Automatic Annotation in Multirelational Information Networks

Richard Barber & Austin Gibbons

December 2011

1 Problem Background

Many networks are completely encapsulated using a single node type and a single edge type. Often a more complicated model composed of multiple distinct node and edge types can be constructed to create a more informative network [2]. We call the former homogeneous networks and the latter heterogeneous. The ability to homogenize networks varies wildly, dependent on the network under analysis and the problem being solved. We present a class of networks - multirelational information networks - where the heterogeneous structure is necessary for performing node classification and detecting missing information in the network.

A multirelational information network is a network G with nodes V that map to real world objects and concepts which we call entities, and with edges E which represent relations between these entities. We use nodes and edges interchangeably with entities and relations. For the multirelational networks we will consider, there exist a true vertex labeling function $l_v : V \rightarrow 2^{\Sigma} \backslash \{\}$, where Σ is an alphabet of node types and an observed vertex labeling function $\hat{l}_v : V \rightarrow 2^{\Sigma}$. Informally, entity types will exist in a hierarchy and we require that the labeling only map a vertex to a set of types $\mathcal{T}$ such that for any pair $(t_{i,}t_j)$, $i \neq j$, t_i is a descendant of t_j in the hierarchy or vice versa. Similarly, there exists labeling functions $\hat{l}_e$ and l_e for the edges, but we will not be exploring the existence of an edge hierarchy in this work.

Our goal is to learn the true vertex labeling function from the observed multirelational information network and labels. Specifically, given network $G = (V, E)$, hierarchy $\mathcal{T}$, and observed labeling function$\hat{l}_v$, we will learn the true labeling function l_v.

Additionally, we predict missing data fields in nodes that have incomplete relations. Namely, the edgeset E that we observe is not the true and complete set of relations that exist in the world. For example, the Mustang entity and the Camaro entity have many features in common, but only the Camaro has the "model years" relation. We would like to automatically recommend relations that may exist for a node based on our estimation of the true set of relations E^*. More precisely, given $G = (V, E)$, we would like to provide $A : V -> E$ s.t. A takes a node v and outputs a subset of edges that it should be incident upon v. The heterogeneous structure of the network is vital to this task, as we are essentially determining the missing pieces of the heterogeneous structure.

Being able to discover entity types and missing relations will provide a method for database maintainers to discover and correct missing information in their entities. We can directly present our algorithm to database providers to automatically discover incompleteness in their data. We will provide a specific example of usefulness in 2.1.

1.1 Related Work

There is a large body of work being done by researchers in the probabilistic graphical models and databases communities on statistical relational learning (SRL) which uses the machinery of logic, probabilistic grammars and graphical models to address learning in relational data [3].

A number of tasks including classification are defined in the SRL formalism, but it should be noted that the SRL program is ambitious and seeks to do much more than answer simple binary or even multi-class classification queries. Indeed, the inclusion of logics and grammars alongside probabilistic learning tools is calculated to make a much richer set of relational queries answerable within the framework.

For our classification task, this broad set of tools presents a drawback of many of the approaches to classification in this field lie in the fact that the relational data is modeled with complex generative structures [4] which make exact and even approximate inference intractable on very large data sets.

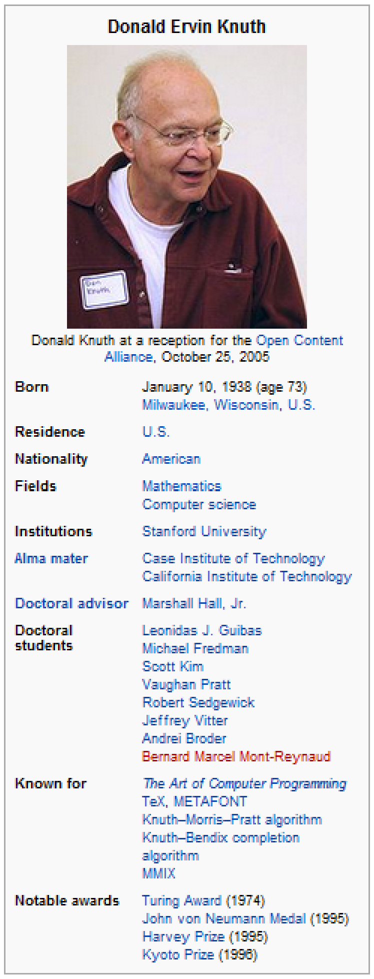
As we describe below we will use discriminative learning on graph and label features only for the type classification task.

Similar work has also been done in feature extraction [5], which takes the opposite approach. This work assumes that there is a collection of features (similar to relations) of which a subset are representative of the data and the rest of which are noise. This class of algorithms moves to correctly eliminate the noise and discover the underlying representation which distinguishes nodes from each other. The techniques that are taken to solve this class of problems are similar to the methodology we present, but applied under a different light.

