AN EMPIRICAL COMPARISON OF LABEL PREDICTION ALGORITHMS ON AUTOMATICALLY INFERRED NETWORKS

Omar Ali[1], Giovanni Zappella[2], Tijl De Bie[1] and Nello Cristianini[1]
[1]*Intelligent Systems Laboratory, Bristol University, Bristol, United Kingdom*
[2]*Dipartimento di Matematica 'F. Enriques', Università Degli Studi di Milano, Milan, Italy*

Keywords: Nodes classification, Social networks, Data mining, Machine learning.

Abstract: The task of predicting the label of a network node, based on the labels of the remaining nodes, is an area of growing interest in machine learning, as various types of data are naturally represented as nodes in a graph. As an increasing number of methods and approaches are proposed to solve this task, the problem of comparing their performance becomes of key importance. In this paper we present an extensive experimental comparison of 15 different methods, on 15 different labelled-networks, as well as releasing all datasets and source code. In addition, we release a further set of networks that were not used in this study (as not all benchmarked methods could manage very large datasets). Besides the release of data, protocols and algorithms, the key contribution of this study is that in each of the 225 combinations we tested, the best performance—both in accuracy and running time—was achieved by the same algorithm: Online Majority Vote. This is also one of the simplest methods to implement.

1 INTRODUCTION

One of the important trends in data mining and pattern analysis is the growing importance of datasets where data items are regarded as nodes in a network. This is due to the increasing focus on domains where this data arises naturally, such as social networking sites or gene regulation analysis, and to the growing size of other datasets where vector-based representations can be very expensive to use.

As several approaches have been proposed to solve this task, it is important to develop a rigorous benchmarking protocol, that includes standardised datasets and open-source code, so that researchers can pursue the most promising avenues and practitioners can have objective information about the merits of each approach.

In this paper we present an empirical study of several state-of-the-art methods for label prediction, applied to our own, automatically inferred and labelled social networks. Our experiments test 15 variants of four algorithms, on three different networks, each having five labellings, for a total of 225 different combinations.

The algorithm-classes we tested are the Weighted Tree Algorithm (Cesa-Bianchi et al., 2010), Graph Perceptron Algorithm (Herbster and Pontil, 2007), Offline Label Propagation (Zhu et al., 2003) and Online Majority Vote. Multiple versions of some were compared.

The data we used was created by us, using named entities from news stories as nodes, co-occurrence information as links, and various entity-properties as labels. The networks we used in this experiment were limited to a size of about 3000 nodes (in order to ensure all algorithms we tried could converge), but we are releasing networks as large as 21K nodes, on the website associated to this article, as well as the source code of all methods.

The networks were generated by analysing over 5 million news articles, extracting 21K people named in the news, with 325K co-occurrence links between them. The extraction process involves co-reference resolution, statistical co-occurrence analysis, and statistical analysis for label assignment. In addition, these networks are produced by an autonomous system and as such are constantly growing in size.

It is remarkable that in all our studies, the same algorithm—Online Majority Vote—was the best performer both in terms of accuracy and in terms of computational complexity. Other algorithms demonstrated other advantages however.

The paper is organised as follows. First, we define the problem and some background on label prediction algorithms. Second, we explain the data that we have used for this experiment and how it was inferred from online news. Third, we release large labelled networks and a library containing implementations of our algorithms. Then, we present and discuss our findings prior to concluding.

2 PROBLEM DEFINITION

Label prediction operates on an undirected network, or graph, $G(V,E)$. There are n vertices in this graph, $V = \{1,...n\}$, and edges between vertices have positive weights, $w_{i,j} > 0$ for $(i,j) \in E$. Large edge weights suggest that the vertices they connect are close to one another.

Associated with the vertices of this graph are a set of binary labels, $\mathbf{y} = (y_1,...y_n) \in \{-1,+1\}$. Every vertex is therefore labelled positively or negatively.

One factor that affects the performance of label prediction algorithms is the regularity of the labels on the input graph. If positive labels tend to be next to one another then the graph is very regular, whereas if they are randomly assigned it is very irregular. Regularity is to be expected in our networks, given than people are connected based on the news in which they appear. Consider for example, Lewis Hamilton and Felipe Massa, who are both labelled as sportsmen and linked together; this is a result of them appearing togther in Formula One news articles.

We measure the irregularity of a graph using the *cutsize*. Intuitively, this is proportional to the number of edges with endpoints that have different labels. We consider the set $E^\phi \subseteq E$, where a ϕ-edge is any (i,j) such that $y_i \neq y_j$. We define $\Phi = \sum_{(i,j)\in E^\phi} w_{i,j}$.

Normalising the cutsize by the total weight of the graph gives us a measure of the *irregularity* of any graph:

$$\text{irregularity}(G(V,E)) = \frac{\sum_{(i,j)\in E^\phi} w_{i,j}}{\sum_{(i,j)\in E} w_{i,j}} \quad (1)$$

3 METHODS

We run these experiments as a black-box test, using implementations of four label prediction algorithms. Some methods tested are online predictors: Online Majority Vote (OMV), Perceptron with Graph Laplacian Kernel (GPA), Weighted Tree Algorithm (WTA), whilst another, Offline Label Propagation (LABPROP) is used for comparison. The following paragraphs give brief outlines of each method.

