D3.1.2 Regression models of trends Tools for Mining Non-stationary Data: functional prototype

Dr. Sina Samangooei, University of Southampton
Dr. Vasileios Lampos, University of Sheffield
Dr. Trevor Cohn, University of Sheffield
Dr. Nicholas Gibbins, University of Southampton
Prof. Mahesan Niranjan, University of Southampton

Abstract.
FP7-ICT Strategic Targeted Research Project (STREP) ICT-2011-287863 TrendMiner Deliverable D3.1.2 (WP3)

Keyword list: text regression, bilinear, regularisation, multi-task learning, multi-output regression, online learning, Social Media

Project	TrendMiner No. 287863
Delivery Date	May 2, 2013
Contract. Date	April 30, 2013
Nature	Prototype
Reviewed By	Thierry Declerck & Paul Ringler
Web links	http://github.com/sinjax/trendminer
Dissemination	PU

TrendMiner Consortium

This document is part of the TrendMiner research project (No. 287863), partially funded by the FP7-ICT Programme.

DFKI GmbH
Language Technology Lab
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
Germany
Contact person: Thierry Declerck
E-mail: declerck@dfki.de

University of Southampton
Southampton SO17 1BJ
UK
Contact person: Mahensan Niranjan
E-mail: mn@ecs.soton.ac.uk

Internet Memory Research
45 ter rue de la Révolution
F-93100 Montreuil
France
Contact person: France Lafarges
E-mail: contact@internetmemory.org

Eurokleis S.R.L.
Via Giorgio Baglivi, 3
Roma RM
00161 Italia
Contact person: Francesco Bellini
E-mail: info@eurokleis.com

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

Ontotext AD
Polygraphia Office Center fl.4,
47A Tsarigradsko Shosse,
Sofia 1504, Bulgaria
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Sora Ogris and Hofinger GmbH
Bennogasse 8/2/16
A-1080 Wien
Austria
Contact person: Christoph Hofinger
E-mail: ch@sora.at

Hardik Fintrade Pvt Ltd.
227, Shree Ram Cloth Market,
Opposite Manilal Mansion,
Revdi Bazar, Ahmedabad 380002
India
Contact person: Suresh Aswani
E-mail: m.aswani@hardikgroup.com

Executive Summary

This document presents research and software development work from Task 3.1 on supervised learning to correlate streaming social media text with movements in political opinion and financial markets. The central idea is that accurate models of these real-valued time series can identify important features of the text. These features might be important terms, users and/or posts, which are likely to be of interest to analysts and can form part of the summarisation algorithms in WP4 and the display in WP5 (along with the model predictions).

The document contains three strands of work:

1. A novel text regression approach which can automatically identify important users from massive streams of social media text input, as well as a lexicon of important words. This includes evaluation on political and financial time-series prediction, based on the use cases identified in WP6 and WP7 and their data sets. Additionally, we integrate rich knowledge sources into the model: namely sentiment word lists and predicted entity mentions, as output from WP2.

2. A fast online implementation of the regression model. Online methods can process data as it arrives in a single pass, and are much more efficient than the standard batch approaches; the more common machine learning approach. Moreover, online inference will allow for simpler integration into the TrendMiner platform for processing streaming social media text. We present an online inference technique for modelling non-stationary data, where the model is allowed to evolve with time, and evaluate the approach against a stationary model.

3. A users' guide to the software components of the deliverable. We include a batch training component written in Python, which uses parallel processing on a SMP server, and a similar online component written in Java, which can run using cloud computing on streaming input.

Contents

Chapter 1

Introduction

Firstly, we present a new bilinear model of user and words for predicting some real valued instances of a set of related tasks. The algorithm works by predicting one or more time-series as a function of the users and the words they use, with a sparsity inducing term to encourage the model to find small sets of non-zero parameters. This lends itself to end-user interpretation by "selecting" the most important features. This model is pioneering in several ways. It is the first text-regression approach at modelling social media users, particularly at such a large scale, and the first to model related time-series as multi-output regression. We demonstrate its efficacy on political and finance time-series tasks using large datasets sourced from the United Kingdom and Austria. Its high modelling accuracy augurs well for its integration into the TrendMiner platform to identify important text and users from social media streams.

The second part of the document details an online implementation of the same regression model. Online methods can process data as it arrives in a single pass, and are much more efficient than the standard batch approaches; the more common machine learning approach. Moreover, online inference will allow for simpler integration into the TrendMiner platform for processing streaming social media text. We present an online algorithm and experimental work demonstrating that our online learner has very similar predictive performance to our state-of-the-art method for batch training. The online setting also lends itself to non-stationary modelling, where the model is allowed to evolve with time. This allows for the set of important users and words to change with time, which often occurs in real-world problems such as social media which is greatly influenced by linguistic trends, events, and changes to users' patterns of behaviour. We show promising results indicating that accounting for dynamic adaptive modelling as part of online inference can improve predictive performance.

