate` *`blank`* or *`blank`* `predicate object` (such as `Spielberg directed` *`blank`* or *`blank`* `directed Gladiator`). Facts with a subject and predicate are called left facts because all the information they retain is on the left side of the statement. Similarly, facts with a predicate and an object are called right facts [12].

We now describe how the salience of a fact is calculated. Consider $o(f)$ to be the number of times a fact is expressed in a set of unique `subject predicate object` triples.

[1]http://openjena.org

Consider also $subj$ to be the number of unique subjects present in those triples, and obj to be the number of unique objects. For left facts, salience is calculated as follows:

$$salience(fact) = \frac{1 - log(o(fact))}{log(obj)} \tag{1}$$

And for right facts, the calculation is as follows:

$$salience(fact) = \frac{1 - log(o(fact))}{log(subj)} \tag{2}$$

The conceptual significance of $subj$ and obj in these equations is to count the number of times each fact could potentially occur, since each left fact could potentially appear with every object in the data set, and each right fact could potentially appear with every subject. Facts that actually *do* appear with almost every available subject or object are extremely common, and are thus not very salient. Facts that are expressed about very few of the available subjects or objects are very rare and therefore highly salient. This intuition, and the resultant calculation, is grounded in the information theoretic concept of relative entropy, discussed in [13].

B. Weighting

The objective of salience weighting is to favor high- or low-salience facts in the generated explanations. This bias can be incorporated into the random permutation that is calculated at the beginning of each time step. In the unweighted explanation calculation, the random walkers choose which edge to take by stepping to the neighbor with the smallest index in this permutation. The original reason for the permutation was to encourage walkers that are near one another to converge, but it can also be modified to add other weights and biases into the explanation-generation process.

In order to encourage walkers' edge choices to favor more salient edges, we would like high-salience edges to be more likely to occur at low indices in the permutation. However, we do not want the distribution of salience in the permutation to be too consistent from trial to trial, otherwise low-salience edges will never reach the top of the rankings, restricting the edge choice of the walkers and severely limiting the breadth of explanations produced.

The permutation as originally described in Fogaras and Rácz ranks *nodes* randomly; however, the salience of a node is not a well-defined concept, and so to enable a salience-weighted permutation algorithm it was necessary to switch to ranking *facts*. This modification was justified for the following reason. Under Fogaras and Rácz's formulation, convergence required only that two walkers meet at the same point, and so to encourage convergence it was enough to encourage walkers to choose the same nodes to step to — hence *nodes* were what was ranked. However, under our formulation, convergence requires that the two walkers also follow the same edge label in the same direction. An edge label plus one endpoint makes up a fact, so to encourage convergence, we are justified in encouraging walkers to choose the same facts to walk along.

The weighted-permutation algorithm we developed works according to the common permutation strategy of assigning a random number to each element to be permuted and then sorting by those numbers. We devised a method to incorporate salience into the generation of those random numbers.

We made two attempts at designing the weighting algorithm. Our first, unsuccessful attempt at a biased algorithm contained an oversight which we corrected in the second version. The first, naive approach worked as follows:

1) Each fact f_i is associated with a nonnegative weight w_i
2) Each fact f_i is assigned a random number r_i between 0 and w_i
3) Elements are ranked in ascending order according to r_i

Using this algorithm, elements with a *low* weight are more likely to get a low number relative to other elements, and are therefore more likely to be ranked first.[2] To use this algorithm to favor salient facts, the weight function used to assign the w_i values could be set to $w_i = 1 - salience(f_i)$, or some other function such as $w_i = 1 - \sqrt{salience(f_i)}$.

When testing this algorithm using $w_i = 1 - salience(f_i)$, a failure mode arose: namely, the very same facts would appear at the top time and time again. We determined that the problem occurred with unique facts because their salience is precisely 1, and their weight was therefore 0. When unique facts were assigned a random number between 0 and their weight they were always assigned 0. This meant that all the unique facts were always first in the list, and so only unique facts were ever generated.