We should also point out that WTA and GPA have strong theoretical guarantees and mistake bound analyses. In particular, the number of mistakes made by WTA is directly proportional to the expected value of Φ on a random spanning tree. Similarly, those made by GPA are proportional to the value of Φ and the diameter of the graph.

In our experimental section, all predictors are operated with a train/test protocol instead of an online one, however this does not make any difference for our purposes.

3.1 Online Majority Vote (OMV)

Online majority vote is the simplest and fastest of methods that we test. It is very intuitive, predicting the label of a vertex by looking at those of its immediate neighbours and taking a majority vote. This vote is weighted according to edge weights and can be written as $\hat{y}_i = sgn(\sum_{j\sim i} w_{i,j} y_j)$ (where $j \sim i$ means that j is connected to i). Time and space complexity are both of $\Theta(|E|)$ (Cesa-Bianchi et al., 2010).

3.2 Offline Label Propagation (LABPROP)

Offline label propagation (Zhu et al., 2003) operates on an entire graph at once and attempts to minimise the energy in a system of labelled vertices. Energy in this case is very similar to the previously mentioned cutsize. Therefore its task is to propagate known labels in such a way as to minimise the cutsize of the graph; that is to produce the most regular graph possible given the known labels.

Given the label values, $\mathbf{y} = (y_1,...y_n) \in \{+1,-1\}$, the algorithm calculates a real-valued function for each unlabelled vertex as follows:

$$f(j) = \frac{\sum_{j\sim i} w_{ij} f(i)}{\sum_{j\sim i} w_{ij}} \quad (2)$$

Where $j \sim i$ denotes that j is connected to i.

The function in 2 can be applied to each unlabelled vertex, iteratively until convergence, in a similar manner to the PageRank algorithm (Page et al., 1998). Positive labels are then applied to vertices with $f(j) > 0$. This decision threshold can be manipulated to reflect the input class distribution, or to incorporate prior knowledge (Zhu et al., 2003).

Our implementation uses a method described in (Zhu and Ghahramani, 2002) and has a time complexity of $O(k|V|^2)$, where k is the average degree of vertices in the input graph.

3.3 Perceptron with Graph Laplacian Kernel (GPA)

A kernel perceptron is used to predict labels on the graph (Herbster and Pontil, 2007). This uses a kernel based on the Laplacian of the graph to predict the labels. This involves inverting the Laplacian, which is computationally expensive; therefore they compute a spanning tree of the graph and predict on that, which is much more efficient. With the methods developed in (Herbster et al., 2009) we operate on a $|V| \times |V|$ matrix, giving a complexity of $O(|V|^2 + |V|S_T)$, where S_T is the structural diameter or maximum depth of the tree.

Intuitively, GPA embeds the graph in a vector space using the Laplacian kernel, so well connected nodes on the graph will be close in this vector space. Given that we expect strongly connected nodes to have the same label, so too do we expect that nearby nodes in the vector space will have the same label; therefore we can use a Perceptron to classify them.

3.4 Weighted Tree Algorithm (WTA)

The weighted tree algorithm (Cesa-Bianchi et al., 2010) is a bit different from competitors since it operates only on trees. This makes this algorithm extremely scalable, running in constant amortised time per prediction, $O(|V|)$ to predict all labels on a tree.

WTA takes a tree as input, then goes one step further and converts the input tree into a line, which it makes its predictions on using a nearest neighbour rule.

A tree, T, is converted into a line by first selecting a vertex of the tree at random. It then performs a depth-first traversal of the tree and stores this path as a line, L'. Backtracking steps are included in this process, meaning that the line can contain duplicate vertices and edges.

Duplicate vertices in the line, L' are then removed. Neighbouring vertices are connected, and the edge takes on the weight of the weakest connection between the removed vertex and its neighbours. This line, L, which has had duplicates removed is the input to the weighted tree algorithm.

The algorithm predicts the label of an unlabelled vertex by traversing the line, L, and returning the closest label that is found. In (Cesa-Bianchi et al., 2010) there is a clear explanation about how to implement this algorithm efficiently.

3.5 Extracting a Tree from a Graph

For a variety of motivations, both WTA and GPA operate on a spanning tree instead of the complete graph. It is therefore important to understand how different techniques to extract those spanning trees will influence our results. There are many alternative ways to transform a graph into a tree and we will evaluate some of them by comparing their properties. The methods evaluated are:

- Minimum Spanning Tree (MST): chosen because it gave some of the best results in previous literature (Herbster et al., 2009)(Cesa-Bianchi et al., 2010).
- Random Spanning Tree: we choose two different types of random spanning tree because of their connections with the resistance distance (as explained in (Cesa-Bianchi et al., 2010)). These trees are extracted using algorithms with different computational complexities: the first is Broder's algorithm (Broder, 1989) (RWST) that runs in $O(|V|\log|V|)$; the second is Wilson's algorithm (Wilson, 1996) (NWST) that runs in $O(|V|)$ (both are not in the worst case).
- Depth-first Spanning Tree (DFST): chosen because it is a common technique for generating spanning trees.
- Minimum Diameter Spanning Tree (SPST): chosen because GPA's mistake bound also depends on the diameter of the tree.

