Analysing Dependency Dynamics in Web Data

Felix Bießmann
Department Machine Learning
TU Berlin
fbiessma@cs.tu-berlin.de

Andreas Harth
Institute AIFB
Karlsruhe Institute of Technology
harth@kit.edu

Abstract

Modern web sites provide easy access to large amounts of data via open application programming interfaces. Users interacting with these sites constantly change the underlying data sets, which can be represented in graph-structured form. Nodes in these dynamic graph structures exhibit dependencies over time (for example, one node changes before other nodes change in the same way). Analysing these dependencies is crucial for understanding and predicting the dynamics inherent to temporally changing graph structures on the web. When the graphs become large however, it is not feasible to take into account all properties of the graph and in general it is unclear how to choose the appropriate features. Moreover, comparing two nodes becomes difficult, if the nodes do not share exactly the same features. In this work we propose an algorithm that automatically learns the features that govern temporal dependencies between nodes in large dynamic graph structures. We present preliminary results of applying the algorithm to data collected from the web, discuss potential extensions of the framework and anticipate how a major problem in machine learning, sparse data, could be tackled by leveraging Linked Data.

Introduction

Modern web sites hosting user-generated content typically provide access to their underlying data bases via specialised application programming interfaces (APIs) or a set of Semantic Web standards. Increasingly web sites follow Linked Data principles for publishing their data (e.g. http://dbtune.org/). Data published as Linked Data follows a few basic rules to facilitate publishing, sharing, and interlinking data on the web[1]. Much of the publicly available data (e.g. statistical information, user preference data, social networks) not yet adhering to Linked Data standards is in principle applicable for analysis with machine learning methods.

Traditional machine learning research has focussed on corpora from dedicated repositories such as UCINet. A recent trend is towards so-called data-driven approaches with a focus on improving learning methods by scaling up the amount of data which is passed as input to algorithms. The appearance of real-world and up-to-date data sets published by online sources which are interlinked has the potential of replacing dedicated corpora and providing insights into phenomena which were not observable before due to a lack of available data, or the high effort involved in collecting data or integrating disparate data sets.

One property of real-world data sets from the web is that the graph structure is not constant over time, the nodes and edges change over time. If there are common sources of information that give rise to similar changes at different nodes in the graph or in different subgraphs, this will lead to temporal dependencies between these nodes or subgraphs. Understanding these dependency dynamics of rapidly evolving graph structures on the web can be helpful for understanding common trends or predicting future ones.

We propose a framework for applying an extension of a standard statistical learning technique, Canonical Correlation Analysis (CCA) (Hotelling 1936), to data from the web to uncover temporal dependencies therein. The questions that can be tackled using this framework include:

- *Which features describe best the dependencies?* If two websites publish similar content at the same time, this will be reflected in the temporal correlation of certain features such as links to other websites from this website. If the websites are large, it is difficult to find the right features, as the number of feature comparisons scales quadratically with the number of features. Our approach automatically learns the features that exhibit temporal correlations, e.g. the links shared by the two websites. As the algorithm is based on kernels (Müller et al. 2001; Schölkopf and Smola 2002; Shawe-Taylor and Cristianini 2004), it is applicable to high-dimensional feature spaces and its computational complexity scales with the number of temporal samples, not the number of features.

- *How do the dependencies change over time?* If one website has published the same content before another website does so, the latter is correlated with the former. This correlation is not instantaneous but has a temporal delay. Thus at time lag zero (i.e. simultaneous time samples of the two websites), the two websites might not be correlated. When taking into account the temporal delay between the two websites however they are correlated. The canonical correlogram (Bießmann et al. 2009) used in

[1] `http://www.w3.org/DesignIssues/LinkedData`

this work reflects these dependency dynamics between the two graph nodes. If the time delay between two websites is always the same, this will be reflected in a pronounced peak in the correlogram at the correct temporal delay between the websites. If one website is publishing the same information simply slower than the other (with variable delay), this will merely be reflected in an asymmetry of the correlogram, i.e. higher correlations for positive time lags $\tau > 0$.

