View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 95 (2016) 327 – 334

Procedia Computer Science

Complex Adaptive Systems, Publication 6
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2016 - Los Angeles, CA

Vector Representation for Sub-Graph Encoding to Resolve Entities

Jinhong K. Guo*, David Van Brackle, Nicolas Lofaso and Martin O. Hofmann

Lockheed Martin Advanced Technology Laboratories
3 Executive Campus, Suite 600, Cherry Hill, NJ 08002, USA

Abstract

Entity Resolution, i.e., determining whether two mentions refer to the same entity, is a crucial step in combining evidence from multiple sources, and is a problem encountered in a wide-range of areas, from modeling causes of cancer to identifying terrorist networks. Entity mentions are represented by attributes and relations to other entities. However, entity attributes and relations from different sources often use different names and specify relationships differently, which leads to low entity resolution precision and recall. Our contribution is based on our observation that relationships are more reliable than attributes when comparison is based on relational similarity, not exact matches. Traditional graph comparison techniques rely on finding precise matches of a significant part of the graph structure, and require custom comparison functions for every type of attribute and every type of relation. This leads to a system that is difficult to maintain and enhance. We encode entity nodes and their graph neighborhoods in semantic vectors, efficiently indexing the vectors, and calculating vector similarity. Our approach is insensitive to small variations in relational graph representation. Our approach uses simple vector addition, permutation, and difference only, leading to reduced computational complexity. Our preliminary experiment shows 83.05% accuracy.

© 2016 Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: Entity Resolution; Vector Representation; Graph Matching;

* Corresponding author *E-mail address:* jinhong.guo@lmco.com

1877-0509 © 2016 Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2016.09.342

1. Introduction

Entity Resolution is a crucial step in combining information from multiple sources. A common use case is fusing the information about an entity mentioned in a new report with the information accumulated about the entity in a data repository. Most of the relevant entity information can be represented as attributed, relational graphs. As the nodes and links in the graphs are from different resources, the key challenge is to correctly associate new observations, which typically only provide a few attributes and/or features in a sparse graph fragment, with nodes and links in the graphs are from different resources, the key challenge is to correctly associate new observations, which typically only provide a few attributes and/or features in a sparse graph fragment, with the stored entity object that has accumulated all attributes and features observed so far. A second challenge is that text-based observations in reports or social media and image or video-based observations often arrive out of order due to varying, potentially long delays in report generation and exploitation, so that entity resolution decisions may have to be reconsidered and sometimes revoked. A third challenge is that some attributes change over time, e.g., people adopt new aliases, join different groups.

Our technical approach is to encode graph fragments in high-dimensional semantic vectors that can be rapidly indexed and compared, and are insensitive to small variations in graph structure. Our recent research in using semantic vectors to encode the meaning and structure of language [1][2][3] initiated the idea of flattening the attributes and relationships using a high dimensional vector representation. Our algorithm uses a weighted distance function between these vectors to determine the similarity between pairs of nodes. Preliminary results showed that the vectors correctly represented the graphical structure around a given node. We conducted a preliminary study using a small set of sparse, fragmented graphs generated from reports. The experiment only considered the graph structure, i.e., the relationships between entity nodes (ignoring entity attributes), but still achieved 83% accuracy for strong matches. For example, it matched a person using two different names because of the consistent relations to other people, organizations, and places.

2. Related Research

Traditional graph comparison techniques rely on finding precise matches of a significant part of the graph structure. However, different source data will often yield slightly different graphs for the same entity that cannot be matched by these techniques. Traditional techniques also require custom comparison functions for every type of attribute and every type of relation. This leads to a system that is difficult to maintain and enhance. In contrast, our approach is insensitive to small variations in graph representation, e.g., relational structure and relation type, since it compares the relational structure of the graphs using a vector space representation. It uses vector distance as a generic comparison metric, but we augment the metric for attributes with well-established semantic distance metrics, e.g., for location and time attributes. Finally, computational complexity is another challenge for traditional approaches. Our approach uses simple vector addition, permutation, and difference only, leading to reduced computational complexity.