3.6 Committees of Trees (CWTA)

We also tested a committee version of WTA (CWTA), which uses 11 of the randomly generated trees (DFST, NWST, RWST). The algorithm is the same as standard WTA except that the output label is the majority vote between different instances of WTA running on different trees.

4 NETWORK GENERATION

4.1 News Collection

Network generation begins with the collection of news articles. These are collected by a spider, which visits over 3,000 syndicated news feeds at regular intervals throughout each day. News feeds are simple XML files that can be easily monitored and parsed by machines. The spider is able to handle a large volume of news and does so using a multi-threaded architecture that minimises the effect of network delays. It parses feeds in the RSS and Atom formats using an open source Java library, Rome[1].

Topic labels are automatically applied to articles based on the feed(s) in which they appear. For example, an article that appears in both the 'business' and 'politics' feeds of the BBC, will inherit both of these topics as *tags*. Tags are simple labels that can be attached to articles in our system.

The data acquisition system forms the basis of many other works (Ali et al., 2010), (Ali and Cristianini, 2010), (Flaounas et al., 2010a), (Flaounas et al., 2010b), (Turchi et al., 2009) and also has its own suite of monitoring tools (Flaounas et al., 2011).

4.2 Named Entity Extraction

References to people are extracted from articles of news using a named entity recognition tool called GATE (Cunningham et al., 2002). People can be referred to in many different ways, by omitting their middle names, or misspelling them, for example. Resolving duplicate references to a specific named entity is the problem of entity matching.

Entity matching is a well-studied problem, however it is one that is greatly affected by scale. Resolving duplicates among a large input of references (we monitor around 40 million) is not computationally feasible as all pairs must be compared, which is of $O(n^2)$ complexity.

We resolve duplicate references using an approach that focuses on high precision. It first uses social network information to limit the search space, then applies string similarity measures and co-reference information to determine which references to merge into entities (Ali and Cristianini, 2010). This and further filtering steps (for example, ignoring those who are mentioned only a few times) reduce our attention to the 50,000 most-popular people in world media, seen over a 2.5 year period.

4.3 Inference of Topic Labels

All articles in the system are associated with various tags inherited from the feeds in which they are mentioned. We can therefore infer a lot about an entity based on the articles that mention it. We would expect to see a significant relation between Lewis Hamilton and sports articles, for example, however we might also find that he appears in many entertainment articles.

Prior to calculating the most significant relations between entities and tags we must first collect basic statistics about each.

[1]Rome Library: https://rome.dev.java.net/.

Our input is a set of N articles, $\{a_1, a_2, ...a_N\} \in A$. Every article can contain a subset of entities in our system, $\{e_1, e_2, ...e_M\} \in E$, as well as a subset of property tags, $\{p_1, p_2, ...p_Q\} \in P$. The following values are then counted:

$$\begin{aligned} c(e) &: \text{number of times entity } e \text{ has appeared} \\ c(p) &: \text{number of times property } p \text{ has appeared} \\ c(e,p) &: \text{number of times } e \text{ has appeared in articles} \\ &\quad \text{tagged with } p \end{aligned}$$

These values allow a contingency table to be produced for every observed pair of entity and tag. This summarises their appearances, including the number of times that they appear together, separately, or not at all.

In some cases it is possible for cell counts to equal zero. This typically occurs when an entity is *always* seen in the context of a particular tag, or vice versa. We may, for example, see Lewis Hamilton appear in only sports news. This causes $c(e) = c(e,p)$ and as a consequence sets cell $b = c(e) - c(e,p) = 0$.

Undefined odds ratios are avoided by applying a Laplace correction to the contingency table. We pretend to have seen four additional articles in which we have seen: the entity alone; the tag alone; the entity and tag together; neither the entity nor the tag. Table 1 shows how values are calculated for each cell of the contingency table.

4.3.1 The Odds Ratio

To measure the strength of association between an entity and a tag we use the odds ratio (Boslaugh and Watters, 2008). If we label the cells of the contingency table as a, b, c, d, we can calculate the odds ratio using (3). The odds ratio is exponentially distributed, which makes thresholding and comparison of values more difficult later on. For this reason we use the natural logarithm of the odds ratio (3), which is approximately normally distributed.

$$\text{OR}(e,p) = \frac{a.d}{b.c}, \quad \text{LOR}(e,p) = \log\left(\frac{a.d}{b.c}\right) \qquad (3)$$

Log odds ratios (LOR) close to zero suggest that the entity and tag appear independently, such that their appearances together were just down to chance. Values greater than zero suggest that there is some association between the pair and values less than zero suggest a negative association; that is that a particular tag means that the entity is unlikely to also appear.

Table 1: A contingency table for appearances of entities and tags, corrected so that cells are always greater than zero.

	p	$\neg p$	Total
e	$c(e,p)+1$	$c(e)-c(e,p)+1$	$c(e)+2$
$\neg e$	$c(p)-c(e,p)+1$	$N-c(e)-c(p)-c(e,p)+1$	$N-c(e)+2$
Total	$c(p)+2$	$N-c(p)+2$	$N+4$

4.3.2 Significant Values of the Odds Ratio

Increasing values of odds ratio for any tag and entity suggests progressively stronger association between the two. Prior to applying tags to any entity it must be decided exactly which values of LOR are significant.