2 Data

2.1 Data Sets

We use the high quality multirelational dataset DBpedia which is based on the structured information contained in Wikipedia's Infoboxes



Donald Ervin Knuth

Donald Knuth at a reception for the Open Content Alliance, October 25, 2005

Born	January 10, 1938 (age 73) Milwaukee, Wisconsin, U.S.
Residence	U.S.
Nationality	American
Fields	Mathematics Computer science
Institutions	Stanford University
Alma mater	Case Institute of Technology California Institute of Technology
Doctoral advisor	Marshall Hall, Jr.
Doctoral students	Leonidas J. Guibas Michael Fredman Scott Kim Vaughan Pratt Robert Sedgewick Jeffrey Vitter Andrei Broder Bernard Marcel Mont-Reynaud
Known for	*The Art of Computer Programming* TeX, METAFONT Knuth–Morris–Pratt algorithm Knuth–Bendix completion algorithm MMIX
Notable awards	Turing Award (1974) John von Neumann Medal (1995) Harvey Prize (1995) Kyoto Prize (1996)

Figure 1: Infobox for Donald Knuth

DBpedia [1] is a dataset which is post-processed versions of Wikipedia Infoboxes like the one above. This dataset makes attempts to clean and standardize the Infobox data, mostly by crowd-sourcing but with some automatic mechanisms.

DBpedia includes a type hierarchy of 237 types and gives a set of type labels for most nodes in the network. DBpedia has no formal specification of the set of attributes a type may have and much less of a system for incorporating constraints and annotations of the relational information. This makes it a good candidate dataset to explore, so we concentrate our efforts here.

The dataset suffers from some amount of incompleteness, incorrectness, and inconsistency throughout. This incompleteness motivates our efforts. If we can generate a highly accurate true labeling function l_v, we can provide it to the DBpedia caretakers to assist in their cataloging efforts.

We concentrate our efforts on DBpedia, but the model we present is general and can be extended to other datasets.

2.2 Network Structure

DBpedia's data is available in raw form in three files: the ontology file, the properties file, and the types file.

The ontology file is an XML file describing the hierarchy of DBpedia types. The root node is Thing since all entities in the dataset are assumed to be Things. We parse the XML and store the data as a tree structure on which a number of queries can be efficiently performed.

The properties file is a list of relations represented as triples (s, p, o) where s is an infobox entity that we call the subject, p is the name of a relation we call a predicate, and o is either a literal value or another entity which we call the object of the relation.

An example is given below

```
<http://dbpedia.org/resource/Aristotle>
<http://dbpedia.org/ontology/influenced>
<http://dbpedia.org/resource/Ptolemy> .
```

We represent this data as an actual graph. For the above example, we create a node corresponding to Aristotle and Ptolemy in our graph and we add an edge with label "influenced" between these two nodes.

The types file is a list of triples (e, n, t) where e is the name of an entity, t is a type of the entity and n is a constant we ignore.

An example is given below

```
<http://dbpedia.org/resource/Autism>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://dbpedia.org/ontology/Disease> .
<http://dbpedia.org/resource/Autism>
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://www.w3.org/2002/07/owl#Thing> .
```

This says that the DBpedia labeling function assigns labels Disease and Thing to the entity Autism. Of course, we can't be certain that the true, unobserved labeling function labels Autism in exactly the same way, but we will try to use the signal of the observed labeling function to learn the true labeling function, <add reference>. We parse this file and store the type information in dictionaries.

We are building all of our work on top of SNAP and using many of its capabilities for network manipulation.

2.3 Features

We generate three sets of features, and perform analysis both independently and conjointly to determine which features offer the most discrimination. We have node local statistics corresponding to canonical node data: in-degree, out-degree, clustering coefficient, and other well practiced statistics. We generate edge type features as a vector of frequencies of the types of edges incident on a node. Recalling our example from our example relations triplet, Aristotle would count "influenced" as an outgoing edge, and Ptolemy would count "influenced" as an incoming edge. Finally, we use the frequencies of adjacent entities relative to a node. For instance, from the triplet above we know Aristotle has at least one neighbor of type Mathematician, Person, and Thing (namely Ptolemy).

3 Method

3.1 Node Classification

Having constructed heterogeneous networks from the DBpedia data, generating statistics from that network, and creating features out of those statistics, we now build a learning model. For each node type, we sample randomly $m + p$ entities of that type and assign them positive labels, and $m + p$ entities from the rest of the database, and assign them negative labels. For each of our feature sets, we train a learning algorithm over the labeled features.

As an example, DBpedia has a Comedian type corresponding to entities who are comedians. We pull $m + p$ comedians and assign them label 1, and $m+p$ non-comedians and assign them label 0. We then take m samples from each of the Comedians and non-Comedians and train a model, then test our model on the remaining $2p$ samples. This demonstrates that for a specific type, we can label entities with high accuracy given noisy data.

We now need to address the general problem of given a node with no associated entity, can we correctly classify it? In order to address this question, we build a model for every type as described above. Then, we recursively descend through the hierarchy and test our feature against each type at the current level. We pick the type with the most confident label, and then move to the next level, stopping when we reach a leaf.

The real challenge to evaluating these types is the similarity in types at the lower levels of the hierarchy. It is easier to distinguish between a Person and a Place as they have very different relations, but it is more challenging to differentiate between a Comedian, an Actor, and a Musician, as they share many common relations, and often an individual is two or even all three of these types.