Fusion of redundant nodes is predicated on a close match of a pairwise comparison of the attributes of the two entities. Text-valued attributes may be compared based on edit distance and numeric attributes based on value differences. A simple approach for computing the probability that the pair is a match is to use a weighted sum of the individual attribute similarity scores. Another popular approach is rule based matching tailored to the specific data types of the application domain. This may work well with some specific, narrowly-scoped applications. However, manual formulation of the rule sets is labor-intensive and hard to generalize. As analysts exploit a greater variety of data sources and greater volumes of data, including text and images from open source, traditional techniques are ill suited to adapt to changing vocabulary, attributes, and data formats, since the data is more heterogeneous and a large percentage is unstructured, incomplete, and of different modalities. Furthermore, the data are more tightly linked and multi-relational (e.g., Walmart, the chain of discount department store or Walmart Pharmacy, the pharmacy within Walmart) [4], which further increases the complexity of hand-crafting comparison rules.

Recently, realizing that relationships are crucial, techniques have been developed to include information about entity relationships, which show greater robustness [5][6][7][8][9][10]. One of the simplest approaches is to use relational features as if they were attributes to provide a richer context for matching. Examples of such features include attributes of the edge (e.g., label on the edge) or neighboring entity (e.g., name), aggregates of related attributes, as well as sophisticated relational features constructed using First Order Logic. Another approach is to use set similarity measures to compare nodes based on a set of related nodes, e.g., comparing neighborhoods with overlap, Jaccard coefficient, and average similarities between set members. Adding relationships to the similarity computation has proven to enhance both precision and recall for entity resolution.

However, most approaches still fail to address some of the challenges described earlier. Most of the performance evaluations published use only a static dataset, such as resolving authors based on a bibliography database, where all the information associated with each node (author) is available at once. In a bibliography database, the changes for each author happen infrequently and revoking an existing relationship is generally not an issue. In military applications, however, information about different entities of interest changes quickly and frequently. This information not only includes never-seen-before attributes and relations, but also corrections of previously asserted attributes and relations. Our approach is to encode the attributes of a node and its relationships with its (directly and indirectly connected) neighbors into a fixed length N-vector. This vector is being updated as new information regarding the node arrives. The graph similarity is measured by the similarity of the vectors that encode the graph node and edge attributes and topology.

3. Technical Approach

The idea of our approach is to start with a high dimensional space in which the many different attributes and the relationships of an entity with the rest of the graph can be represented. Then we project these high-dimensional data onto a lower-dimensional subspace, an N-dimensional space. The resulting vectors, encoding the attributes and relationships, are stored with each node in the graph.

Widdows and Cohen [13] gave a detailed description of reasoning with high dimensional vectors. The core concepts and algebraic operations are summarized in the following.

- Random vectors can be used to represent concepts
- High-dimensional, randomly generated vectors are nearly orthogonal. Thus, the similarity between two vectors is near 0. The aspect of randomly generated vectors makes them discriminative.
- *Superposing* two vectors x and y generates a third vector $\boldsymbol{x+y}$ such that $\boldsymbol{x\cdot(x+y)}$ and $\boldsymbol{y\cdot(x+y)}$ are relatively large, indicating similarity. Here, + and $\cdot$ are element-wise addition and multiplication. This enables retrieving one vector from the sum of vectors by using multiplication of the vector into the sum. This allows testing whether a certain attribute is part of an entity vector representation.
- *Binding* two vectors x and y generates a third vector $\boldsymbol{x\otimes y}$, such that $\boldsymbol{x\cdot(x\otimes y)}$ and $\boldsymbol{y\cdot(x\otimes y)}$ are usually near 0. However, if y and y' are close to each other, $\boldsymbol{x\otimes y}$ and $\boldsymbol{x\otimes y'}$ should be close. This means that binding two vectors generates a vector that is different from either original vector. However, if any two vectors are close in the high dimensional vector space, binding to the same vector will not change the closeness of the resulting two vectors. As we will discuss below, the operation of binding allows us to represent a value associated with a specific attribute. As long as the attribute values are similar, binding their vector representation with the vector representation of the attribute name will not diminish their similarity.
- There is an inverse of binding, $x \oslash y$, such that $(x \oslash (x \otimes y)) \cdot y \approx 1$, which enables unbinding of two vectors (unbinding y from x), resulting in a vector that is very close to the original vector (y). This is useful for updating the entity vector with changed attribute values or relations.