To decide this we calculate the standard error of the LOR, and confidence intervals as shown in equation (4).

$$\mathrm{SE}_{\mathrm{LOR}} = \sqrt{\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d}}, \quad \mathrm{CI}_{\mathrm{LOR}} = \mathrm{LOR} \pm z.\mathrm{SE}_{\mathrm{LOR}} \tag{4}$$

Where z represents the level of significance that is required, as given by the standard normal distribution ($\mu = 1, \sigma = 1$). At the 99.9% significance level, for example, we use $z = 3.2905$.

Recall that values of LOR are more likely to be considered significant if they are sufficiently far from zero, either positive or negative. Whether or not a value is sufficiently far from zero is determined by the confidence interval; if the confidence interval includes zero then we cannot say that the value is significant, but if it does not include zero then we can.

4.3.3 Comparing Odds Ratios

At this stage we have the tools to decide whether any entity and tag are significantly associated or not. To do so we use the odds ratio and our confidence in it; if the odds ratio is large *and* our confidence interval is small we can say that the association is significant.

We now consider how the strength of association between pairs of tag and entity can be compared to one another. For example, is Lewis Hamilton more strongly associated to sports or entertainment news? To be able to make such a comparison we scale the odds ratio by the standard error, at our desired level of significance. This ensures that all associations can later be compared directly in order to make fast decisions as to which tag connects most strongly to which entity.

This correction is shown in (5). Any associations with $|a_{\mathrm{odds}}| < 1$ cannot be said to be significant, whereas any with $|a_{\mathrm{odds}}| \geq 1$ are considered significant.

$$a_{\mathrm{odds}} = \frac{\mathrm{LOR}(e,p)}{z.\mathrm{SE}_{\mathrm{LOR}}} \tag{5}$$

4.3.4 Example Topic Labels

Every entity now has a comparable association score, a_{odds} that describes its level of association to any particular topic of news. This information is continuously updated daily such that topic associations can be requested immediately. For the purposes of label prediction we simply use the strongest topic association for each entity in the network being studied.

Table 2 shows some randomly-sampled examples of people who are significantly associated to business, entertainment, and politics articles. Note that there are some examples where duplicates still exist (‘Trichet’), or where an organisation has been classified as a person (‘X Factor’). This is due to the automated nature of the system; references are extracted, cleaned and resolved without any human input.

4.4 Creating Networks

Previous steps have collected news, extracted and resolved references to entities, then inferred topic labels. We use the results of these steps to produce networks for use in label prediction.

4.4.1 Significant Links

People are linked based on the number of articles in which they co-occur. A network, $G(V,E)$, has weighted vertices, $w(v)$, and weighted edges, $w(e) = w(v_1, v_2)$. The weight of a vertex, $w(v)$ is equal to the number of articles in which v is mentioned, and the weight of an edge $w(v_1, v_2)$ is equal to the number of articles that v_1 and v_2 are both mentioned in. In addition each social network is generated from a set of news articles, A, which amounts to $N = |A|$ observations.

Raw co-occurrence counts do not represent the true strength of connection between pairs of people. Popular people will tend to be connected strongly due to their high occurrence counts, therefore we use the χ^2 (chi-squared) test statistic to find significantly linked pairs of people.

Table 2: Examples of people who are significantly associated with a selection of topics. Some mistakes get through the automated system and are italicised.

Business	Entertainment	Politics
Sir Richard Branson	Justin Bieber	President Barack Obama
Jean-Claude Trichet	*X Factor*	John McCain
President Jean-Claude Trichet	Angelina Jolie	Harry Reid
Trichet	Natalie Portman	Mitch McConnell
Ben Bernanke	Britney Spears	John Boehner
Chris Bowen	Anne Hathaway	Sarah Palin
Russell Simmons	Harry Potter	Rep. Charles Rangel
Ken Burns	Kim Kardashian	Speaker Nancy Pelosi
Madoff	Johnny Depp	David Cameron
Bernanke	Justin Timberlake	Ed Miliband

As an initial step, we discard any edges where $w(e) < 5$ in order to remove some noise from the network, and as a requirement for the χ^2 statistic.

The χ^2 test of independence compares *observed* counts of occurrences and co-occurrences against what would be *expected* if entities appeared independently. For any pair of entities, (v_1, v_2), we observe:

- N, number of articles.
- $O_{v_1} = w(v_1)$, occurrence count of entity a.
- $O_{v_2} = w(v_2)$, occurrence count of entity b.
- $O_{v_1,v_2} = w(v_1, v_2)$, co-occurrence count of (v_1, v_2).

Therefore, under the null hypothesis, we expect v_1 and v_2 to co-occur $(O_{v_1} * O_{v_2})/N$ times.

We build a 2x2 contingency table for the pair (v_1, v_2) with four possible outcomes: $(v_1, v_2), (\neg v_1, v_2), (v_1, \neg v_2), (\neg v_1, \neg v_2)$. This is built once for our observed values, O and again for the expected values, X. The chi-squared statistic is then calculated using (6).

$$\chi^2(v_1, v_2) = \sum_{v_1 \in \{v_1, \neg v_1\}} \sum_{v_2 \in \{v_2, \neg v_2\}} \frac{(O_{v_1,v_2} - X_{v_1,v_2})^2}{X_{v_1,v_2}} \tag{6}$$

Edges of the input network, G, are then re-weighted to the χ^2 test statistic. Specifically, $w'(v_1, v_2) = \chi^2(v_1, v_2)$. Next we consider the values of the χ^2 statistic and what makes them statistically significant.