To remedy this problem, we added a parameter to increase the randomness. The modified, successful algorithm works as follows:

1) Each fact f_i is associated with a nonnegative weight w_i
2) Given a parameter b, each fact is assigned a random number r_i between 0 and $b + (1 - b) * w_i$
3) Elements are ranked in ascending order according to r_i

In this way, the parameter b can be tuned to increase or decrease the randomness of the permutation.

The properties of the weight function do not constrain the calculation, so most any weight function could be used. To favor low-salience facts, for example, salience or the square root of salience can be used directly. To favor medium-salience facts, $w_i = salience(f_i) * (1 - salience(f_i))$ could be used. Any number of other functions are possible depending on the desired salience characteristics of the resulting explanation.

C. Relevance of Salience-Weighting

As for whether salient facts are actually more useful, it seems to depend on the intended use case. In some cases, obvious paths are more useful, and in other cases, obscure paths are more useful. If the user knows very little about the area of inquiry he or she is likely to prefer explanations that refer to common and well-understood facts and properties. Conversely, if the user is an expert in the domain, he or she is likely to

[2]"Weight" might be a misnomer here as it implies elements with high weights are favored, but the opposite is the case with this algorithm.

prefer more obscure data. A user with average expertise will probably want only middle-salience explanations.

For example, consider a movie-watcher who has never seen any *Star Wars* movies. He or she may be interested to know about low-salience (i.e. common) facts like these:

```
{ofGenre, Science Fiction}
{hasMusicContributor, John Williams}
{hasActor, James Earl Jones}
```

These facts reference well-known people and broad genres, which could help give a novice a grasp of what relates the *Star Wars* movies to one another. High-salience facts such as the following:

```
{hasProducer, Gary Kurtz}
{hasDirector, Irvin Kershner}
{hasEditor, T. M. Christopher}
```

would be too obscure; an unfamiliar viewer is unlikely to know who, say, Irvin Kershner is, as he is not well-known for directing any other blockbusters. However, a *Star Wars* afficionado who already knows that the *Star Wars* movies are Science Fiction films will find the low-salience facts too obvious. He or she may be interested in knowing about the more unusual details that relate *Star Wars Episode IV: A New Hope* and *Star Wars Episode V: The Empire Strikes Back* to one another, and would be pleased to discover that the relatively-unknown editor T. M. Christopher was involved in the production.

Fortunately, our method enables the salience to be weighted by a custom weight function, so that the user can tune the salience of the resulting explanations to his or her needs. Additionally, because explanations are generated on-demand, the desired obscurity could be specified at query time, which would enable the user to ask for more or less obscure explanations in real time as they explore their data.

VII. Applications

Our work on semantic similarity has been developed with an eye towards a variety of applications, primarily in the military and intelligence domains. In general terms, similarity measures and explanations are very useful tools for analysis of large graph-based datasets.

Consider a simple example. Suppose we are collating information on Libya and we encounter the profiles for the following individuals:

- Muammar al-Gaddafi
- Muammar El-Gadhafi
- Moammar Kadaffi

Despite considerable variation in their spellings, these names all refer to the same Libyan former head of state. In fact, some sources report over one hundred ways to spell this person's name [14] due to ambiguities in the transliteration from Arabic. It would be ideal if we can use additional information to disambiguate the names in order to ascertain that these profiles represent the same person. Using information about the relationships (and actions) of the person from each profile, we could derive similarity scores for each pair of profiles, and merge those profiles as appropriate.

In another setting, suppose we are monitoring the network activities of a group of employees of company X. Each employee belongs to one of four departments: Engineering, Sales, Finance, or Corporate. We can monitor email traffic, access to corporate applications, printers, file repositories and other network activities. Let's focus on George, who is in Sales. We receive alerts that George has been accessing financial software and financial projection data files. Is this activity unusual? Our algorithm could be applied to compare George and the profile of his activities with those of employees at his and other departments. Does George seem to be behaving more like employees in Finance than he was before, or less similarly to his fellow employees at Sales than we might expect? If so, it certainly indicates that George's behavior has changed for some reason, and may warrant investigation.