For the work described in this paper, we chose a binary form of random vectors, called Random Indexing (RI) vectors, to encode the node and edge (attributes and relationships) information at each node. Random Indexing is computationally simple and highly scalable, especially compared to techniques that compute semantic vectors by training a multi-layer neural network. Our implementation is based on the high dimensional random vector techniques described by Kanerva [11]. Encoding of the attributes is realized by the binding operation and simple vector addition. The binding operation is implemented through the XOR logic function ($*$). There is another reversible operation, vector permutation [12] that we use to represent relationships. A more detailed description of our approach follows.

3.1. Encoding Attributes and Relations of an Entity

To bind a specific value to an attribute, we use vector multiplications. For example, if A is the vector for an attribute, e.g., "name", and V is the value for the attribute, for example, "Alice", the resulting vector that represent the statement that the name being Alice is defined by $A * V$. It is important in our application domain that the RI vector representation is able to easily change attribute values, for example, a person's name. To replace an existing value V with a new value V', we can simply perform (see [11] for mathematical derivation)

$$V' * (A * V) * V = A * V' \tag{1}$$

This simple technique still works when we replace a single attribute value in the composite representation of an entity with many attributes. Assuming an entity has n attributes, each with attribute name represented by vector N_i, and its value, represented by vector V_i, the resulting vector that encodes the multiple attributes of one entity is defined by,

$$V_e = A_1 * V_1 + A_2 * V_2 + \cdots + A_n * V_n \tag{2}$$

To replace a specific attribute, e.g., V_1 with V_1', we plan to perform

$$V_e - A_1 * V_1 + A_1 * V_1' = V_e + A_1 * (V_1' - V_1) \tag{3}$$

However, we have not yet explored how to best realize the subtraction operation in the RI framework. To do this, we believe that we will need to use real number vectors [13] instead of the binary vectors we have been experimenting with.

To summarize, high dimensional random vectors allow us to

- enrich the vector representation of an entity with additional attributes, e.g., adding a new attribute A_{n+1} with value of V_{n+1} as attributes are discovered, by simply adding it to the existing semantic vector
- correct erroneous attribute values with new values as discussed above (Equation 3).

To illustrate with an example, an entity Alice with two attributes can be represented in the form of

Alice	
Name	Alice
Affiliation	ATL

$$V_{Entity\_Alice} = A_{name} * V_{Alice} + A_{affiliation} * V_{ATL} \tag{4}$$

where V_{Entity_Alice} is the vector representation for the entity Alice and $A_{name}, A_{affiliation}$ are the vector representation denoting attributes of name and affiliation and V_{Alice}, V_{ATL} are the vector representation of the values to these attributes.

3.2. Encoding Graph Structures with Permutation

Permutation has been shown to efficiently encode order in a sequence of elements. We use it to encode the graph structures centered on an entity by encoding the chain (direct, one-level, two-level connections) of its relationships. If we permute one of the vectors in a pair of bindings (A*V), e.g., A, we have $B = \Pi A * V$, where Π is the permutation matrix. Permutation can be easily reversed.

$$\begin{gathered} \Pi^{-1}(B * V) = \Pi^{-1}\Pi A * V * V = \Pi^{-1}\Pi A = A \\ \Pi A * B = \Pi A * \Pi A * V = V \end{gathered} \tag{5}$$