4.4.2 Thresholding

A suitable threshold value of χ^2 is selected using a required level of significance. As used previously, we use a level of 99%, or a p-value of 0.01. This means that one edge in every hundred that we keep is expected to be down to random chance.

Our test makes multiple comparisons between pairs of people and the test statistic is calculated for all pairs which co-occur. In reality we implicitly compare $g = \frac{n(n+1)}{2} - n$ pairs, for n total entities. Therefore we apply the Bonferroni correction and adjust the significance level to $1/g$. For example, with 2,000 people, we could make 1,999,000 comparisons, so we correct $p = 0.01$ to $p/g = 5.0e^{-9}$.

We get a threshold, at this level of significance, by looking up the inverse of the χ^2 cumulative distribution function. Continuing the previous example gives us a threshold of 34.19.

4.4.3 Stricter Filtering

One problem with significance testing is the choice of null hypothesis, which in this case was that people appear in news articles independently. This is not very discriminative, however, as the majority of co-occurrences that we observe are indeed significant. This is simply a result of the fact that news is not reported randomly, but to report actual events about people.

To *further* filter our networks we simply increase the threshold value of the χ^2 statistic. Increasing it results in a sparser network, however we need a way to determine when to stop increasing this value. For this we measure the average log degree, $\bar{d}$, of the network:

$$\bar{d} = \frac{\sum_{v \in V} \log(d(v) + 1)}{|V|} \tag{7}$$

Filtering proceeds by removing edges that are smaller than the threshold, then measuring $\bar{d}$. If $\bar{d}$ is too large, the threshold is increased and the network filtered again. This continues until $\bar{d}$ falls below a target value. Thus, we replace a χ^2 threshold with a threshold value of $\bar{d}$. In our experiments we use a threshold value of $\bar{d} = 1.4$.

Table 3: A description of the networks used in these experiments.

Name	Start Date	End Date	Nodes	Edges
Sparse	2009-10-01	2009-10-04	1,072	3,825
Medium-Sparse	2009-10-01	2009-10-04	3,627	11,875
Dense	2010-01-01	2010-01-04	1,919	53,077

Table 4: Irregularity of the networks for each tag, expressed as a percentage of mismatched edges, taking into account edge weight.

Network	Sport	Science	Gossip	Politics	Business
Sparse	4.43	11.9	12.9	16.1	16.8
Medium	7.39	14.8	11.6	12.6	15.7
Dense	4.02	9.78	12.5	11.1	9.72

4.4.4 The Networks

Three social networks were generated from news stories seen within different periods of time. Next, disconnected components were discarded so that only the largest connected component remained. Edges were re-weighted so that their values are linearly distributed, using $w'_{ij} = log(w_{ij})$. Table 3 summarises the networks that were produced for this experiment.

Next, the vertices of the network were labelled with one of five topics of interest: politics, business, science, sport, and entertainment. Only the most strongly associated label was chosen for any vertex, as it is common for them to have more than one.

This labelled network was then used to generate one new network for each of the five topics, where binary labels denote the presence of absence of the topic label in question. Specifically, we convert the labelled network, G, with labels, $\mathbf{y} \in \{1,2,3,4,5\}$, to multiple graphs, $G_1,...G_5$, each with binary labels, $\mathbf{y} \in \{-1,+1\}$. This ensures that the networks are suitably presented to the algorithms under trial, and therefore each network and algorithm is tested separately on each topic.

Finally, we measured the irregularity of each of these networks to understand how potentially difficult each of the networks is to label. See Table 4 for a summary of the irregularity of each network.

5 DATA AND CODE

As an addition to the networks investigated in this paper we also release[2] a selection of larger networks ranging in size from 14,916 nodes and 54,533 edges, through to 21,487 nodes and 325,329 edges. All are labelled with a large number of statistically associated topic labels, as well as labels derived from the location of the news outlets a person is mentioned in; we can, for example, say that Pope Benedict XVI is currently (as of April 10th 2011) most-strongly associated to news from Italy, The Vatican and Iraq.

These larger networks were not used in our experiments as we could not test all methods on them within available time and memory constraints. However, we release them to encourage further work towards the scalability of methods in this field of research.

We also release the code used for our experiments. Our repository contains all the Java classes necessary to reproduce our experiments, including the predictors and utilities for the creation of spanning trees. This allows users to modify our settings from a simple configuration file, as well as to write their own predictors implementing our interfaces for comparison with existing ones.

6 RESULTS

Each of the algorithms were evaluated on fifteen input networks (three source networks with five topics in each). Ten iterations were run for each combination of algorithm and network. In the case of the tree-based methods this involved the generation of ten spanning trees per algorithm. Every iteration was run using a 50% train, 50% test split; half of the labels are obscured prior to each iteration. Tables 5–7 show results for each topic and algorithm on each of the input networks.