3.2 Relation Annotation

We explored several methods for providing candidate relations for an entity. As a baseline we first implemented random selection. This naturally did not perform well. We then tried simply returning the attributes in order of frequency for the corresponding sub-type in the hierarchy. This was a significant improvement over randomness and demonstrated the usefulness of our approach, but left room for greater success.

We then implemented the following mechanism. Consider every pair of relations belonging to a node, and offer the relation which best "completes" a triplet. From our existing data we extract the relative co-occurrence of relation triplets which all belong to the same entity. We then store this information in a database

$$<< r_1, r_2, r_3 >, \ \textit{count} >$$

then our method for recommending relations becomes matching the relation which creates the largest count:

```
for x,y relations of node n
        max_r
        for candidate relations r :
                max_r := max(getCount(x, y, max_r), getCount(x, y, r))
        end
emit(max_r)
```

Thus we analyzed the success over our three algorithms when given a node n and its relations $\mathcal{T}(n)$

- Random: predict a new relation t randomly such that $t \notin \mathcal{T}(n)$
- Frequency: $argmax_{t \notin \mathcal{T}(n)} \sum_{v \in V} I(t \in \mathcal{T}(v))$
- Bayes-2: $argmax_{t \notin \mathcal{T}(n)} \prod_{t_i, t_j \in \mathcal{T}(n)} P(t|t_i, t_j)$

Note that our Bayes-2 is an approximation to the more general prediction based on on probability $P(t|t_1, t_2, ..., t_n)\ \forall t \in \mathcal{T}(n)$. Performing this calculation over every relation is much more computationally expensive, and we will show in section 4 that Bayes-2 already performs exceptionally well.

We can test the ability to predict features by taking an existing node and removing a small subset of its relations, and observe whether or algorithm is able to correctly recover the relations we deleted.

4 Results

We were able to successfully detect a high percentage of labels as well as missing annotations. We use the precision, recall, and accuracy at the top five labelings to classify the effectiveness of our algorithms. These feedback measures offer insight into the algorithms ability to both predict correct labelings and to avoid incorrect predictions. We present the following data which demonstrates the success of labeling using all features over the DBpedia database at different depths in the type hierarchy. We present these features because they were the most effective. Note that if a mistake is made in a higher level, then that node will necessarily be incorrect at the next level.

Depth	1	2	3	4
Precision	0.781250	0.8	0.625	0.6667
Recall	0.949367	0.780488	0.416667	0.3333
Accuracy	0.75	0.653061	0.3333	0.5

We can quickly see that using our feature information can provide insight into the label of a node. This gives strength to our claim that the heterogeneous structure of the network plays a vital role in making these kinds of predictions. We can conclude that our strategy is a useful tool for completing dataset labelings such as labeling from Wikipedia's ontology. We can conclude that it is possible for a human to fill in missing information in structured data much more efficiently by using the labeling hints we provide.

We can even observe the success for individual type and use this information to target specific types and also perform domain analysis. For example, observe the precision, recall, and accuracy for the "Criminal" type:

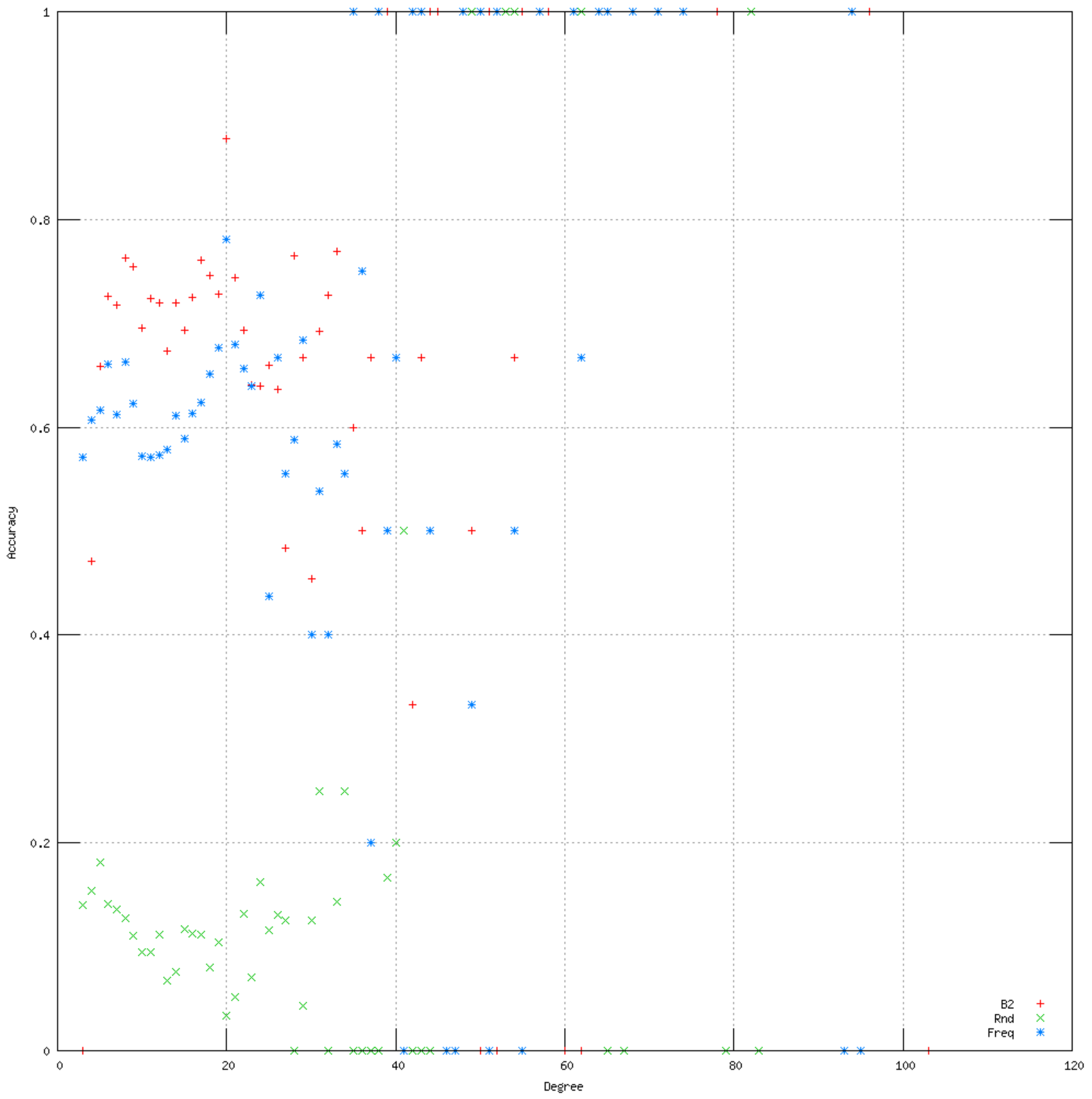
Features	Node Type	Node Statistics	Edge Type	All
Precision	0.558824	0.558824	0.933333	0.964912
Recall	1	1	0.982456	0.964912
Accuracy	0.558824	0.558824	0.95098	0.96708