In essence, our similarity work can be applied to any data expressing relationships between entities. Our algorithm is highly scalable, so it can be applied on very large datasets, and our work on explanations allows the similarity results to be clearly communicated to end users of the data analysis. For these reasons, we believe our work to be highly applicable to a wide variety of military and intelligence tasks.

VIII. Future Work

The future of this work lies in two directions. The first is to perform experiments to assess the improvement of using SSDM as compared to SimRank and other approaches. We would also like to perform user tests to determine the perceived utility of different explanations in a real-world environment. The second direction lies in further research on extensions to the work and on new approaches to enhance it.

Experimentation with SSDM will likely take place in a relevant application domain such as social network analysis (in the intelligence domain) or computer network activity analysis (the cyber domain). Metrics for the meaning of similarity will have to be developed for each domain before the algorithm can be evaluated. This is especially true with user testing, where the experiments need to account for subjectivity and prior domain knowledge. Selecting and vetting the appropriate data sets for automated evaluation is another challenge — graph-based data, which is manually annotated for similarity, does not appear to be very common. One approach we may take is to compare structural similarity generated by SSDM to similarity derived from entity attribute comparison (presence and value of certain attributes). Many well-established algorithms exist in this area to provide a baseline for attribute-based similarity.

The most promising direction of further research we envision lies in the domain of calculating *predicate* similarity. With our current algorithm, walkers must traverse identical predicates — a strategy designed to prevent relations such as `A is C` and `B isNever C` from contributing positively to `A`'s similarity with `B`. However, it intuitively seems that `A is C` and `B isOften C` should certainly contribute to `A` and `B`'s similarity. Doing so would require calculating the similarity between predicates (in this case, `is` and `isOften`) before calculating the similarities between entities. One possible way to do this would be to run SSDM on the ontology to calculate predicate similarity before moving on to calculate object similarity. Another possible option would be to use a language-based metric as a source of predicate similarity, such as by using WordNet [15] similarity.

A third option considers the insight that "similar predicates are those that connect similar entities to other similar entities," for example, the predicates `teaches` and `hasInstructor` are considered similar because they appear in relationships such as `Dr. Smith teaches Chemistry` and `CH1301 hasInstructor Dr. Jones` where `Chemistry` is known to be similar to `CH1301` and `Dr. Smith` is similar to `Dr. Jones`. This formulation of predicate similarity is obviously recursive with respect to entity similarity: in order to calculate predicate similarity, we must first calculate entity similarity, and vice versa. However, SimRank, SSDM, and many other algorithms are also based on recursive definitions and derive iterative approximations to the optimal result set. Logically then, we could apply this recursive definition of predicate similarity in order to simultaneously derive both entity (subject and object) and predicate similarity. This is reminiscent of similar iterative and approximate approaches in robotic navigation to solve simultaneous localization and mapping (SLAM) [16][17] problems. It is reasonable to expect that predicate similarity may be derived analogously.

Additional directions also include determining what other weighting schemes besides salience-weighting might be useful. For example, if a dataset includes metadata about provenance or other information about the trustworthiness of each fact (node in the graph), then the SSDM calculation could be weighted to favor more trusted facts over facts from less reliable sources. Other possibilities surely exist.

And finally, while the current explanation format is certainly human-understandable, it does not yet read as easily as text. Fortunately, the data in question is already in subject-predicate-object form; that is to say, it is already in a sentence-like structure, and therefore natural language report generation is a goal well within reach. The additional work required would include determining what sentence structures each predicate fit with, and determining how to ensure that the subjects and objects were tagged with human-readable labels that could be included in a generated report (and not just URIs or other machine-readable descriptors).