The most striking result is that OMV makes the fewest errors in all of the tested cases. LABPROP performed similarly well, but always fell short of OMV. Both algorithms are very similar, with the only major difference being that LABPROP considers information from more than just the immediate neighbours of a vertex. This suggests that such information only serves to mislead the decision when working on social networks.

Why this is the case needs further investigation. However, the most likely explanation is in the nature of social networks. Consider, for example, the small world hypothesis (Travers and Milgram, 1969) and findings that people are on average connected by only 5.2 steps in a network. Any algorithm that takes into account information found outside of the immediate neighbours of a vertex will quickly consider information from a large proportion of the network.

WTA and GPA both perform less well on the

[2]https://patterns.enm.bris.ac.uk/labelled-networks

Table 5: Sparse Network - Percentage error of each algorithm (standard deviation in brackets), quoted for a number of topics, along with the overall average.

Algorithm	**Business**	**Gossip**	**Politics**	**Sport**	**Science**	**Avg.**
WTA MST	23.4 (3.1)	17.4 (3.3)	18.9 (1.6)	8.24 (2)	18.8 (1.5)	17.4 (2.3)
WTA NWST	23.9 (1.9)	18.5 (2.2)	20.7 (1.6)	8.32 (0.96)	18.7 (1.4)	18.0 (1.6)
WTA DFST	23.3 (1.6)	17.6 (1.3)	21.2 (1.7)	8.04 (1.1)	18.2 (1.3)	17.7 (1.4)
WTA RWST	23.5 (1.5)	20.4 (2.1)	20.7 (1.8)	9.41 (1.4)	18.7 (1.3)	18.5 (1.6)
WTA SPST	24.0 (2.1)	20.6 (1.8)	21.8 (1.5)	9.60 (1.2)	18.7 (1.5)	18.9 (1.6)
CWTA DFST	18.5 (1.2)	11.5 (1.1)	14.9 (1.2)	5.08 (0.61)	14.2 (1)	12.8 (1)
CWTA NWST	18.3 (1.2)	11.7 (1.3)	15.6 (1.1)	5.22 (0.75)	13.7 (1.1)	12.9 (1.1)
CWTA RWST	18.4 (1.4)	11.8 (1.4)	15.4 (1.3)	5.12 (0.64)	13.9 (0.92)	12.9 (1.1)
GPA MST	26.8 (5.9)	24.0 (8.9)	26.0 (9.3)	14.3 (9.2)	22.5 (11)	22.7 (8.8)
GPA NWST	27.7 (5)	28.4 (8)	27.0 (5.3)	14.3 (7.7)	22.4 (9.2)	24.0 (7)
GPA DFST	33.3 (7.7)	32.2 (6.4)	33.2 (8.4)	16.1 (9.7)	23.1 (10)	27.6 (8.4)
GPA RWST	29.6 (6.2)	28.7 (7.1)	27.2 (7)	14.4 (9.7)	19.5 (6.3)	23.9 (7.2)
GPA SPST	27.1 (6.8)	24.7 (7.8)	27.7 (9.5)	16.5 (14)	20.7 (11)	23.4 (9.9)
LABPROP	17.4 (1.1)	10.3 (1.1)	14.9 (1.4)	4.74 (0.8)	13.2 (1)	12.1 (1.1)
OMV	**16.9** (0.91)	**8.91** (0.86)	**13.6** (1.2)	**4.35** (0.62)	**13.0** (1)	**11.3** (0.92)

Table 6: Medium-Sparse Network - Percentage error of each algorithm (standard deviation in brackets), quoted for a number of topics, along with the overall average.

Algorithm	**Business**	**Gossip**	**Politics**	**Sport**	**Science**	**Avg.**
WTA MST	20.0 (0.75)	16.6 (0.78)	19.6 (0.59)	9.32 (0.6)	19.9 (0.65)	17.1 (0.67)
WTA NWST	20.6 (0.59)	17.5 (0.75)	20.2 (0.82)	9.73 (0.53)	19.7 (1)	17.6 (0.74)
WTA DFST	20.2 (0.82)	17.4 (0.84)	19.7 (0.6)	9.03 (0.59)	19.8 (0.66)	17.2 (0.7)
WTA RWST	20.7 (0.76)	17.3 (0.63)	19.8 (0.71)	10.0 (0.67)	19.2 (0.66)	17.4 (0.69)
WTA SPST	21.2 (1.1)	17.9 (0.57)	21.1 (0.6)	10.3 (0.64)	20.4 (0.57)	18.2 (0.7)
CWTA DFST	17.5 (0.66)	13.2 (0.86)	17.5 (0.68)	6.68 (0.48)	16.5 (0.77)	14.3 (0.69)
CWTA NWST	17.4 (0.75)	13.0 (0.61)	17.1 (0.75)	6.66 (0.58)	16.1 (0.68)	14.1 (0.67)
CWTA RWST	17.4 (0.54)	13.0 (0.68)	17.3 (0.66)	6.78 (0.54)	16.2 (0.69)	14.1 (0.62)
GPA MST	24.2 (5.9)	25.4 (6.7)	23.7 (5.8)	12.3 (2.6)	21.5 (7.1)	21.4 (5.6)
GPA NWST	29.0 (9.6)	28.0 (5.7)	28.3 (7.2)	15.8 (7.3)	23.1 (9.3)	24.9 (7.8)
GPA DFST	26.1 (4.1)	33.7 (7.8)	26.5 (6.7)	17.2 (7.7)	22.9 (8.1)	25.3 (6.9)
GPA RWST	26.1 (7.8)	25.5 (6.5)	25.3 (5)	14.4 (4.3)	24.4 (10)	23.2 (6.7)
GPA SPST	27.5 (13)	26.1 (12)	24.7 (6.6)	13.4 (4.9)	18.7 (6.4)	22.1 (8.7)
LABPROP	16.3 (0.93)	11.7 (0.82)	16.1 (0.73)	6.17 (0.57)	15.2 (0.4)	13.1 (0.69)
OMV	**15.5** (0.84)	**10.5** (0.56)	**15.8** (0.56)	**5.68** (0.39)	**14.8** (0.62)	**12.5** (0.59)