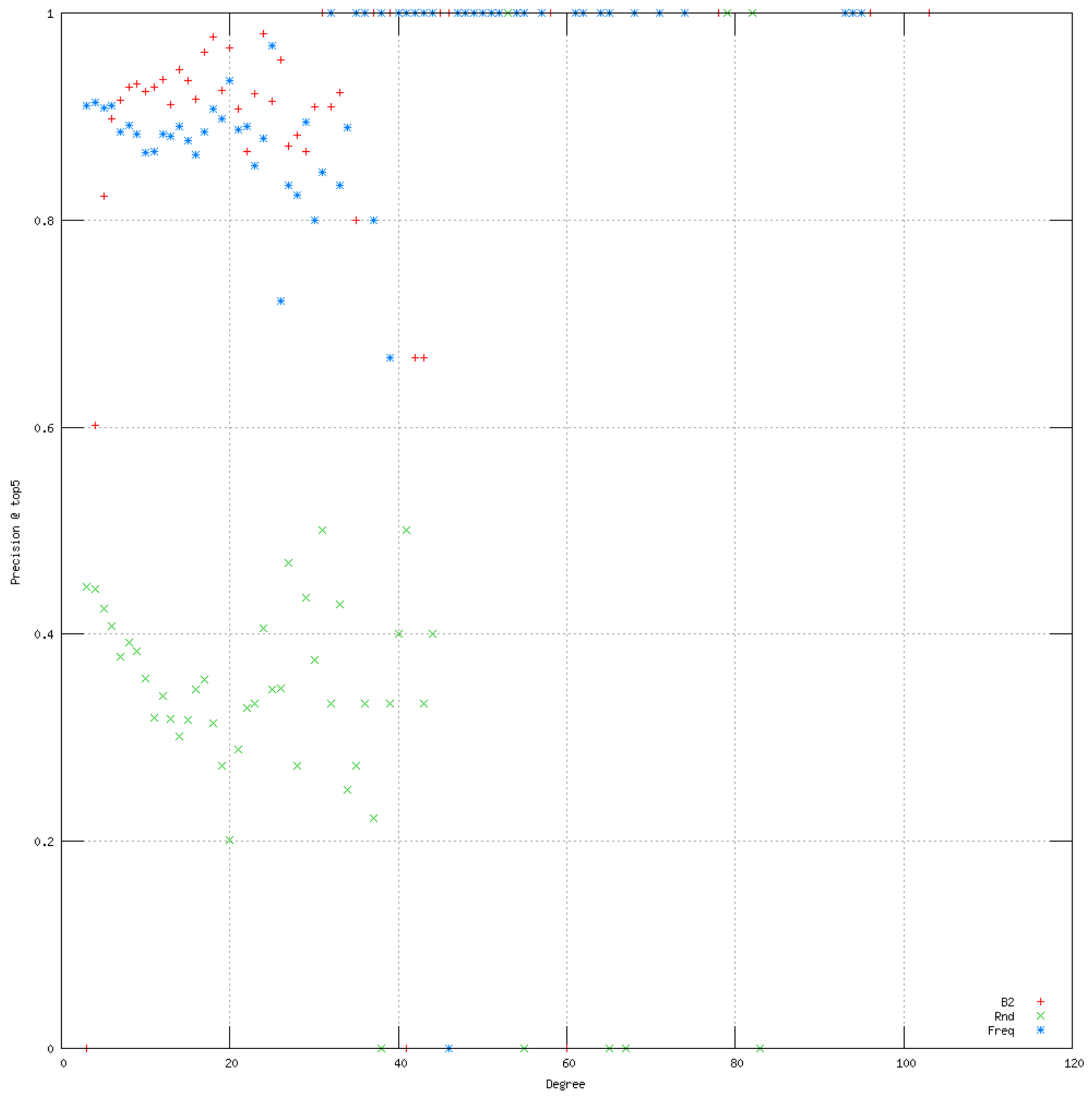
It may be useful for us to know that we can predict Criminal with perfect recall, but that there are many non-Criminals whose node information is very similar.