The final part presents a users' guide to the software components of the deliverable. We include a batch training component written in Python, which uses parallel processing on a SMP server, and a similar online component written in Java, which can run using cloud computing on streaming input.

Breakdown of deliverable's objectives:

- **Release 2 of regression models, including deep features from WP2 (temporal, sentiment and event information).**
  Chapter 2 describes a multitude of novel models for temporal text regression; Chapter 3 extends the majority of those models making it possible for them to be applied in an online learning setting. Section 2.3.3 presents a framework for incorporating sentiment features extracted from WP2 and Section 2.3.4 showcases how event information can be incorporated into our regression models.

- **Modelling improvements in the form of parallel implementation, regularisation and multivariate output.**
  All models presented in this script (Chapters 2 and 3) are applying several types of regularisation in order to select and weight a small subset of the entire set of features; some of the models are performing multi-output regression using multi-task learning frameworks (see for example Section 2.2.3).

- **A software system implementing algorithms for non stationary modelling; associated report on implementation details and performance measures; initial implementation.**
  The software systems implementing both the batch and online learners are detailed in Section 3.4. Furthermore, the code to replicate several of the experiments presented in Section 3.3 are also delivered

Published Paper. Vasileios Lampos, Daniel Preoţiuc-Pietro and Trevor Cohn. *A user-centric model of voting intention from Social Media*. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL), 2013. [23]

Chapter 2

Regression models of trends

Web Social Media platforms have ushered a new era in human interaction and communication. The main by-product of this activity is vast amounts of user-generated content, a type of information that has already attracted the interest of both marketeers and scientists because it offers – for the first time at a large-scale – unmediated access to peoples' observations and opinions.

One exciting avenue of research concentrates on mining interesting signals automatically from this stream of text input. For example, by exploiting Twitter posts, it is possible to infer time series that correlate with financial indicators [7], track infectious diseases [21, 22, 32] and, in general, nowcast the magnitude of events emerging in real-life [37, 20]. Other studies suggest ways for modelling opinions encapsulated in this content in order to forge branding strategies [17] or understand various socio-political trends [42, 31]. The main theme of the aforementioned works is linear regression between word frequencies and a real-world quantity. They also tend to incorporate hand-crafted lists of search terms to filter irrelevant content and use sentiment analysis lexicons for extracting opinion bias. Consequently, they are quite often restricted to a specific application or domain and therefore, generalise poorly to new data sets [13].

In this section,[1] we propose a generic method which initially is independent of the characteristics described above (use of search terms or sentiment analysis tools), but in the next steps of our work would be able to easily incorporate such features. Our approach is able to explore not only word frequencies, but also the space of users by introducing a bilinear formulation for this learning task. Regularised regression on both spaces allows for an automatic selection of the most important terms and users (amongst thousands of possible candi-

[1] This section is based on our accepted paper for ACL 2013 [23].

dates), performing at the same time an improved noise filtering. In addition, more advanced regularisation functions enable multi-task learning schemes that can exploit shared structure in the feature space. The latter property becomes very useful in multi-output regression scenarios, where selected features are expected to have correlated as well as anti-correlated impact on each output (*e.g.*, when inferring voting intention percentages for competing political parties).

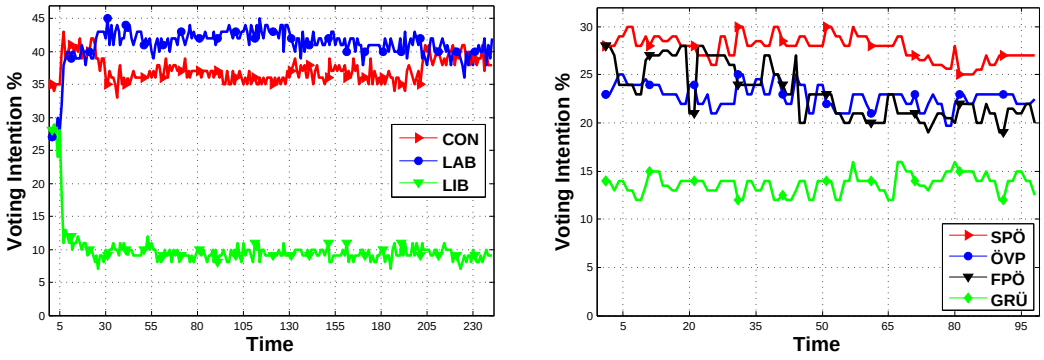
We evaluate our methods on the domain of politics using data from the microblogging service of Twitter to infer voting trends. Our proposed framework is able to successfully predict voting intentions for the top-3 and top-4 parties in the United Kingdom (UK) and Austria respectively. In both case studies – bound by different characteristics (including language, time-span and number of users) – the average prediction error is smaller than 1.5% for our best model using multi-task learning. We also show how our methods are applicable in the financial domain. Finally, our qualitative analysis shows that the models uncover interesting and semantically interpretable insights from the data.