IX. Conclusion

Questions of similarity crop up any time users want to make sense of data describing relationships between entities, and data of this form (i.e. graph data or linked data) is ubiquitous. The contributions we describe in this paper help users find similarities in graph data efficiently using SSDM, and understand those similarities using similarity explanations. We defined what an explanation of a similarity score should convey; we implemented a system that can produce such scores and explanations efficiently; and we enabled the obscurity of our explanations to be tuned to meet user's needs. We believe that these contributions will help users in the intelligence and defense communities to make sense of their data, by enabling them to not only find relevant similarities more efficiently, but also to understand those similarities on an intuitive level.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, May 2008. [Online]. Available: http://www.ds3web.it/miscellanea/the_semantic_web.pdf

[2] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data — the story so far," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 5, 2009. [Online]. Available: http://eprints.ecs.soton.ac.uk/21285/1/bizer-heath-berners-lee-ijswis-linked-data.pdf

[3] T. Heath. (2011) Linked data — connect distributed data across the web. [Online]. Available: http://linkeddata.org/

[4] G. Jeh and J. Widom, "SimRank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '02. New York, NY: ACM, 2002, pp. 538–543. [Online]. Available: http://doi.acm.org/10.1145/775047.775126

[5] M. P. Consens, O. Hassanzadeh, and A. M. Teisanu. (2008, September) Linked movie database. [Online]. Available: http://www.linkedmdb.org/

[6] A. Islam and D. Inkpen, "Semantic text similarity using corpus-based word similarity and string similarity," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 2, pp. 1–25, July 2008. [Online]. Available: http://doi.acm.org/10.1145/1376815.1376819

[7] F. M. Couto, M. J. Silva, and P. M. Coutinho, "Measuring semantic similarity between Gene Ontology terms," *Data and Knowledge Engineering*, vol. 61, no. 1, pp. 137–152, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169023X06000875

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. [Online]. Available: http://books.google.com/books?id=si3R3Pfa98QC

[9] D. Fogaras and B. Rácz, "Scaling link-based similarity search," in *Proceedings of the 14th international conference on World Wide Web*, ser. WWW '05. New York, NY: ACM, 2005, pp. 641–650. [Online]. Available: http://doi.acm.org/10.1145/1060745.1060839

[10] D. Aldous and J. Fill, "Reversible markov chains and random walks on graphs," 2002, in preparation. [Online]. Available: http://stat-www.berkeley.edu/users/aldous/RWG/book.html

[11] D. Kolas, I. Emmons, and M. Dean, "Efficient Linked-List RDF Indexing in Parliament," in *Proceedings of the Fifth International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*, ser. Lecture Notes in Computer Science, vol. 5823. Washington, DC: Springer, October 2009, pp. 17–32. [Online]. Available: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-517/ssws09-paper2.pdf

[12] Commonsense Computing Group. (2011, June) Tutorial: Making matrices from your own data. [Online]. Available: http://csc.media.mit.edu/docs/divisi1/tutorial_make.html

[13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: Wiley, 1991. [Online]. Available: http://www.elementsofinformationtheory.com/

[14] S. Bass, "How many different ways can you spell 'Gaddafi'?" *ABC News*, September 2009. [Online]. Available: http://abcnews.go.com/blogs/headlines/2009/09/how-many-different-ways-can-you-spell-gaddafi/

[15] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to wordnet: An on-line lexical database," *International Journal of Lexicography*, vol. 3, no. 4, pp. 235–244, 1990. [Online]. Available: http://wordnet.princeton.edu/

[16] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, pp. 56–68, December 1986. [Online]. Available: http://dl.acm.org/citation.cfm?id=33838.33842

[17] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *IEEE/RSJ International Workshop on Intelligent Robot Systems (IROS)*, November 1991, pp. 1442–1447. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=174711&tag=1