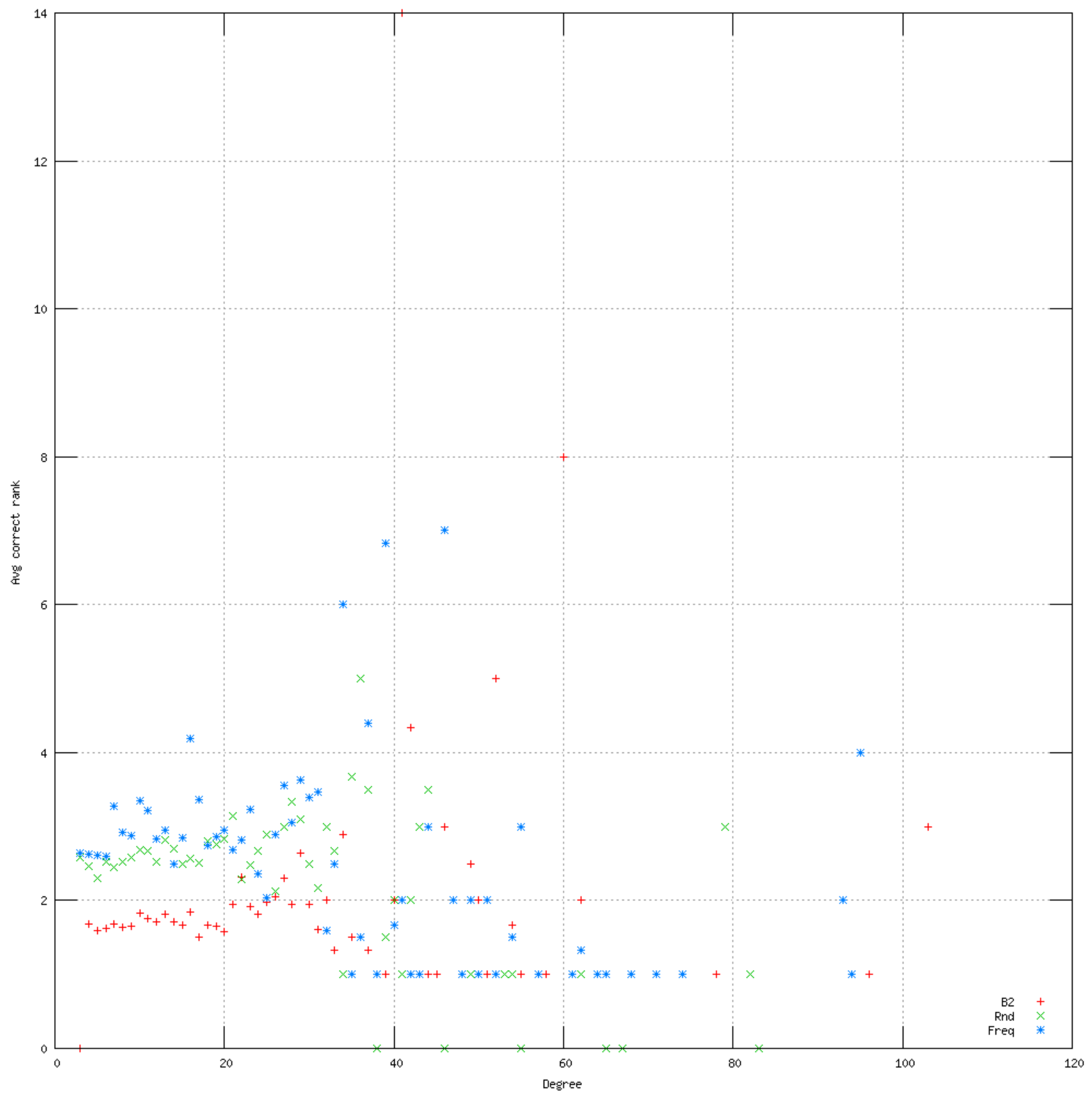
For the full listing of type labeling please see the stats.txt file in supplemental materials.

Additionally, our implementation runs in a reasonable wall-clock time. We can predict thousands of labels in under ten seconds.