2.1 Description of data sets

For the evaluation of the proposed methodologies we have created two data sets of Social Media content with different characteristics based in the UK and Austria respectively. They are used for performing regression aiming to infer voting intention polls in those countries. Data processing is performed using the TrendMiner architecture for Social Media analysis [35] (see also Deliverable **D3.1.1**).

2.1.1 Tweets from users in the UK

The first data set (we refer to it as $\mathbf{C_{uk}}$) used in our experimental process consists of approx. 60 million tweets produced by approx. 42K UK Twitter users from 30/04/2010 to 13/02/2012. We assumed each user to be from the UK, if the location field in their profile matched with a list of common UK locations and their time zone was set to G.M.T. In this way, we were able to extract hundreds of thousands of UK users, from which we sub-sampled 42K users to be distributed across the UK geographical regions proportionally to



(a) 240 voting intention polls for the 3 major parties in the UK (April 2010 to February 2012)

(b) 98 voting intention polls for the 4 major parties in Austria (January to December 2012)

Figure 2.1: Voting intention polls for the UK and Austria.

their population figures.[2]

2.1.2 Tweets for Austria

The second data set ($\mathbf{C_{au}}$) is shorter in terms of the number of users involved (1.1K), its time span (25/01 to 01/12/2012) and, consequently, of the total number of tweets considered (800K). However, this time the selection of users has been made by **SORA** (Paul Ringler *et al.*) who decided which accounts to monitor by subjectively assessing the value of information they may provide towards political-oriented topics. Still, we assume that the different users will produce information of varying quality, and some should be eliminated entirely. However, we emphasise that there may be smaller potential gains from user modelling compared to the UK case study. Another important distinction is language, which for this data set is primarily German with some English.

2.1.3 Ground Truth

The ground truth for training and evaluating our regression models is formed by voting intention polls from YouGov (UK) and a collection of Austrian pollsters[3] – as none performed high frequency polling – for the Austrian case study. We focused on the three major parties in the UK, namely Conservatives (CON),

[2]Data collection was performed using Twitter API, `http://dev.twitter.com/`, to extract all posts for our target users.

[3]Wikipedia, `http://de.wikipedia.org/wiki/Nationalratswahl_in_\%D6sterreich_2013`.

Labour (LAB) and Liberal Democrats (LBD) and the four major parties in Austria, namely the Social Democratic Party (SPÖ), People's Party (ÖVP), Freedom Party (FPÖ) and the Green Alternative Party (GRÜ). Matching with the time spans of the data sets described in the previous sections, we have acquired 240 unique polls for the UK and 65 polls for Austria. The latter have been expanded to 98 polls by replicating the poll of day i for day $i-1$ where possible.[4] There exists some interesting variability towards the end for the UK polls (Fig. 2.1a), whereas for the Austrian case, the main changing point is between the second and the third party (Fig. 2.1b).

2.2 Methods and models for regularised and multi-output text regression

The textual content posted on Social Media platforms unarguably contains valuable information, but quite often it is hidden under vast amounts of unstructured user generated input. In this section, we propose a set of methods that build on one another, which aim to filter the non desirable noise and extract the most informative features not only based on word frequencies, but also by incorporating users in this process.

2.2.1 The bilinear model

There exist a number of different possibilities for incorporating user information into a regression model. A simple approach is to expand the feature set, such that each user's effect on the response variable can be modelled separately. Although flexible, this approach would be doomed to failure due to the sheer size of the resulting feature set, and the propensity to overfit all but the largest of training sets. One solution is to group users into different types, such as journalist, politician, activist, etc., but this presupposes a method for classification or clustering of users which is a non-trivial undertaking. Besides, these naïve approaches fail to account for the fact that most users use similar words to express their opinions, by separately parameterising the model for different users or user groups.

