CLASSIFIED

RECONSTRUCTING PRIMARY INFORMATION FROM SECONDARY INFORMATION

. . . . .

Drew J. Lipman, Ph.D.

HYPER GIANT

-Introduction

According to studies Data Scientists spend 60% of their time cleaning and organizing data. As a part of this, Data Scientists engage in what is known as "feature engineering." Feature engineering is the process of using existing, known, information or features to extract new information or features about the data. A classic example is to extract age or gender from salutation of passengers of the Titanic. This information, age and gender, is not, generally, included on the manifest of passengers, and yet it is very useful if you are trying to extract information about survival rates or make predictions of if a particular passenger survived.

There are generally two versions of feature engineering, one version is forward construction, for example, extracting age from date of birth, where the extracted features are directly derived from the original information. That is, all the information needed for the new feature is contained in the original feature. The other version of feature engineering, which is the focus of this discussion, is backwards construction. In backwards construction we are trying to recover features that were used to construct the given information, formally this is a type of inverse problem. In this case we do not have the information that was used originally to produce the feature we have. For example, gender and age of the Titanic passengers were used in order to choose the correct salutation used to address them. So, the inverse problem is to take the result, the salutation, and compute the inputs, the age and gender. In this framework, age and gender are a form of *primary information* while the salutation is a form of *secondary information*. This version of feature engineering uses secondary information, information *derived* from the primary features, in recovering the missing primary features.

While the framework of an inverse problem is very general, different problems produce different, and problem specific, difficulties. One tool that should be used for this process is Graph Databases or Knowledge Graphs. Often how data point features interact or how a single feature across a database interacts can be modeled using Graph Databases, and this presents a new set of tools that should be used to do feature engineering. In particular, we will model the Graph Databases using vertex and arc labeled directed graphs (digraphs). Using this framework, we have a *primary digraph* and a *secondary digraph*. The vertex weights and vertices of the secondary digraph are the same as the vertex weights and vertices of the primary digraph, the arcs, and their weights, of the secondary digraph are derived from the arcs, and their

weights, of the primary digraph. Using this language, the feature engineering problem is described as follows: given the secondary digraph and, potentially, parts of the primary digraph, extract useful information and features about the primary digraph that we did not previously have.

This framework presents many open questions including determining when it is feasible to extract a particular feature: arc weights, or the arcs themselves. Variations on this problem include: what is the minimal secondary digraph features we need to extract a particular feature? Equivalently, what features can be extracted given a set of secondary features? Another, related, problem is how dependent the resulting methods are on the choice of the Graph Database topologies? A final problem is if we can not extract complete information about a feature, can we extract useful partial information? These questions can be extended to include partial primary information.

-Example Problems

In this paper three example problems will be used to highlight various difficulties, and useful methods that can be used in recovering the primary graph:

The first example problem has, as primary information, the Parents Graph. That is, we are given a set of people, going back several generations, and we have an arc from a vertex x to another vertex y, which will be denoted as (x,y), if x is a parent of y. The derived information is the Cousin Graph, which is a graph with an edge (x,y) if x and y are first cousins. Equivalently, there is a path in the Parents Graph with arcs $(a,x),(b,a),(b,c),(c,y)$. That is there is an edge in the Cousin Graph if and only if x and y have a common grandparent, b.

The second example problem has, as primary information, the Maternal Graph. This is Similar to the Parent Graph, but now each vertex has at most one parent, their mother. The secondary information is the Maternal Cousin Graph where there are arcs $(a,x),(b,a),(b,c),(c,y)$ in the Maternal Graph. That is, the Maternal Cousin Graph has an edge (x,y) if and only if x and y have the same Maternal Grandmother, b.

The third example problem has, as primary information, the Handshake Graph a time-stamped graph of handshakes among a collection of people. Where the graph has an edge between x and y with weight t if x and y shook hands at time t. The secondary information is known as the Influence Digraph, which has an arc (x,y) if there is a path, of arbitrary length, connecting x to y where the weights, timestamps, are increasing for each edge. For simplicity we make the assumption that no person is shaking two hands at the same time. Note that the Influence Digraph can be used to model potential pathogen spread in a group of people. In particular, an arc (x,y) indicated that y could be infected by a pathogen originating at x. The lack of such an arc indicates that it is not possible to have spread a contact pathogen from x to y.

In these three examples we will assume we are missing a single piece of information. In particular we will assume we are only missing an arc from the Parent Graph, an arc from the Maternal Graph, and a timestamp from the Handshake Graph. In addition we will assume we have all the information from the Cousin Graph, the Maternal Cousin Graph and the Influence Digraph. Note that there is a graph topology implied by the

structure. Namely, we will assume that the Maternal Graph has the property that each vertex has at most one arc coming in.

-Background and Definitions

In Graph Theory there is a similar class of problems known as *graph reconstruction problems*. In this area there are long standing open conjectures about Graph Reconstruction. Starting with Ulam and Kelly's Reconstruction Conjecture, many questions and results about reconstructing, rebuilding, the original graph from some other data which was originally derived from the graph. For example, in Ulam's Reconstruction Problem, the secondary information is a collection of (unlabeled) graphs, called the deck, each produced by deleting a distinct vertex for each graph in the deck. The conjecture is that this secondary information uniquely determines the graph. That is, given two graphs, with more than three vertices, if the decks are the same then the graphs are the same. This conjecture can be stated as: given the secondary graphs, the unique, correct, primary graph can be constructed.