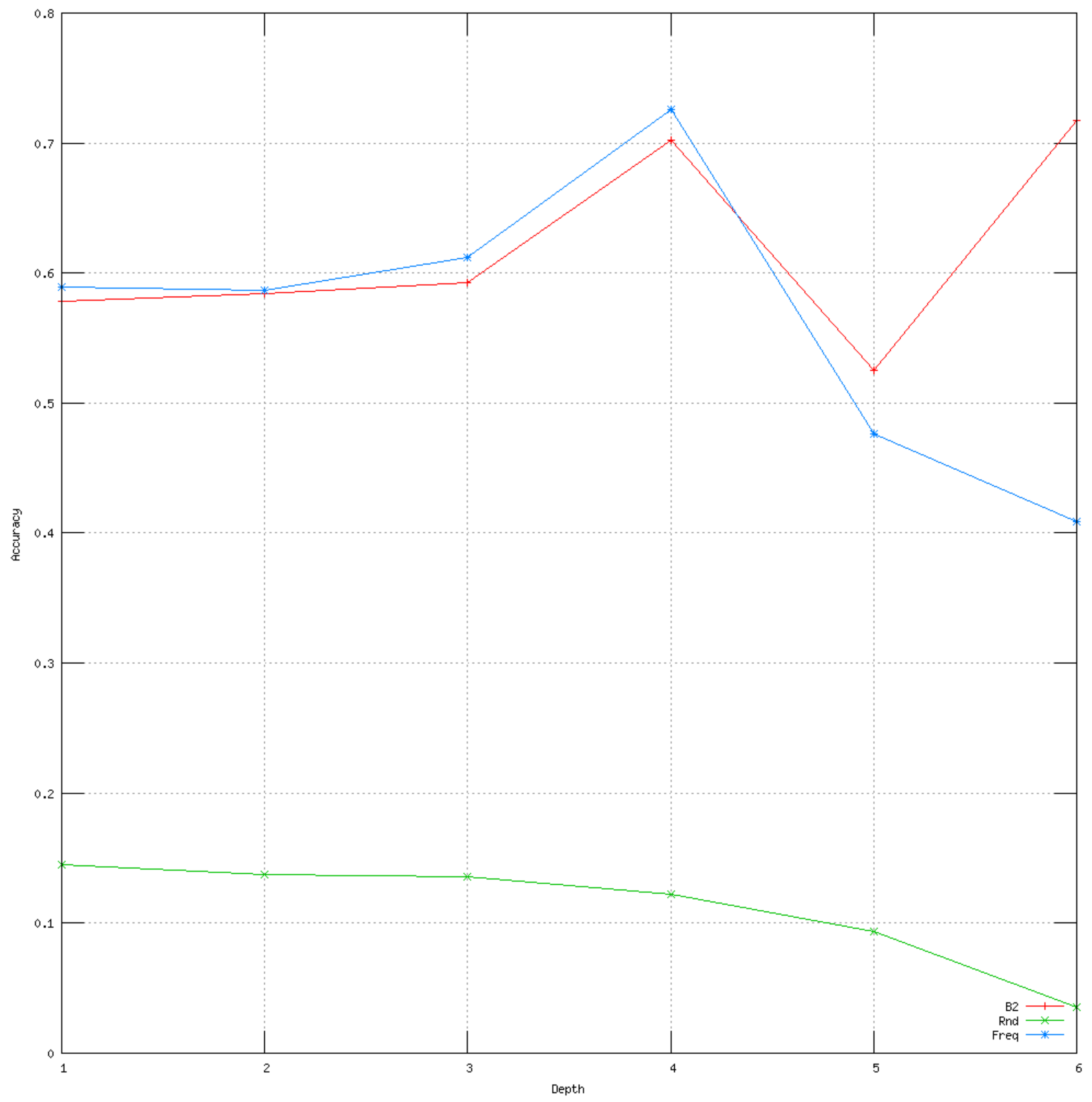
We analyze the success of our annotation mechanism over each degree count. Recall that we are testing the success by removing a random relation and observing where it is predicted. We present the accuracy and precision at the top five results, and additionally the average rank of the relation, which is its position in the prediction scheme sorted by probability of membership. The x-axis is predictions for nodes with x degrees, and the y-access represents the accuracy, prediction, and average rank respectively. We were hoping to observe an interesting trend based on the degree count, but we have mostly observed the general level of success and the sparsity of the graph. Many of the nodes with degrees above 40 have a unique degree count, and thus we see either 0 or 1 accuracy and prediction whether our prediction was correct or incorrect. You can see that both Frequency and Bayes-2 perform well even for nodes with large degrees. From these graphs we conclude that both Frequency and Bayes-2 are effective systems for automatic annotation, and Bayes-2 is especially effective at producing the correct result at a very high rank (often the first or second prediction)