We propose to account for individual users while restricting all users to

[4] This has been carried out to ensure an adequate number of training points in the experimental process.

share the same vocabulary. This is formulated as a bilinear predictive model,

$$f(X) = \boldsymbol{u}^{\mathrm{T}} X \boldsymbol{w} + \beta\,, \tag{2.1}$$

where X is an $m \times p$ matrix of user-word frequencies and $\boldsymbol{u}$ and $\boldsymbol{w}$ are the model parameters. Let $\mathcal{Q} \in \mathbb{R}^{n \times m \times p}$ be a tensor which captures our training inputs, where n, m and p denote the considered number of samples (each sample usually refers to a day), terms and users respectively; $\mathcal{Q}$ can simply be interpreted as n versions of X (denoted by $\mathcal{Q}_i$ in the remainder of the script), a different one for each day, put together. Each element $\mathcal{Q}_{ijk}$ holds the frequency of term j for user k during the day i in our sample. If a user k has posted $c_{i\cdot k}$ tweets during day i, and $c_{ijk} \leq c_{i\cdot k}$ of them contain a term j, then the frequency of j for this day and user is defined as $\mathcal{Q}_{ijk} = \frac{c_{ijk}}{c_{i\cdot k}}$.

Aiming to learn sparse sets of users and terms that are representative of the voting intention signal, we formulate our optimisation task as follows:

$$\{\boldsymbol{w}^*, \boldsymbol{u}^*, \beta^*\} = \underset{\boldsymbol{w},\boldsymbol{u},\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\boldsymbol{u}^{\mathrm{T}} \mathcal{Q}_i \boldsymbol{w} + \beta - y_i\right)^2 + \psi(\boldsymbol{w}, \rho_1) + \psi(\boldsymbol{u}, \rho_2)\,, \tag{2.2}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is the response variable (voting intention), $\boldsymbol{w} \in \mathbb{R}^m$ and $\boldsymbol{u} \in \mathbb{R}^p$ denote the term and user weights respectively, $\boldsymbol{u}^{\mathrm{T}} \mathcal{Q}_i \boldsymbol{w}$ expresses the bilinear term, $\beta \in \mathbb{R}$ is a bias term and $\psi(\cdot)$ is a regularisation function with parameters ρ_1 or ρ_2. The first term in Eq. 2.2 is the standard regularisation loss function, namely the sum squared error over the training instances.[5]

In the main formulation of our bilinear model, as the regularisation function $\psi(\cdot)$ we use the **elastic net** [47], an extension of the well-studied ℓ_1-norm regulariser, known as the LASSO [40]. The ℓ_1-norm regularisation has found many applications in several scientific fields as it encourages sparse solutions which reduce the possibility of overfitting and enhance the interpretability of the inferred model [16]. The elastic net applies an extra penalty on the ℓ_2-norm of the weight vector, and can resolve instability issues of LASSO which arise when correlated predictors exist in the input data [46]. Its regularisation function $\psi_{\mathbf{el}}(\cdot)$ is defined by:

$$\psi_{\mathbf{el}}\left(\boldsymbol{w}, \lambda, \alpha\right) = \lambda \left(\frac{1-\alpha}{2} \|\boldsymbol{w}\|_2^2 + \alpha \|\boldsymbol{w}\|_1 \right), \tag{2.3}$$

where $\lambda > 0$ and $\alpha \in [0, 1)$; setting parameter α to its extremes transforms elastic net to ridge regression ($\alpha = 0$) or vanilla LASSO ($\alpha = 1$).

[5] Note that other loss functions could be used here, such as logistic loss for classification, or more generally bilinear variations of Generalised Linear Models [29].

Eq. 2.2 can be treated as a biconvex learning task [2], by observing that for a fixed $\boldsymbol{w}$, learning $\boldsymbol{u}$ is a convex problem and vice versa. Biconvex functions and possible applications have been well studied in the optimisation literature [36, 34]. Their main advantage is the ability to solve efficiently non-convex problems by a repeated application of two convex processes, *i.e.* a form of coordinate ascent. In our case, the bilinear technique makes it possible to explore both word and user spaces, while maintaining a modest training complexity.

Therefore, in our bilinear approach we divide learning in two phases, where we learn word and user weights respectively. For the first phase we produce the term-scores matrix $\mathcal{V} \in \mathbb{R}^{n \times m}$ with elements given by:

$$\mathcal{V}_{ij} = \sum_{z=1}^{p} u_z \mathcal{Q}_{ijz}. \tag{2.4}$$

$\mathcal{V}$ contains weighted sums of term frequencies over all users for the considered set of days. The weights are held in $\boldsymbol{u}$ and are representative of each user. The initial optimisation task is formulated as:

$$\{\boldsymbol{w}^*, \beta^*\} = \underset{\boldsymbol{w},\beta}{\operatorname{argmin}} \|\mathcal{V}\boldsymbol{w} + \beta - \boldsymbol{y}\|_2^2 + \psi_{\mathbf{el}}\left(\boldsymbol{w}, \lambda_1, \alpha_1\right) , \tag{2.5}$$

where we aim to learn a sparse but consistent set of weights w^* for the terms of our vocabulary.