- *Which features account best for this change?* The algorithm presented below learns the features that give rise to the dependency changes between nodes in a graph, even if the dimensionality of the feature space is considerably larger than the number of time samples. If two websites publish only some information, i.e. certain features, with a specific temporal delay and other information at the same time this will be reflected in the analysis results.

A tenet of Linked Data is knowledge representation, whereby data is encoded into a graph structure and rich representation mechanisms allowing for drawing logical inferences. In contrast, machine learning algorithms typically operate on vector or matrix representations of data and most methods that preserve the graph structure, such as graph or string kernels (Shawe-Taylor and Cristianini 2004), do not take into account the temporal dimension. In this work we regard the data as a multivariate time series of features extracted from a graph. This enables us to analyse the temporal dynamics of the graph and the temporal dependencies therein.

In the remainder of the paper we start by introducing an example, followed by an overview of our approach. We continue by presenting experimental results, discuss the challenges in applying machine learning algorithms to Linked Data, and conclude with a summary and outlook.

Example

For the example we use data from http://last.fm/, a social music platform. Users of the service can listen to music, connect to other users and receive recommendations based on the collective listener behaviour of the site's users. last.fm contains a database of tracks, albums, and artists, which may be connected to entries in external data sources. All data on last.fm, including the listening behaviour of users, is made publicly available via an API[2].

Consider a user called JoanLandor for which data is being made available. Data about JoanLandor consists of attributes such as name, her social network (i.e. the people she knows), group membership, and a set of favourite (i.e. often listened to) tracks, albums, and artists. Friends in turn have data attached similar to JoanLandor's. Data about any object in the user subgraph can contain links to external information sources such as http://musicbrainz.org/, http://www.bbc.co.uk/ or http://www.wikipedia.org/, however, not all artists, albums or tracks are connected to external information sources. In addition to the static data about a user, there exist also weekly artist charts per user as depicted in figure 1, which provides the temporal dimension as input to the machine learning algorithm.

Methods

Overview

In this section we outline data collection, preprocessing and analysis steps performed. The process can be divided into the following parts:

- Data Collection: Crawl graph time series
- Feature Extraction: Vectorise graph features
- Dependency Estimation: Compute Temporal Kernel CCA

Data Collection

We collected user graphs from publicly available data on the web. As outlined in the example above, we collected for each of the U user nodes $n_u,\ u = [1, 2, \dots, U]$ in a social network a subgraph expanding from that node. For each time sample $t \in [0, 1, 2, \dots, L]$, where L is the total number of time samples, we stored the corresponding subgraph in a dynamic graph structure $g_u(t)$. An example of one time sample is shown in fig. 1. The data set specific details are explained in the experiments section. The temporal resolution of the graph time series is upper-bounded by the website under investigation. In the case of http://www.last.fm/ we acquired one snapshot of a user graph per week. For websites changing at a faster pace, such as http://twitter.com/ or news feeds, the temporal resolution could be higher.

Feature Extraction

After having downloaded data for every point in time t we extract features from the dynamic graph $g_u(t)$. Let us consider a node n_u from which we collect M features. The multivariate variable extracted from n_u is denoted $x \in \mathbb{R}^M$. Every feature represents one dimension in the vector space of our multivariate time series. The features considered here are simply histograms of graph edges from a user graph to other nodes. This histogram is similar to the term frequency-inverse document frequency (tf-idf) measure in text mining (Salton and McGill 1986), except that we normalise along the temporal dimension (see below). Feature extraction is done for every time sample $t \in [0, 1, 2, ..., L]$. The resulting data matrix $X = [x_1, x_2, x_3, \dots, x_L] \in \mathbb{R}^{M \times L}$ contains the samples x_t in chronological order as column vectors.

In the example above the first feature could be the number of a certain genre label associated with the top ten tracks of a users weekly chart-list. For instance if the first label is *Disco* and a user has three songs in his first weekly chart list labelled with the tag *Disco*, then the matrix entry $X_{(1,1)}$ is 3. If in the following week one more disco song is in the chart list, the corresponding value in the second week $X_{(1,2)}$ is 4.