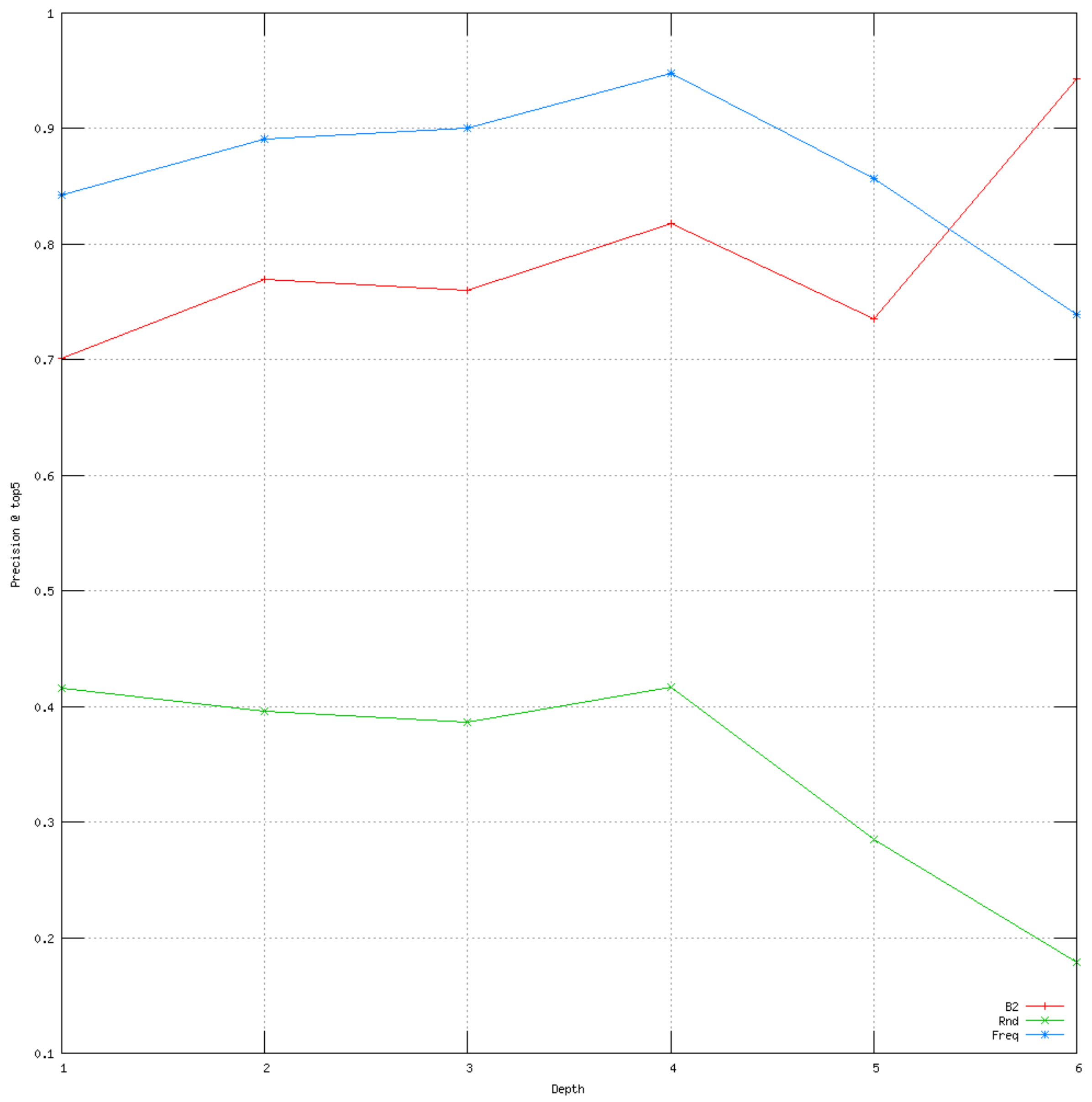


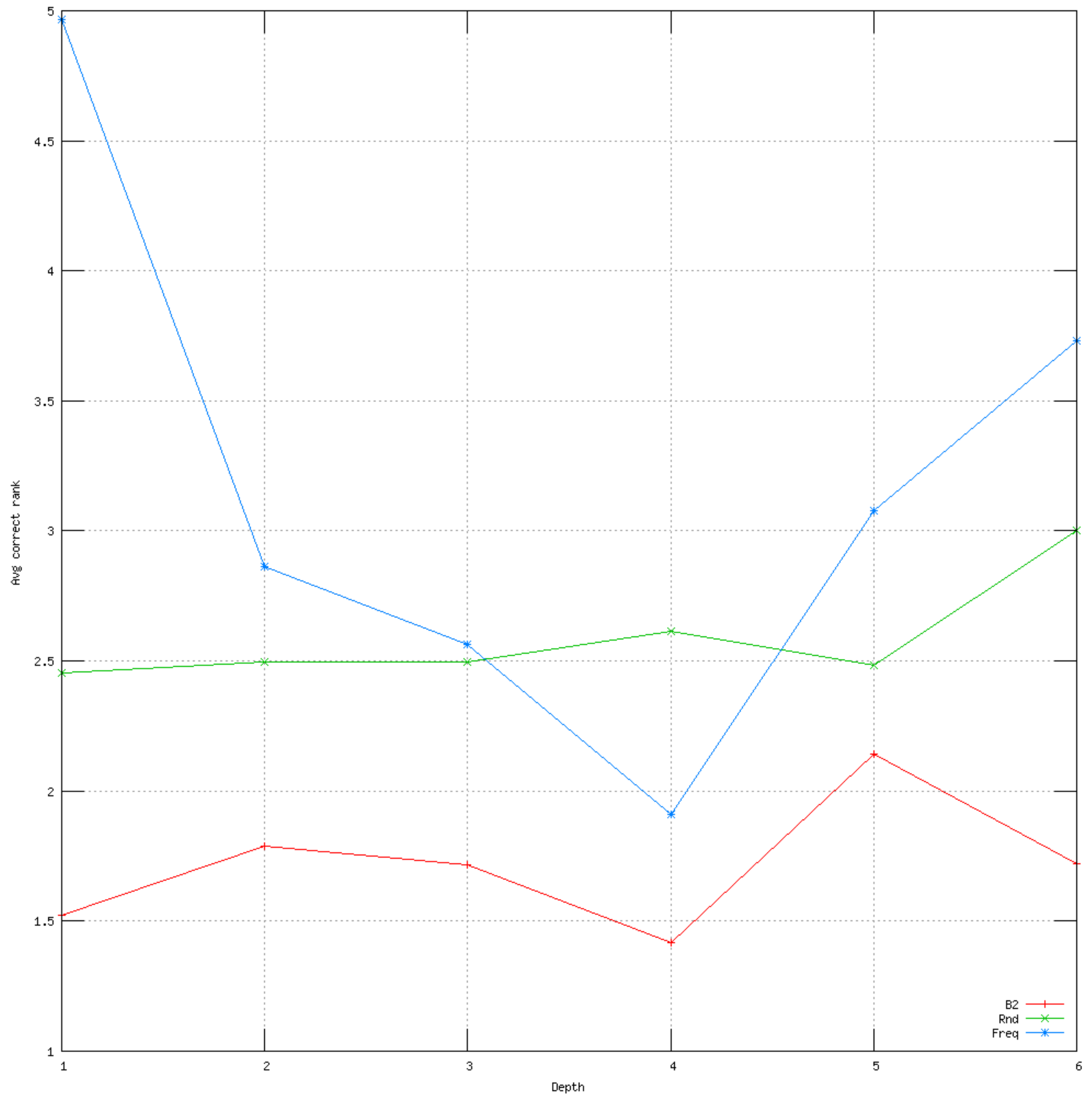

Precision @ top5
Degree
B2 +
Rnd ×
Freq ✱
0
0.2
0.4
0.6
0.8
1
0
20
40
60
80
100
120



We then conducted a similar analysis over the "depth" of the entity in the type hierarchy. For example, if an entity has type {Thing, Person, Comedian, Mime} they have a depth of four. We can observe that once again Frequency and Bayes-2 present a significant improvement over random selection, and that Bayes-2 begins to work slightly better as the depth increases. These graphs provide a strong demonstration of the ability to recover missing information in multirelational networks.


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Accuracy
1
2
3
4
5
6
Depth
B2
Rnd
Freq


Precision @ top5
Depth
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
2
3
4
5
6
B2
Rnd
Freq



For the full list of results of prediction for each type over our different heuristic functions, please see “missing data” directory in supplemental materials.

5 Future Work

The most immediate step we strive to take is to present out findings to the DBpedia caretakers. Then in addition to applying this to the DBpedia model, we can apply it to other networks (such as Freebase, which is also representative of the Wikipedia dataset and expressible as a multirelational network) and observe how it performs over that data.

Another interesting application of entity labeling can be determining which labels are redundant. For a single sub-type, we could remove the label for each entity with that node. Then by running each entity through our node labeling algorithm, we can find the labeling which the majority of entities are classified (which is different from their original label, as it no longer exists in the database). We can use this to eliminate redundancy in the network, and to detect type paths which are isomorphic.