In the second phase, we are using $\boldsymbol{w}^*$ to form the user-scores matrix $\mathcal{D} \in \mathbb{R}^{n \times p}$:

$$\mathcal{D}_{ik} = \sum_{z=1}^{m} w_z^* \mathcal{Q}_{izk} \, , \tag{2.6}$$

which now contains weighted sums over all terms for the same set of days. The optimisation task becomes:

$$\{\boldsymbol{u}^*, \beta^*\} = \underset{\boldsymbol{u},\beta}{\operatorname{argmin}} \|\mathcal{D}\boldsymbol{u} + \beta - \boldsymbol{y}\|_2^2 + \psi_{\mathbf{el}}\left(\boldsymbol{u}, \lambda_2, \alpha_2\right) . \tag{2.7}$$

By inserting the weights of the second phase back to phase one, we can iterate the process as in each step we are dealing with a convex problem. We cannot claim that a global optimum will be reached, but biconvexity guarantees that our global objective (Eq. 2.2) will decrease in each step of this iterative process. In the remainder of this paper, we refer to the method described above as Bilinear Elastic Net (**BEN**).

2.2.2 Exploiting term-target or user-target relationships

The previous model assumes that the response variable $\boldsymbol{y}$ holds information about a single inference target. However, the task that we are addressing in this paper usually implies the existence of several targets, *i.e.*, different political parties or politicians. An important property, therefore, is the ability to perform multiple output regression. A simple way of adapting the model to the multiple output scenario is by framing a separate learning problem for each output, but tying together some of the parameters. Here we consider tying together the user weights $\boldsymbol{u}$, to enforce that the same set of users are relevant to all tasks, while learning different term weights. Note that the converse situation, where $\boldsymbol{w}$'s are tied and $\boldsymbol{u}$'s are independent, can be formulated in an equivalent manner.

Suppose that our target variable $\boldsymbol{y} \in \mathbb{R}^{\tau n}$ refers now to τ political entities, $\boldsymbol{y} = \left[\boldsymbol{y}_1^{\mathrm{T}} \boldsymbol{y}_2^{\mathrm{T}} ... \boldsymbol{y}_\tau^{\mathrm{T}}\right]^{\mathrm{T}}$; in this formation the top n elements of $\boldsymbol{y}$ match to the first political entity, the next n elements to the second and so on. In the first phase of the bilinear model, we would have to solve the following optimisation task:

$$\{\boldsymbol{w}^*, \beta^*\} = \underset{w,\beta}{\operatorname{argmin}} \sum_{i=1}^{\tau} \|\mathcal{V}\boldsymbol{w_i} + \beta_i - y_i\|_2^2 + \sum_{i=1}^{\tau} \psi_{\mathbf{el}}\left(\boldsymbol{w}_i, \lambda_1, \alpha_1\right) , \tag{2.8}$$

where $\mathcal{V}$ is given by Eq. 2.4 and $\boldsymbol{w}^* \in \mathbb{R}^{\tau m}$ denotes the vector of weights which can be sliced into τ sub-vectors $\{\boldsymbol{w}_1^*, ..., \boldsymbol{w}_\tau^*\}$ each one representing a political entity. In the second phase, sub-vectors $\boldsymbol{w}_i^*$ are used to form the input matrices $\mathcal{D}_i$, $i \in \{1, ..., \tau\}$ with elements given by Eq. 2.6. The input matrix $\mathcal{D}'$ is formed by the vertical concatenation of all $\mathcal{D}_i$ user score matrices, *i.e.* $\mathcal{D}' = \left[\mathcal{D}_1^{\mathrm{T}} \ ... \ \mathcal{D}_\tau^{\mathrm{T}}\right]^{\mathrm{T}}$, and the optimisation target is equivalent to the one expressed in Eq. 2.7. Since $\mathcal{D}' \in \mathbb{R}^{\tau n \times p}$, the user weight vector $\boldsymbol{u}^* \in \mathbb{R}^p$ and thus, we are learning a single weight per user and not one per political party as in the previous step.