[2] There exists an RDF representation of last.fm data at http://dbtune.org/last-fm/, however, that version does not include records of weekly playlists on which our algorithm operates. We hence use an idealised graph-structured representation of the data attainable through the last.fm API for ease of exposition.

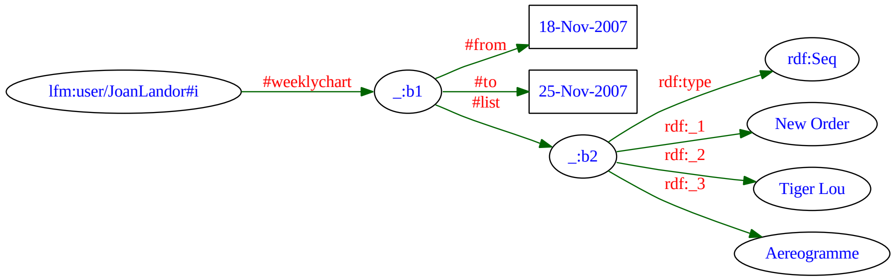

Figure 1: Example of one time sample of a dynamic graph $g_u(t)$, representing the weekly artist charts for the user JoanLandor with the top-three artists.

Temporal kernel CCA (tkCCA)

In this work we used temporal kernel canonical correlation analysis (tkCCA) (Bießmann et al. 2009) in order to analyse the dependencies between two nodes n_x, n_y in a graph. For the sake of a simpler notation we consider only centered and normalised time series, i.e. $\sum x_i = 0$, $\sum x_i^2 = 1$. Given two multivariate time series $x \in \mathbb{R}^M$ and $y \in \mathbb{R}^N$, tkCCA finds a convolution $w_x(\tau)$ and a projection w_y that maximise the canonical correlation between the two time series:

$$\max_{w_x(\tau), w_y} \operatorname{Corr}\left(\sum_\tau w_x(\tau)^\top X_\tau, w_y^\top Y\right). \tag{1}$$

Canonical here means that the correlation is *invariant* under affine transformations of the data. The coefficients of $w_x(\tau)$ and w_y indicate which feature of n_x is coupled to which feature of n_y and importantly $w_x(\tau)$ also reveals the relative time shift of that coupling between the two nodes.

For an illustrative example let us consider $w_x(\tau = -5)$ has a high coefficient in its first feature and also w_y has a high value in its first feature dimension. For feature selection that means node n_x and n_y show a strong correlation in their first feature, hence this feature might be important. As for the temporal dynamics of the dependency this means the correlation between n_x and n_y is maximal if the data of n_y is shifted 5 samples backwards in time with respect to n_x. Referring back to our working example the interpretation would be: user x listened preferentially to the genre *Disco* 5 weeks *before* user y did so, probably because there was some band associated with this genre that became popular and user x discovered that band *before* user y did so[3].

Equation 1 can be efficiently solved by embedding one time series into time shifted copies of itself:

$$\tilde{X} = \begin{bmatrix} X_{\tau_1} \\ X_{\tau_2} \\ \vdots \\ X_{\tau_T} \end{bmatrix} \in \mathbb{R}^{MT \times L}, \tag{2}$$

where X_{τ_i} denotes time shifted copies of the data in X and $\tau_i = [-T, -T+1, \ldots, 0, \ldots, T-1, T]$ denotes the relative time shift with respect to the data in Y and T is the maximal time lag. The data matrix $\tilde{X}$ now contains time shifted (with respect to the data in Y) copies of X as additional feature dimensions. The linear kernel matrices $K_{\tilde{X}}$, K_Y are given as the inner product of the data matrices $\tilde{X}$, Y:

$$\begin{aligned} K_{\tilde{X}} &= \tilde{X}^\top \tilde{X}, \\ K_Y &= Y^\top Y. \end{aligned} \tag{3}$$