Table 7: Dense Network - Percentage error of each algorithm (standard deviation in brackets), quoted for a number of topics, along with the overall average.

Algorithm	**Business**	**Gossip**	**Politics**	**Sport**	**Science**	**Avg.**
WTA MST	22.2 (1)	14.9 (0.51)	17.7 (0.84)	6.99 (0.75)	19.7 (0.96)	16.3 (0.81)
WTA NWST	28.6 (1.1)	20.0 (0.92)	21.3 (1.1)	9.19 (0.89)	23.1 (1.2)	20.4 (1)
WTA DFST	23.6 (0.89)	17.8 (0.7)	18.0 (0.87)	7.01 (0.8)	20.0 (1)	17.3 (0.86)
WTA RWST	27.9 (0.97)	20.4 (1.3)	21.7 (0.96)	9.40 (0.81)	23.0 (0.85)	20.5 (0.98)
WTA SPST	28.7 (1.5)	29.5 (0.97)	22.1 (0.75)	11.7 (0.75)	23.3 (1.1)	23.1 (1)
CWTA DFST	17.8 (0.9)	9.83 (0.53)	13.2 (0.63)	3.57 (0.59)	14.8 (0.77)	11.8 (0.68)
CWTA NWST	18.6 (0.93)	9.97 (0.66)	14.3 (0.91)	4.56 (0.97)	15.0 (0.76)	12.5 (0.85)
CWTA RWST	18.1 (0.97)	9.58 (0.69)	13.8 (1)	4.09 (0.72)	14.6 (0.89)	12.1 (0.85)
GPA MST	38.8 (18)	26.2 (5.2)	22.6 (6.9)	8.55 (0.83)	24.6 (18)	24.2 (9.7)
GPA NWST	37.3 (8.7)	31.1 (5.7)	24.7 (6.4)	14.4 (7.9)	29.6 (16)	27.4 (9)
GPA DFST	36.0 (7)	34.9 (10)	27.7 (6.9)	13.7 (5.6)	24.8 (7.9)	27.4 (7.5)
GPA RWST	38.6 (11)	28.0 (3.9)	23.8 (6.4)	12.8 (7.6)	26.3 (13)	25.9 (8.3)
GPA SPST	35.9 (17)	32.7 (8.9)	22.1 (3.8)	10.6 (4.1)	30.1 (24)	26.3 (12)
LABPROP	17.8 (1.1)	7.84 (0.72)	11.1 (0.73)	3.91 (0.66)	15.0 (0.9)	11.1 (0.82)
OMV	**16.2** (0.8)	**7.50** (0.59)	**10.2** (0.63)	**3.21** (0.36)	**13.8** (0.66)	**10.2** (0.61)

tested networks[3]. We have very similar results to the WTA authors' study (Cesa-Bianchi et al., 2010), which found that WTA was generally more accurate than GPA. We also see very similar patterns in the choice of spanning tree; both methods tend to perform best when using the minimum spanning tree (MST), although the differences are typically small. It is important to remember how much information is discarded in converting a graph to a tree, however, a point which will be discussed further in the concluding section.

Committees of trees (CWTA) were also examined and found to perform at a similar level to OMV and LABPROP. In this case the choice of spanning tree appears to make very little difference to the overall performance.

Another aspect to this experiment is the choice of topic. It is interesting to see that all algorithms are better able to propagate sport labels than business labels. Our results show a strong correlation between the irregularity of each network and the error rate; we know that the sparse business network is about four times more irregular than sports, for example. Such aspects are very good candidates for further study into the content of the media.

7 CONCLUSIONS

In this paper we have performed an empirical study of recent algorithms in label prediction. The main aim of this has been to determine which method is the most suitable for predicting missing labels in our automatically generated social networks.

We have found that Online Majority Vote (OMV) was always the most accurate, and in addition is the simplest to implement and is scalable. This might reflect the nature of our social networks and not be a general property, but we believe the network we have generated presents a number of properties that are common to real world social networks.

Both tree based methods (WTA, GPA) do not quite match OMV on this task, however they are the only methods to offer strong theoretical guarantees.

Offline label propagation (LABPROP) performed nearly as well as OMV, which is most likely due to considering information that is too far away from the unlabelled node. It is an interesting algorithm and its similarity to PageRank lends it to be easily parallelisable, particularly given recent methods in cloud computing (Malewicz et al., 2010). Note however, that OMV and WTA are faster in practice.