The method described above allows learning different term weights per response variable and then binds them under a shared set of user weights. As mentioned before, one could also try the opposite (*i.e.* start by expanding the user space); both those models can also be optimised in an iterative process. However, our experiments revealed that those approaches did not improve on the performance of BEN. Still, this behaviour could be problem-specific, *i.e.* learning different words from a shared set of users (and the opposite) may not be a good modelling practice for the domain of politics. Nevertheless, this observation served as a motivation for the method described in the next section, where we extract a consistent set of words and users that are weighted differently among the considered political entities.

2.2.3 Multi-task learning with the ℓ_1/ℓ_2 regulariser

All previous models – even when combining all inference targets – were not able to explore relationships across the different task domains; in our case, a task domain is defined by a specific political label or party. Ideally, we would like to make a sparse selection of words and users but with a regulariser that promotes inter-task sharing of structure, so that many features may have a positive influence towards one or more parties, but negative towards the remaining ones. It is possible to achieve this multi-task learning property by introducing a different set of regularisation constraints in the optimisation function.

We perform multi-task learning using an extension of group LASSO [45], a method known as $\boldsymbol{\ell_1/\ell_2}$ regularisation [3, 25]. Group LASSO exploits a predefined group structure on the feature space and tries to achieve sparsity in the group-level, *i.e.* it does not perform feature selection (unlike the elastic net), but group selection. The ℓ_1/ℓ_2 regulariser extends this notion for a τ-dimensional response variable. The global optimisation target is now formulated as:

$$\begin{aligned}\{W^*, U^*, \boldsymbol{\beta}^*\} = \underset{W,U,\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{t=1}^{\tau}\sum_{i=1}^{n}\left(\boldsymbol{u}_t^{\mathrm{T}}\mathcal{Q}_i\boldsymbol{w}_t + \beta_t - y_{ti}\right)^2 \\ + \lambda_1\sum_{j=1}^{m}\|W_j\|_2 + \lambda_2\sum_{k=1}^{p}\|U_k\|_2\,,\end{aligned} \tag{2.9}$$

where the input matrix $\mathcal{Q}_i$ is defined in the same way as earlier, $W = [\boldsymbol{w}_1 \ldots \boldsymbol{w}_\tau]$ is the term weight matrix (each $\boldsymbol{w}_t$ refers to the t-th political entity or task), equivalently $U = [\boldsymbol{u}_1 \ldots \boldsymbol{u}_\tau]$, W_j and U_j denote the j-th rows of weight matrices W and U respectively, and vector $\boldsymbol{\beta} \in \mathbb{R}^\tau$ holds the bias terms per task. In this optimisation process, we aim to enforce sparsity in the feature space but in a structured manner. Notice that we are now regularising the $\ell_{2,1}$ mixed norm of W and U, which is defined as the sum of the row ℓ_2-norms for those matrices. As a result, we expect to encourage the activation of a sparse set of features (corresponding to the rows of W and U), but with nonzero weights across the τ tasks [3]. Consequently, we are performing filtering (many users and words will have zero weights) and, at the same time, assign weights of different magnitude and sign on the selected features, something that suits a political opinion mining application, where pro-A often means anti-B.

Eq. 2.9 can be broken into two convex tasks (following the same notion as in Eqs. 2.5 and 2.7), where we individually learn $\{W, \boldsymbol{\beta}\}$ and then $\{U, \boldsymbol{\beta}\}$; each step of the process is a standard linear regression problem with an ℓ_1/ℓ_2

regulariser. Again, we are able iterate this bilinear process and in each step convexity is guaranteed. We refer to this method as Bilinear Group ℓ_1/ℓ_2 (**BGL**).

2.2.4 Using groups of terms

A great deal of Natural Language Processing research is based on the use of fixed lists of terms, formed automatically or manually. Sentiment analysis, in particular, is usually driven by various types of keywords or phrases mapped to a certain category of sentiment. Those keywords can usually be grouped in classes, for example representing a positive, negative or neutral sentiment bias.

In this section, we propose a bilinear text regression model able to account for such word groupings. Its aim is to exploit this group structure and learn the model parameters accordingly; different groups may influence the response (target) variable differently and in addition not all features included in each may be relevant to the task at hand. The bilinear global optimisation function is formulated as follows:

$$\begin{aligned}\{\boldsymbol{w}^*, \boldsymbol{u}^*, \beta^*\} = \underset{\boldsymbol{w},\boldsymbol{u},\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \beta - \sum_{\ell=1}^{L} \boldsymbol{u}^{\mathrm{T}} \mathcal{Q}_{\ell i} \boldsymbol{w}_\ell \right)^2 \\ + \lambda_1 \sum_{\ell=1}^{L} \|\boldsymbol{w}_\ell\|_2 + \lambda_2 \|\boldsymbol{w}\|_1 + \psi_\ell(\boldsymbol{u}, \lambda_3, \alpha)\,,\end{aligned} \tag{2.10}$$