We can then optimise equation 1 by solving the generalised eigenvalue problem:

$$\begin{bmatrix} 0 & K_{\tilde{X}} K_Y \\ K_Y K_{\tilde{X}} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \rho \begin{bmatrix} K_{\tilde{X}}^2 & 0 \\ 0 & K_Y^2 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{4}$$

The filters $w_{\tilde{x}}$, w_y are given as a linear expansion of the data points (Akaho 2001; Fukumizu, Bach, and Gretton 2007):

$$\begin{aligned} w_{\tilde{x}} &= \tilde{X}\alpha \\ w_y &= Y\beta. \end{aligned} \tag{5}$$

As the coefficients of $w_{\tilde{x}}$ are simply the coefficients for all time lags stacked upon one another, the convolutive filter $w_x(\tau)$ can be easily recovered by reshaping $w_{\tilde{x}}$:

$$w_{\tilde{x}} = \begin{bmatrix} w_x(\tau_1) \\ w_x(\tau_2) \\ \vdots \\ w_x(\tau_T) \end{bmatrix}. \tag{6}$$

From the convolutive filter $w_x(\tau)$ we can compute a *canonical correlogram* between n_x and n_y, similar to a normal cross correlogram, but for high dimensional data: The canonical correlogram is simply the cross-correlation function between the canonical components (i.e. the data sets projected onto the filters $w_x(\tau)$ and w_y):

$$\begin{aligned} \rho(\tau) &= \operatorname{Corr}\left(w_x(\tau)^\top X_\tau, w_y^\top Y\right) \\ &= \frac{w_x(\tau)^\top X_\tau Y^\top w_y}{w_x(\tau)^\top X_\tau X_\tau^\top w_x(\tau) \cdot w_y^\top Y Y^\top w_y} \\ &= \frac{\alpha^\top K_\tau K_Y \beta}{\alpha^\top K_\tau^2 \alpha \cdot \beta^\top K_Y^2 \beta} \end{aligned} \tag{7}$$

where $K_\tau = X_\tau^\top X_\tau$ denotes the kernel computed from the time-shifted data matrices X_τ. The canonical correlogram

[3] If the two users just generally like a certain feature, this would not lead to temporal co-variation between the users and thus cannot account for the described phenomenon

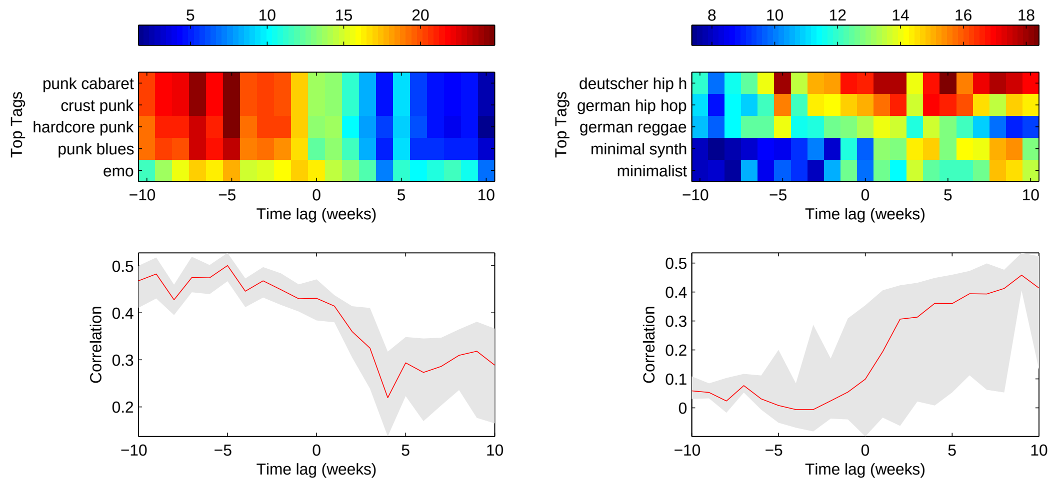

Figure 2: Temporal changes in canonical correlation between single users and all other users; plotted are feature weights (*top row*) and canonical correlogram (*bottom*, median in red, grey area indicates 25th and 75th percentile) for different relative time shifts between single user and all users; **Trend setting** *(left column)*: This users has listened to the respectively tagged artists before everybody else did so, as indicated by the high correlations for negative time lags; the user changed his preferences when others started listening to these categories, as shown by the small correlations for positive time lags; **Trend following** *(right column)*: This user has avoided the respective categories until the masses listened to them and then joined the common listening preference;