Part of the difficulty in pursuing automatic annotation is computational expense. We could expand out "triple completion" to higher degrees (quadruple, quintuple, and so forth) in the pursuit of better results. Unfortunately, it quickly becomes computationally infeasible to do so. We instead would like to pursue approximation techniques for detecting the mutual information across the relations within an entity and make predictions under that basis. It is fair to do this, as the data we observe is an approximation of the true data already, and thus we could potentially find an effective approximation strategy.

The next step for analysis is to understand why some predictions occur and others do not. Can we detect the features which are indicative of a successful entity labeling or a successful relation annotation? If we can successfully address these questions, we may be able to expose a "ground truth" in our database, such as the level of similarity between relations.

References

1. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. Web Semantics: Science, Services, and Agents on the WorldWide Web. September 2009.

2. Jiawei Han, Yizhou Sun, Xifeng Yan, and Philip S. Yu. Mining heterogeneous information networks. Tutorial KDD 2010.

3. Eds. Lise Getoor and Ben Taskar. Introduction to statistical relational learning. MIT Press, 2007.

4. Hassan Khosravi and Bahareh Bina. A survey on statistical relational learning. Advances in Artificial Intelligence, 201

5. Daphne Koller, Mehran Sahami. Toward Optimal Feature Selection. In 13th International Confer- ence on Machine Learning, pages 284{292, July 1996.