where L is the number of term groups, respectively $\mathcal{Q}_{\ell i}$ and $\boldsymbol{w}_\ell$ refer to the input data (word frequencies per group) and term weights for group ℓ, λ's and α are the regularisation parameters, and the remaining terms follow the notation of the previous paragraphs. The above optimisation task formulates a combination of a sparse Group LASSO regulariser (for the groups of words) [45, 11] together with Elastic Net's objective function (for the users). Therefore, we are overall regularising three quantities: the term groups (ℓ_2-norm), the members of each group (ℓ_1-norm) and the users (Elastic Net regularisation function). This method is denoted as Bilinear Group LASSO – Elastic Net (**BGLEN**) in the remainder of this script.

2.3 Experiments

The proposed models are evaluated on $\mathrm{C_{uk}}$ and $\mathrm{C_{au}}$ which have been introduced in Section 2.1. We measure predictive performance, compare it to the

performance of several competitive baselines, and provide a qualitative analysis of the parameters learned by the models.

2.3.1 Data preprocessing

Basic preprocessing has been applied on the vocabulary index of C_{uk} and C_{au} aiming to filter out some of the word features and partially reduce the dimensionality of the problem. Stop words and web links were removed in both sets, together with character sequences of length <4 and <3 for C_{uk} and C_{au} respectively.[6] As the vocabulary size of C_{uk} was significantly larger, for this data set we have additionally merged Twitter hashtags (*i.e.* words starting with '#') with their exact non topic word match, where possible (by dropping the '#' when the word existed in the index). After performing the preprocessing routines described above, the vocabulary sizes for C_{uk} and C_{au} were set to 80,976 and 22,917 respectively.

2.3.2 Predictive accuracy

To evaluate the predictive accuracy of our methods, we have chosen to emulate a real-life scenario of voting intention prediction. The evaluation process starts by using a fixed set of polls matching to consecutive time points in the past for training and validating the parameters of each model. Testing is performed on the following δ (unseen) polls of the data set. In the next step of the evaluation process, the training/validation set is increased by merging it with the previously used test set (δ polls), and testing is now performed on the next δ unseen polls. In our experiments, the number of steps in this evaluation process is set to 10 and in each step the size of the test set is set to $\delta = 5$ polls. Hence, each model is tested on 50 unseen and consecutive in time samples. The loss function in our evaluation is the standard Mean Square Error (**MSE**), but to allow a better interpretation of the results, we display its root (**RMSE**) in tables and figures.[7]

The parameters of each model (α_i for BEN and λ_i for BEN and BGL, $i \in \{1, 2\}$) are optimised using a held-out validation set by performing grid search. Note that it may be tempting to adapt the regularisation parameters in each phase of the iterative training loop, however this would change the

[6] Most of the times those character sequences were not valid words. This pattern was different in each language and thus, a different filtering threshold was applied in each data set.

[7] RMSE has the same metric units as the response variable.

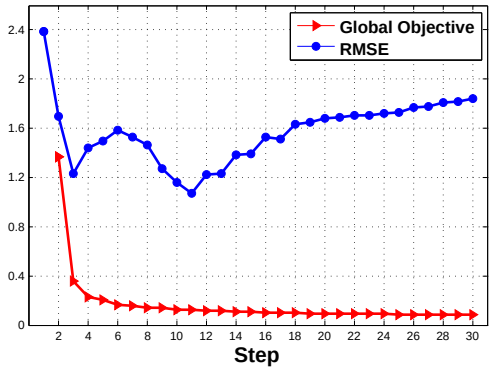


Figure 2.2: Global objective function and RMSE on a test set for BEN in 15 iterations (30 steps) of the model.

global objective (see Eqs. 2.2 and 2.9) and thus convergence will not be guaranteed. A key question is how many iterations of training are required to reach convergence. Figure 2.2 illustrates how the BEN global objective function (Eq. 2.2) converges during this iterative process and the model's performance on an unseen test set. Notice that there is a large performance improvement after the first step (which alone is a linear solver), but overfitting occurs after step 11. Based on this result, for subsequent experiments we run the training process for two iterations (4 steps), and take the best performing model on the held-out validation set.