$\rho(\tau)$ shows how the canonical correlation between the two nodes n_x, n_y change dependent on the relative time lag between the two nodes. As in our application the number of features is larger than the number of samples, regularisation techniques have to be used to constrain the complexity of $w_x(\tau)$ and w_y. Optimisation of the regularisation parameters was done as in (Bießmann et al. 2009). Confidence intervals for the results below were estimated using 100 bootstrap resamplings.

Experiments

We performed preliminary experiments on two datasets attainable through the web:

Dataset 1

We extracted resource descriptions of $U = 2$ people (Tim Berners-Lee and Kjetil Kjernsmo) from weekly Linked Data crawls. We extracted resources associated via object properties. This yields for each user $u = [1,2]$ a $T \times W$ data matrix X_u, containing $W = 25$ column vectors $x \in \mathbb{R}^T$, where $T = 1129$ is the number of unique object property/resource pairs and W is the number of weekly time samples.

Results for Dataset 1

The goal was to find dependencies between people, for example the change of common interests over time. The overlap between the resource descriptions and especially the rate of change in the descriptions of the people were minimal (as generally the case in most hand-curated Linked Data resources). As the input data was of suboptimal quality, i.e. did not contain temporally covarying data, the dependency measure used did not reveal useful results.

Dataset 2

We extracted a set of $U = 30$ friends on http://www.last.fm/ and downloaded each user's weekly artist charts. From the top ten artists, we extracted the tags associated with that artist. This yields for each user $u = [1,2,...,U]$ a $T \times W$ data matrix X_u, containing $W = 160$ column vectors $x \in \mathbb{R}^T$, where $T = 1293$ is the number of unique tags. So for each week $w = [1,2,...,W]$ we constructed a vector of tag counts x, representing one column of X_u. We then added up all X_u into a common matrix Z to obtain a collective tag histogram over time:

$$Z = \sum_{u=1}^{U} X_u. \quad (8)$$

The $T \times W$ matrix Z contains the collective listening behaviour of the entire friend subgraph crawled. This allows to analyse the dependency dynamics between the single user data in X_u and the data of an entire subgraph of friends in Z. Considering the mean listening behaviour as computed in eq. 8 instead of single users exploits the graph structure of the data: Common features of single users can be summed up to a richer data set. Some users show only irregular listening behaviour and the poor data quality does not allow to draw meaningful inferences, just as in the case of dataset 1. However we can make use of the data of users with irregular listening behaviour by including them in Z.

Results for Dataset 2

In the second experiment we were interested in how single users listening behaviour in X_u was dependent on the collective user listening behaviour in Z. The results show how this dependency changes over time and in particular which features contribute to these dependency changes. Despite the large amount of data collected, the quality of some users' data was still poor due to the fact that not all users use last.fm equally often. In this study we are mainly interested in users that show clear dependencies to the common listening preferences, as those are the ones that are useful for understanding past trends or predicting future ones.

For the ideal user, i.e. one that regularly uses this platform, the results of tkCCA show interesting dependency patterns. In figure 2 we show representative results for two users exemplifying two prototypes of listening behaviour, trend setters and trend followers. The two columns show for each user the canonical correlogram (bottom row) and the feature weights for the top five features over time (top row). Inspecting the correlogram reveals how similar the listening profile of a user was to the collective listening behaviour and how this similarity changed over time. In our setting, for negative time lags the user was ahead of the masses, for positive time lags the masses were ahead of the user. We selected the top 5 features according to their first derivative over time, that is the features plotted are those that changed most over time.