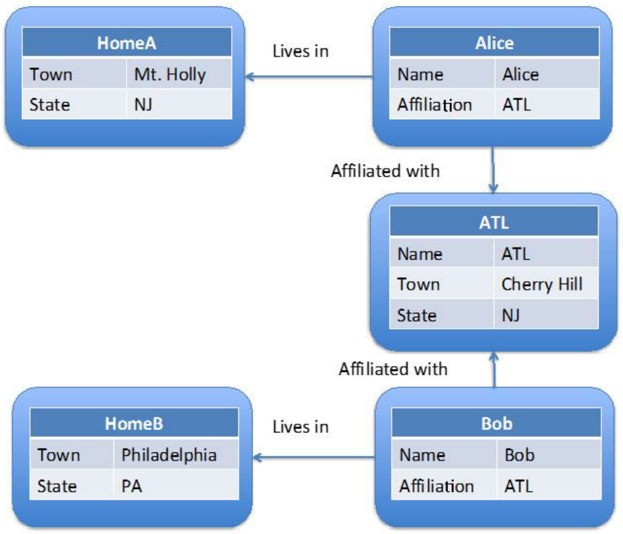

Fig 1 An example graph representing the relationships between different entities

We attach the information that an entity has a relationship with another entity similar to how we attach attributes and their values, but use vector permutation to indicate the distance of the relationship: a single permutation for a direct relationship, a double permutation for a second-level relationship and so on. Using the example in Fig 1., Alice has a direct relationship with entity nodes ATL and HomeA. She also has a second-level relationship with Bob and a third-level relationship with Bob's HomeB. We constrain the length of the chain of connections and only model relations up to two levels of connection.

$$V_{Alice} = A_{Entity\_Alice} * V_{Entity\_Alice} + \Pi A_{affiliated} * V_{Entity\_ATL} + \Pi\Pi A_{affiliated} * V_{Entity\_Bob} \qquad (6)$$

3.3. Preliminary Experiments

In previous research, we experimented with RI encoding of the semantic meanings of words and analyzed the results in [1][2][3]. Our next step was to evaluate the performance of using permutation to encode the graph structures. For this experiment, we exclusively represented the relational structure of the subgraph, ignoring the attributes, in a single vector per entity. We use the same form of Random Indexing vectors as in [1][2][3]. They are sparse high dimensional vectors of elements 0, 1 and -1.

We generated a set of entities from a collection of reports using our Entity Extraction algorithm [14]. Our data source contained 79 text reports that mention 32 different persons total. The only directly available attribute in the data for a person is his/her name. We were able to use each person's unique name as the ground truth, since we only used the graph structure, not the attributes, nor the labels of the type of relationships, in this experiment. We classify the relationships into two types in this experiment, i.e., positive or negative relationship. By positive relationship, we refer to any relationships with a friendly or neutral sense, such as *friend of* or *work with*. By negative, we refer to relationships such as *attacked by*, etc. The encoding scheme we use for the example in Fig 1 is as following.

$$V_{Alice} = A_{Entity} * V_{Entity\_Alice} + \Pi(A_{positive} * V_{Entity\_ATL}) + \Pi\Pi(A_{positive} * V_{Entity\_Bob}) \qquad (7)$$

With only one type of attribute, we simply chose A_{Entity} as an identity vector, such that $A_{Entity} * V = V$. Similarly, with only two types of relationships, we chose the operation of $A_{positive} * V = V$ and $A_{negative} * V = -V$. The relationships in the subgraphs are generated from the events, organizations and places that are described in each of the reports. Due to the limited number of reports, we separated the data (reports) into n sets and used n-fold cross-validation to assess the results. Our procedure was:

- Sort the files alphabetically and number them 1 to 79.

- Select
  - The 10 odd files of file 1-20 (1, 3, 5, … 19)
  - The 10 even files of file 1-20
  - The 10 odd files of file 21-40
  - The 10 even files of file 21-40
  - etc.