[3]While this paper was under review, a new algorithm has been presented (Cesa-Bianchi et al., 2011) that outperforms WTA on a set of benchmarks. This new algorithm has not yet been introduced into the current experiment, and this will be part of future work.

One of the biggest problems with social networks are the number of edges they can contain. Storage can quickly become an issue and execution times become dominated by the edge count. This works in favour of methods that scale in the number of *nodes* in a network, for example, WTA or GPA.

What is perhaps most interesting about the tree based methods is the sheer amount of data that they throw away prior to running. Converting a graph into a tree involves the removal of many edges; in the networks that we tested this corresponds to a loss of 72% of edges in the Sparse network, 69% in the Medium-Sparse network and 96% in the Dense network.

To the best of our knowledge there is no very good way to observe connections in a social network and build a suitable tree instead of a network. If this were the case the pre-processing step of converting a graph to a tree would be unnecessary, which would yield two advantages: a way to build a sparse representation of a network, and methods to predict labels on them. Although the tree based methods did not perform quite as well on these networks, they have a lot of potential should issues such as online compression of social network data be satisfactorily solved.

In summary, we find that OMV was the most suitable method for our networks at present. However, we also find many important trade-offs between methods. WTA and GPA offer strong theoretical performance guarantees and also scale in the number of nodes, rather than edges. WTA and OMV are both very fast methods, and WTA can improve its accuracy when run on a committee of trees. LABPROP is accurate at the expense of being slow, however it lends itself to parallelisation and would prove a useful option in a distributed setting, for example.

Finally, we have released the networks used in this study, along with several new networks that are larger in size. An updated version of the source code for label prediction algorithms has also been released. We hope that this data and source code will help researchers in future studies, both on labelled social networks and label prediction algorithms.

ACKNOWLEDGEMENTS

We thank Nicolò Cesa-Bianchi and Claudio Gentile for their advice during these experiments. Nello Cristianini was supported by the Royal Society via a Wolfson Research Merit Award, and by the European

Project ComPLACS and Omar Ali by an EPSRC DTA Grant.

REFERENCES

Ali, O. and Cristianini, N. (2010). Information Fusion for Entity Matching in Unstructured Data. *Artificial Intelligence Applications and Innovations*.

Ali, O., Flaounas, I., Bie, T. D., Mosdell, N., Lewis, J., and Cristianini, N. (2010). Automating News Content Analysis: An Application to Gender Bias and Readability. *JMLR: Workshop and Conference Proceedings*, pages 1–7.

Boslaugh, S. and Watters, P. A. (2008). *Statistics in a Nutshell: A Desktop Quick Reference (In a Nutshell (O'Reilly))*. O'Reilly Media.

Broder, A. (1989). Generating random spanning trees. *30th Annual Symposium on Foundations of Computer Science*, pages 442–447.

Cesa-Bianchi, N., Gentile, C., Vitale, F., and Zappella, G. (2010). Random spanning trees and the prediction of weighted graphs. In *Proceedings of the 27th International Conference on Machine Learning*.

Cesa-Bianchi, N., Gentile, C., Vitale, F., and Zappella, G. (2011). See the Tree Through the Lines: The Shazoo Algorithm. In *Proceedings of the 25th Conference on Neural Information Processing Systems*.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics*, pages 168–175.

Flaounas, I., Ali, O., Turchi, M., Snowsill, T., Nicart, F., De Bie, T., and Cristianini, N. (2011). NOAM : News Outlets Analysis and Monitoring System. *SIGMOD*, pages 1–3.

Flaounas, I., Turchi, M., Ali, O., Fyson, N., De Bie, T., Mosdell, N., Lewis, J., and Cristianini, N. (2010a). The structure of the EU mediasphere. *PloS one*, 5(12):e14243.

Flaounas, I. N., Fyson, N., and Cristianini, N. (2010b). Predicting Relations in News-Media Content among EU Countries. *Cognitive Information Processing (CIP), 2nd International Workshop on*.

Herbster, M. and Pontil, M. (2007). Prediction on a graph with a perceptron. *Advances in neural information processing systems*, 19:577.

Herbster, M., Pontil, M., and Rojas-galeano, S. (2009). Fast Prediction on a Tree. *Neural Information Processing Systems*.

Malewicz, G., Austern, M., Bik, A., Dehnert, J., Horn, I., Leiser, N., and Czajkowski, G. (2010). Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 International Conference on Management of Data*, pages 135–146. ACM.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. *World Wide Web Internet And Web Information Systems*, pages 1–17.

Travers, J. and Milgram, S. (1969). An Experimental Study of the Small World Problem. *Sociometry*, 32(4):425.

Turchi, M., Flaounas, I. N., Ali, O., De Bie, T., Snowsill, T., and Cristianini, N. (2009). Found in Translation. In *ECML/PKDD*, pages 746–749, Bled, Slovenia. Springer.

Wilson, D. B. (1996). Generating random spanning trees more quickly than the cover time. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, pages 296–303.

Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. *School Comput. Sci., Carnegie Mellon Univ., Tech. Rep. CMUCALD-02-107*.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning*, volume 20, page 912.