Machine Learning Engineer Nanodegree

Capstone Proposal: Understanding 'Things' using Semantic Graph Classification

Rahul Parundekar
January 27th, 2017

Proposal

Domain Background

The world around us contains different types of things (e.g. people, places, objects, ideas, etc.). Predominantly, these things are defined by their attributes like shape, color, etc. These things are also defined by the "roles" that they play in their relationships with other things. For example, Washington D.C. is a place and U.S.A is a country. But they have a relationship of Washington D.C. being the capital of USA, which adds extra meaning to Washington D.C. This same role is played by Paris for France.

The field of Knowledge Representation and Reasoning within Artificial Intelligence deals with representing these things, types, attributes and relationships using symbols and enabling the agent to reason about them. As it happens, a convergence has come about in this field of knowledge representation from the Databases domain - Graph databases can use graphs with nodes and edges to represent data much similar to the knowledge graphs.

Many domains use semantic grahps to represent their information because nodes, properties and edges of graphs are very well suited to describe the attributes and relationships of things in the domain.

For example:

1. Spoken systems - the output of Natural Language Processing is a parse tree.
2. Social networks are graphs.
3. High level semantic information in images are graphs of arrangements of things.
4. The arrangement of objects on the road for autonomous driving is a graph.
5. A user's browsing pattern of products, usage graphs, etc. is a graph (e.g. browing products, plan of actions, etc.).

In the classic sense, Machine Learning focuses on specific kinds of understanding - classification, clustering, regression, etc. The algorightims in these deal with

feature vectors (e.g. the features used for classification, etc.) and are aimed at essentially discriminating between different types of input to produce some output. To make decisions based on the state of the world an A.I. Agent can read from the world using sensors etc., can easily perform a classification task once it learns the relation between the data to its output decision. For the most part, the feature vectors used in such cases as input encode the attributes of the things, BUT not necessarily the relationships between things. And while the designer of the inputs and outputs of the algorithms may manually craft features to represent some of these relationships, the Agent has no automatic way of comprehending and using these relationships.

Can we use machine learning to make Agents better understand Things, including their attributes AND their relationships? If we are able to inspect the attributes and relationships of the things together and infer their roles, find its types, etc. our agent can act on those. If an Agent is able to classify things by understanding its semantic relationships, we could in the future generalize it to an Agent that can act on the meaning of the things.

Existing research in this domain:

- Ontology alignment is a field of study that researches on understanding the classes of things by aligning types from one data source to the types defined in another source to increase interoperability. In my previous work on aligning ontologies, we employed a brute force method to discover new classes to describe things using 'Restriction Classes' defined by restricting the properties to their values and creating a set of instances to match that restriction. This and other instance based methods can be used to understand things by either creating new class definitions or aligning the definition of things to other classes.
- Graph based classification methods have been used in toxicology detection.
- Text categorization using graph classification is also investigated.

We pick one of these problems as a candidate for exploring the use of Machine Learning to understand semantic data.

Problem Statement

Given the semantic data about things (i.e. their attributes and relationships), can we identify their types (e.g. hierarchy of classes) or categories (e.g. roles it plays) interest?

For example, if you look at examples of categories in DBpedia, Achilies has been put into the categories - demigods, people of trojan war, characters in Illead, etc. What makes him part of those categories? Can we learn the definitions of these based on the attributes and relationships of Achilies?

Datasets and Inputs

An Ontology is one formalism of to represent semantic data. It specifies the facts about the things' attributes and their relationships that the Agent believes to be true. DBpedia is one such Ontology whose lofty goal is to create a knowledge repository of general knowledge about the world such that Agents can have a grounding of the popular concepts and entities and their relationships. DBpedia datasets contains structured information extracted out of Wikipedia (e.g. page links, categories, infoboxes, etc.) (See DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia for details of extraction).

The semantic data in DBpedia can be represented as a graph of nodes and edges. In this case, the things are the nodes and the links/relationships between the things are its edges. For determining the types and categories of a Thing, we start with the node for that thing in the center and look at its neighborhood (e.g. things 1 or 2 hop away) subgraph. We model this as a classification problem.

The input is a list of [different subgraphs centered around things] and the class values are a vector of [types of those things]. To predict the categories of the things, the input is a list of [different subgraphs centered around things] and the class values are a vector of [categories of those things]. DBPedia contains this ground truth information. We can split the data into training, validation & testing set using cross validation, etc. strategies. This data is our ground truth with high fidelity as it has been mannually created by human wikipedia contributors.

Solution Statement

As described in the Graph classification paper, we will use graph kernels. For example, we can use a random walk technique on our input of subgraphs to extract features that can be readily used with our machine learning algorithms. We can then use classification algorithms to identify the types or categories of the instances in our dataset.

By usign this approach we convert graph classification into the classic classification problem and can use differnt classification & algorithms like SVMs, Logistic Regression, Deep learning, etc. to correctly classify the instances

Benchmark Model

Since there is not much research in this field, we will set our benchmark as the classes that can be identified using the brute force approach of ontology alignment in creating restriction classes (i.e. sets of things formed by restricting properties to values) and matching them to the Types & categories already annotated in dbpedia.

Evaluation Metrics

We will compare both the accuracy of finding class definitions as well as as the quality of the definitions by investigating the sets of instances in defined in the class.

For the first part, the ontology alignment approach will align the classes (types and roles) with the list of instances. Our approach will also classify the instances. We can then compute the precision, recall and f1 score of these approaches by looking at the predicted values and the ground truth.

For the second part, we qualitatively compare the set of instances in the classes in the ground truth with the false negative and false positive instances that we predict. This gives us insight into how different algorithms work.

Project Design

The workflow of the capstone approach is as follows:

a) We first extract the data from DBpedia dataset and create the graph classification data - The data from DBpedia is in Semantic Web formats (e.g. RDF) we need to translate that to a simpler graph format that we can store in a database. We will use the "Type", "InfoBox properties" and "Categories" datasets. For example Neo4J is a graph that can be used to store the whole data about the instances as a single graph.

b) We also track the identifiers of the things that have types and categories as our ground truth. We can then split this list of instances into training, testing and validation datasets.

c) Before we create the data for learning, we need to do some pre-processing by analying the nature of the data: for example, properties with numeric values might need to be discretized, we need to understand the coverage of the classes (as the counts will not be equal), some properties might not be useful for classification (e.g. metadata like author name, etc.)

d) We then model our classification problem as a multi-label classification and make the data ready for machine learning- To create the data that we can use for learning, we iterate over the list of things to create the features using graph kernel methods. We then have a typical classification data - features (extracted from e.g. random walks) and classes - the types or the categories.

e) First we will investigate type classification. Then, we will investigate category classification. The target multi-lable vector can represent 1 or 0 depending whether the thing belongs to that type or category (one class can belong to multiple types e.g. all classes in its hierarchy of types)

f) While doing this, we can vary 3 things:

- We can vary the graph kernal methods - e.g. with random walks, we can change the length of the random walk, the probabilities with which we take the node or edge in the walk, the neighborhood, etc.
- We can vary the length of the walks to extract as features.
- We can also vary the classification method - e.g. we can use deep learning, one-vs-rest clasifiers, etc.

We can create a comparison of the different combinations used and finally pick one approach that looks best suited for this type of classification.

g) We then use this final approach to do both type and category prediction and calculate the metrics.

h) For our benchmark model, we use the restriction classes approach to create our brute force benchmark.

i) Finally we compare the results of our classification approach and the benchmark, using quantitative and qualitative analysis as described in the Metrics section.