Since Ulam's Reconstruction Conjecture was first stated in 1957 (by Kelly) many partial results have been found, and similar questions have been posed using different derived information. Including Harary's variation for edge deletion where the edge-deck is the set of all subgraphs formed by deleting an edge. Similarly, some classes of graphs have been shown to be reconstructible, in particular Regular Graphs, Trees, Disconnected Graphs, Maximal Planar Graphs, and Outerplanar Graphs. In the setting of directed graphs there are infinite families of non-reconstructible graphs, in particular tournaments when they are not strongly connected.

While a wide range of questions involving recovering the original graph from secondary information have been posed they, for the most part, can not be applied to Graph Databases as they either only apply to unlabeled graphs, unweighted graphs, as in the original reconstruction conjecture, or are properties which generally do not appear in field conditions. This presents a large number of open problems based on reconstructing the primary Graph Database from secondary information, in particular reconstructing labeled, edge weighted, digraphs from a secondary digraph.

-Uniqueness and Probabilistic Approaches

As with many classical graph reconstruction problems: particular problems are often intractable. In particular, there are small toy examples of Parent Graphs which have the same derived Cousin Graph, and the Parent Graphs are the same after deletion of one directed edge. Consider the example Parent Graphs with vertex sets $\{a,b,c,d,e,f\}$. The first with arc set $\{(a,d),(b,c),(b,d),(c,e),(d,f)\}$, the second Parent Graph has arc set $\{(a,d),(a,c),(b,d),(c,e),(d,f)\}$. Note that the only difference is the first has arc (b,c) and the second has (a,c). The two Cousin Graphs both have the, unique, edge ef.

While this example shows that recovering the missing edge is, fundamentally, intractable we can still extract useful information from these relationships. In partic-

ular, if the cousin relationships are rich enough there is a finite number of potential arc locations for the missing arc from the Parent Graph. That is, we can locate a small number of potentials to query, which can be useful if the tests used are expensive. In particular, constructing the Cousin Graph of the Parent Graph with the deleted edge restricts the location of the missing edge to a subgraph of the parent graph related to the vertices with the missing relationship. Similarly, in the Handshake Graph, the missing timestamp can take on a, potentially unbounded, range of values. But if the Influence Digraph contains enough arcs which used the edge with the missing timestamp, then we can bound the interval of values.

An easy to understand example of the difficulty is in the Maternal Graph. Suppose we have as the true Maternal Graph, vertices $\{w,x,y,z\}$ and arcs $\{(w,x),(x,y),(y,z)\}$ but we are missing the arc (x,y). Note that the Maternal Cousin Graph is a graph with no arcs. So, even when we know what the Maternal Graph looks like, a directed path, without information outside of the model, say date of birth, we can not tell if the missing arc is (x,y) or (z,w). As both arcs produce isomorphic Maternal Graphs, and Maternal Cousin Graphs.

Alternatively consider, if we have, as the true Maternal Graph, vertices $\{v,w,x,y,z\}$ and arcs $\{(v,w),(w,x),(v,y),(y,z)\}$ and missing arc (v,w). In this setting the Maternal Cousin Graph has one edge, (x,z). As this is a Maternal Graph we know that each vertex has at most one arc coming in, which means we know the missing arc is (v,w) or is (x,v) as there are no other options. However, only (v,w) will produce the edge (x,z) in the Maternal Cousin Graph.

Given that these problems do not have, in general, a single solution this presents the question of when are the solutions unique, and if they are not, can we extract useful information anyway? For example, in the Maternal Graph, if the Maternal Cousin Graph has a rich enough structure, the possible locations of the missing arc is limited. In particular, if we construct the Maternal Cousin Graph of the Maternal Graph with the missing arc, and we do not have an edge (x,y) in the new Maternal Cousin Graph that was in the original Maternal Cousin Graph, we know that we are missing a particular arc needed to produce this cousin relationship. In particular we are missing an arc modeling the mother of x, the grandmother of x, mother, or grandmother of y. Given we are missing exactly one arc, we know x or y will have their grandmother known, and thus we know exactly what arc we are missing in relation to x and y. Thus, for example, if we are missing the arc modeling the mother of x we know it has to be a daughter of the grandmother of y but not y's mother, i.e. an Aunt of y. In this case, the Secondary Graph provides enough information that in conjunction with the known topology of the Maternal Graph we can limit the missing arc to a relatively small set of possibilities. So, in this example, we have an almost unique solution because the Maternal Graph has a nice topology and the Maternal Cousin Graph has a rich enough structure.

Similarly, if we are missing the timestamp from an edge (x,y) in the Handshake Graph, but we know that it is used on a path from x to a vertex w, that is we have a path which uses (x,y) and then has a path from y to w, we can attach an upper bound to the missing timestamp. If we list all the paths from y to w where the timestamps in-

crease, so they produce an arc in the Influence Digraph, the timestamp of (x,y) must be smaller than the largest initial timestamp. That is, if we have a unique path from y to w and the first edge has timestamp t then we know that the only way the path from x to w can use the edge (x,y) is if the missing timestamp is smaller than t. Similarly, we can bound the missing timestamp below if it is the last edge on a path with increasing timestamps. While this does not, in general, provide an exact value, it does provide an interval in which the handshake occurred. If we have additional information, like all timestamps are stored as integer values, this may, in fact, be enough to extract a specific value.

-Conclusions and Future Directions

From this discussion we can see that this method of approaching feature engineering offers many potentials for extracting new features, determine when a unique answer can be extracted, formalizes the information we need to extract features, and provide meaningful potential answers when the answer is not unique. Future directions of inquiry include:

- A more robust understanding of when we can extract a unique solution,
- Study the relationship between how the secondary information is defined and the size and location of potential answers, and how this relates to multiple missing features,
- Study when there are multiple sources of secondary information,
- Determine a metric that measures how robust secondary information is in relation to recovering missing Primary information.