The user in the left column of figure 2 shows high correlations with other users for negative time lags and low correlations for positive time lags. The tags depicted in the feature weights above exhibit the same dynamics. This means that the user listened to the songs labelled with the respective tags long before everybody else did. When more users started to listen to these categories however, the user changed his listening preference and avoided these categories, a listening behaviour which could be interpreted as *trend setting*.

Results for another user shown in the right column of figure 2 display the opposite listening behaviour. The correlations with all users for negative time lags were relatively low, meaning that the user's listening behaviour is not very similar to what all other users will listen to in the future. For positive time lags however, the correlation increases, indicating that as soon as some trend emerged, the user joined the trend. The trend can be identified by inspecting the feature weights over time: the tags constituting that trend show high positive values for positive time lags, a kind of behaviour that can be interpreted as *trend following*.

It is important to note that the although we used the same data representation for the single user data X_u and the collective user data Z, the framework presented can easily deal with data of different dimensionality. This means that we could have estimated the dependencies as well using a different set of tags for either data source. For instance one could imagine another social music platform using different tags. The dependencies between users of different communities could still be analysed, given that the tags are correlated over time.

Discussion

The results show that using a graph-structured data representation we can extract useful information about the temporal dynamics of the underlying data. Prototypical dependency relationships such as *trend setting* or *trend following* can be detected provided the input data is of sufficient quality.

In this work we used music genre tags as features. We chose this feature as it is best suited for our purpose of dependency estimation. Other features, such as the weekly artists playcounts (the weekly in-degree of the artist node) can exhibit a sparse temporal structure, i.e. the artists have high playcounts in few weekly charts but are zero in most weekly charts. This is exemplified in figure 3, *left panel*. Note the maximum in the histogram (right border of time series plot) at playcount 0. In contrast the music genre tags have richer temporal dynamics than the artists' names, as shown in figure 3, *right panel*. The histogram clearly shows the smoother temporal dynamics of the feature time series when genre labels are considered instead of artists' names.

A major problem with tags is that they are a coarse-grained categorisation scheme and often refer to general categories such as "my favourite album" or "albums i own". Instead of tackling these problems explicitly, as in (Lehwark, Risi, and Ultsch 2007), using a more structured taxonomy to classify artists could also yield more meaningful categories.

One of the main advantages of our approach is that by using the links between entities in Linked Data sources, one can incorporate new taxonomic classification schemes. Currently only last.fm provides tag information about artists, however, future work could include features from other platforms as well, for example by exploiting the common song identifiers between last.fm and MusicBrainz for popular tracks. However, the above mentioned sparseness of the input data (i.e. the insufficient track-overlap between users across weeks of listening data) prevents us from using a song-based resolution level. Another example of the sparseness issue is the experiment carried out on Linked Data taken from the Web. As the number of shared identifiers in Linked Data is currently low, the dependency measure used did not reveal useful results.

At first glance it might seem desirable to apply algorithms which are less sensitive to sparse input. The approach outlined in this work is different: We anticipate that existing and rapidly growing publicly accessible data bases could be used in combination as a rich corpus suitable as input to standard machine learning algorithms, which we have demonstrated to work in principle on graph-structured data with a temporal dimension.

The issue of knowledge sparseness on the Semantic Web has been noted in e.g. (Sabou, Lopez, and Motta 2006). Our notion of sparseness is a different one: here we refer to the small amount of temporal (co-)variation of connections between nodes in a data graph. Similar to our work, (Hotho et al. 2006) use tags as resolution and calculate individual ranks at a given time t_m by taking into account all data occurring before m. In contrast, our algorithm uncovers dependencies (such as leader and follower) over time.