Using the test set, one file at a time, the procedure continued as follows:

1. Run our Entity Extraction (EE) software to extract entities and their relationships.
2. Use one set of the data as test set, build the "base" (comparison) relational graph from the remaining sets
3. Build the semantic vector for each of the observed entities extracted from the file, using the relationships extracted from that file only
4. Compute the vector similarity of the observed entity in question with rest of the entities
5. Pick the highest ranked entity in the database (in terms of similarity score) as the candidate for resolution
6. If the similarity score > threshold_1, declare that the observed entity is an observation of the highest ranked entity
7. If the similarity score is in between threshold_1 and threshold_2, we assume there is a possibility that the observed entity is a observation of the highest ranked entity
8. If the similarity score < threshold_2, consider the observed entity an observation of a new entity

Overall, resolution accuracy for matches higher than threshold_1 or lower than threshold_2 is 83.05%, using predefined thresholds.

In the experiment, we only go as deep as 2 levels in entity relationships. Fig 2 shows an example of subgraphs and their similarity scores. For the two entities, Alice and Elise, if we only look at the first level of connections, the two graph structures look very different. The cosine similarity of the two vectors is 0.4. However, if we consider both level 1 and 2 relationships, the cosine similarity increases as we observe that the two subgraph structure are indeed similar if we consider both levels.

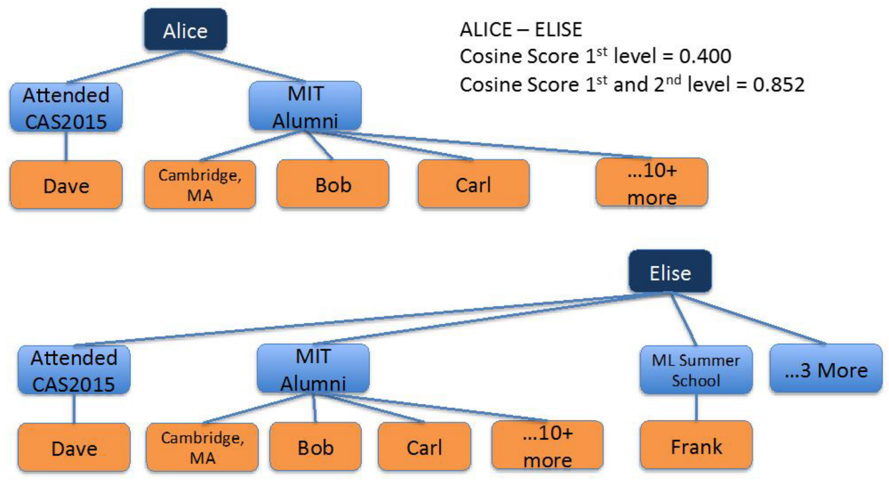

Fig 2 Example of subgraphs with their similarity scores

The two thresholds were not optimized to fit the experiment. Due to the small amount of data, the graphs generated from one document and even some of the graphs built from multiple documents are very small. In reality, graphs generated even from large dataset can be very sparse and fragmented.

The experiments presented here demonstrate the potential applicability of a vector space model to difficult entity resolution problems. In our previous and current research, we have studied semantic representations for node labels and graph structures. However, there are many other forms of data that can be associated with an entity, especially when dealing with reports from multiple sources and in various modalities. Additional issues include,

- **Representing numerical values.** While names and descriptions are typically represented in textual form, data, such as details of financial transactions and geographical proximity, are represented by continuous numerical values. Widdows and Cohen [13] discussed techniques to generate vectors that represent numeric quantities using orthogonal endpoint vectors and weighted interpolations.
- **Representing visual information.** Vector representation for image and video is a well-studied field. The vector encoding technique discussed in this paper is well suited to combine different vectors into a single representation. However, we will likely need to use real or complex vectors [13] instead of a binary vector form.

4. Conclusions and Future Research

In this paper, we presented a simple and versatile method to encode entity attributes and relationships with high dimensional random vectors. Our work is motivated by the sparseness and the inconsistencies in large multi-source datasets and the resulting complexity of subgraph matching. The main advantage of our technique is the capability of adding new as well as revoking outdated or erroneous attributes and relationships. Another advantage is its low computational complexity. Our previous experiments showed Random Indexing vectors capability to encode semantic meanings [1][2][3]. In this paper, we presented some preliminary result on using Random Indexing vectors with permutation to encode graph structures, i.e., entity relationships.

We have only started to explore the performance of high dimensional random vectors. One future research direction is determining how to best initialize and choose the sparseness of the vectors. When vectors are too sparse, multiplications will result in more zeros than desired. Since the operations are bit-wise, complexity is not an issue. However, sparseness is important to energy-efficient implementation in hardware.

Another focus of our future research is in enhancing the robustness of encoding the graph structures. While graph structures are more uniform in databases such as citeseer, etc., graph segments generated from reports by different analyst can differ significantly. We will extend our initial research in using permutation to better handle the variations in graph structures with techniques such as N-grams.

References

1. Paradis, P.D.,(2013) Guo, J.K., Moulton, J., Cameron, D., Kanerva, P., Finding Semantic Equivalence of Text Using Random Index Vectors, Complex Adaptive Systems 2013
2. Paradis, R.D.(2013a), Moulton, J., Guo, J.K., Kanerva, P., Using Random Index Vectors for Semantic Equivalence of Text using WordNet, poster presentation, IJCNN 2013
3. Guo, J.K., Van Brackle, D. and Hofmann, M.O., Enhancing A Rule-Based Event Coder with Semantic Vectors, Complex Adaptive Systems, 2014
4. Entity Resolution: Theory, Practice, and Open Challenges, Tutorial at AAAI-12 http://linqs.cs.umd.edu/projects//Tutorials/ER-AAAI12/Home.html
5. Ananthakrishna, R., Chaudhuri, S., and Ganti, V. 2002. Eliminating fuzzy duplicates in data warehouses. In The International Conference on Very Large Databases (VLDB). Hong Kong, China.
6. Bhattacharya, I. and Getoor, L. 2004. Iterative record linkage for cleaning and integration. In The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD). Paris, France.
7. Kalashnikov, D., Mehrotra, S., and Chen, Z. 2005. Exploiting relationships for domain- in-dependent data cleaning. In The SIAM International Conference on Data Mining (SIAM SDM). Newport Beach, USA.
8. Dong, X., Halevy, A., and Madhavan, J. 2005. Reference reconciliation in complex information spaces. In The ACM International Conference on Management of Data (SIGMOD). Baltimore, USA.
9. Wick, M., Singh, S., Pandya, H., and McCallum, A. (2013) "A Joint Model for Discovering and Linking Entities." Proceedings of the 2013 Workshop on Automated Knowledge Base Construction, pp. 67-72

10. Chen, Zhaoqi, Kalashnikov, Dmitri V., and Mehrotra, S. (2007) "Adaptive Graphical Ap-proach to Entity Resolution." Proceedings of the 7th ACM/IEEE-CS Joing Conference on Digital Libraries, pp 204-213.
11. Kanerva, P. (2009). Hyperdimensional Computing: An introduction to computing in distributed representation with high-dimensional random vectors. Cognitive Computation 1(2):139-159
12. Sahlgren, M., Holst, A. and Kanerva, P., Permutations as a Means to En-code Order in Word Space. Proceedings of the 30th Annual Meeting of the Cognitive Science So-ciety, Washington, DC. Pp.1300-1305. 2008
13. Widdows, D. and Cohen, T., Reasoning with Vectors: a Continuous Model for Fast Robust Inference, Logic Journal of the IGPL, 2015
14. Guo, J.K., Van Brackle, D., Lafoso, N. and Hofmann, M.O., Extracting Meaningful Entities from Human-Generated Tactical Reports, Complex Adaptive Systems 2015