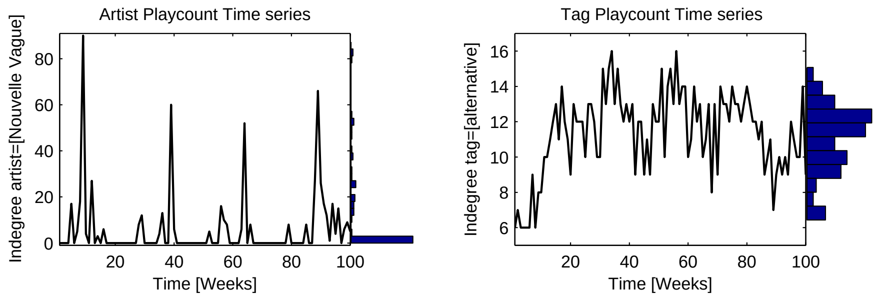

Figure 3: Representative example time series for features extracted from artist play-counts (*left*) and genre label play-counts (*right*); histograms for respective play-count values are attached on the right side; artist play-counts exhibit a sparse temporal structure, whereas genre tag play-counts show richer temporal dynamics;

Conclusion

We have shown how user-generated data sourced from the web can be fruitfully applied to machine learning algorithms to uncover temporal dependencies in data represented as graphs. To be able to derive meaningful results, the input data has to be of sufficient quality and linkage. The rate of change in the subset of Linked Data which we used in our first experiment is currently too small to detect any meaningful temporal correlations, probably because people only irregularly update their hand-crafted RDF documents. In contrast, our second experiment shows that data emanating from users interacting with a web site is well suited for uncovering meaningful temporal patterns.

We have identified the quality of temporal structure in the input data as major factor influencing the quality of results generated by our algorithm. To derive meaningful input from Linked Data for machine learning algorithms we use aggregated data (such as tags or genres) as input rather than entity descriptions directly, given that aggregated data exhibits a dense temporal structure to exploit.

Currently published Linked Data rarely contains data with a temporal dimension. The benefit of automatically publishing detailed data containing temporal properties from within existing web sites would be twofold: Better integration would make large amounts of high-quality data easily available to machine learning algorithms, and machine learning researchers would profit from the new application domain of dynamic web data. Widespread availability of large amounts of high-quality interlinked data allows for a wide range of new applications, for example helping researchers to better understand the behaviour of groups of people or providing better services to users.

Acknowledgements

We would like to thank Pascal Lehwark for help with the last.fm API and Klaus-Robert Müller, Frank C. Meinecke, Julia Veit, and Phillip Sorg for comments on the manuscript. This work has been partialy supported by DFG's Multipla project (grant number 38457858) and the European Union's PASCAL 2 Network of Excellence (ICT-216886).

References

Akaho, S. 2001. A kernel method for canonical correlation analysis. In *In Proceedings of the International Meeting of the Psychometric Society (IMPS2001*. Springer-Verlag.

Bießmann, F.; Meinecke, F.; Gretton, A.; Rauch, A.; Rainer, G.; Logothetis, N.; and Müller, K.-R. 2009. Temporal kernel cca and its application in multimodal neuronal data analysis. *Machine Learning Journal, Special Issue Learning from Multiple Sources, in press*.

Fukumizu, K.; Bach, F.; and Gretton, A. 2007. Statistical consistency of kernel cca. *Journal of Machine Learning Research* 8:361–383.

Hotelling, H. 1936. Relations between two sets of variates. *Biometrika* 322–376.

Hotho, A.; Jschke, R.; Schmitz, C.; and Stumme, G. 2006. Trend detection in folksonomies. In *First International Conference on Semantics And Digital Media Technology (SAMT)*, 56–70.

Lehwark, P.; Risi, S.; and Ultsch, A. 2007. Visualization and clustering of tagged music data. In *31st Annual Conference of the German Classification Society*.

Müller, K.-R.; Mika, S.; Rätsch, G.; Tsuda, K.; and Schölkopf, B. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks* 12(2):181–201.

Sabou, M.; Lopez, V.; and Motta, E. 2006. Ontology selection for the real semantic web: How to cover the queen's birthday dinner? In *15th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 96–111.

Salton, G., and McGill, M. J. 1986. *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc.

Schölkopf, B., and Smola, A. 2002. *Learning with Kernels*. Cambridge, MA, USA: MIT Press.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.