	CON	**LAB**	**LBD**	$\boldsymbol{\mu}$
$\mathbf{B}_{\boldsymbol{\mu}}$	2.272	1.663	1.136	1.69
$\mathbf{B}_{\mathbf{last}}$	2	2.074	1.095	1.723
LEN	3.845	2.912	2.445	3.067
BEN	1.939	1.644	1.136	1.573
BGL	**1.785**	**1.595**	**1.054**	**1.478**

Table 2.1: UK case study — Average RMSEs representing the error of the inferred voting intention percentage for the 10-step validation process; $\boldsymbol{\mu}$ denotes the mean RMSE across the three political parties for each baseline or inference method.

We compare the performance of our methods with three baselines. The first makes a constant prediction of the mean value of the response variable $\boldsymbol{y}$ in the training set ($\mathbf{B}_{\boldsymbol{\mu}}$); the second predicts the last value of $\boldsymbol{y}$ ($\mathbf{B}_{\mathbf{last}}$); and the third baseline (**LEN**) is a linear regression over the terms using elastic net regularisation. Recalling that each test set is made of 5 polls, $\mathrm{B}_{\mathrm{last}}$ should be considered as a hard baseline to beat[8] given that voting intentions tend

[8]The last response value could be easily included as a feature in the model, and would likely

	SPÖ	ÖVP	FPÖ	GRÜ	μ
$\mathbf{B}_{\mu}$	1.535	1.373	3.3	1.197	1.851
$\mathbf{B}_{\mathbf{last}}$	**1.148**	1.556	**1.639**	1.536	1.47
LEN	1.291	1.286	2.039	**1.152**	1.442
BEN	1.392	1.31	2.89	1.205	1.699
BGL	1.619	**1.005**	1.757	1.374	**1.439**

Table 2.2: Austrian case study — Average RMSEs for the 10-step validation process.

to have a smooth behaviour. Moreover, improving on LEN partly justifies the usefulness of a bilinear approach compared to a linear one.

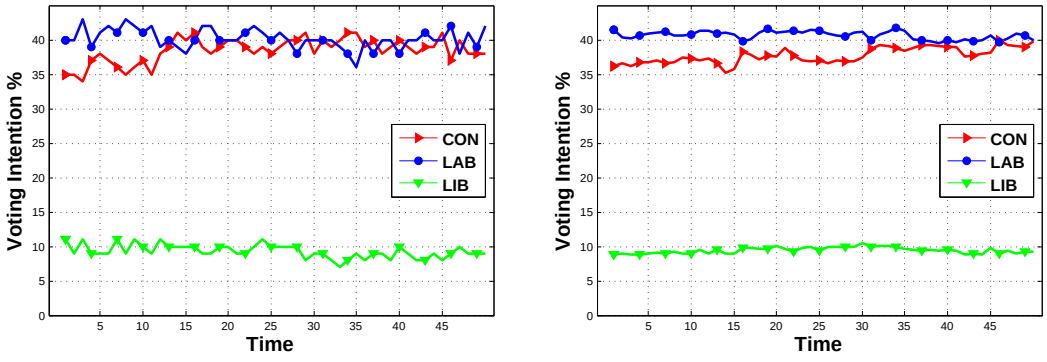
Performance results comparing inferred voting intention percentages and polls for C_{uk} and C_{au} are presented in Tables 2.1 and 2.2 respectively. For the UK case study, both BEN and BGL are able to beat all baselines in average performance across all parties. However in the Austrian case study, LEN performs better that BEN, something that could be justified by the fact that the number of users in C_{au} was small enough (1.1K) and filtering them further had a negative impact. Nevertheless, the difference in performance was rather small (approx. 0.26% error) and the inferences of LEN and BEN followed a very similar pattern ($\bar{\rho} = .94$ with $p < 10^{-10}$).[9] Multi-task learning (BGL) delivered the best inference performance in both case studies, which was on average smaller than 1.48% (RMSE).

Inferences for both BEN and BGL have been plotted on Figures 2.3 and 2.4. They are presented as continuous lines of 50 inferred points (per party) which are created by concatenating the inferences on all test sets.[10] For the UK case study, one may observe that BEN (Fig. 2.3b) cannot register any change – with the exception of one test point – in the leading party fight (CON versus LAB); BGL (Fig. 2.3c) performs much better in that aspect. In the Austrian case study this characteristic becomes more obvious. BEN (Fig. 2.4b) consistently predicts the wrong ranking of ÖVP and FPÖ, whereas BGL (Fig. 2.4c) does much better. Most importantly, a general observation is that BEN's predictions are smooth and do not vary significantly with time. This might be a result of overfitting the model to a single response variable which usually has a smooth behaviour. On the contrary, the multi-task learning property of BGL reduces this type of overfitting providing more statistical evidence for the terms and users and thus, yielding not only a better inference performance, but also a

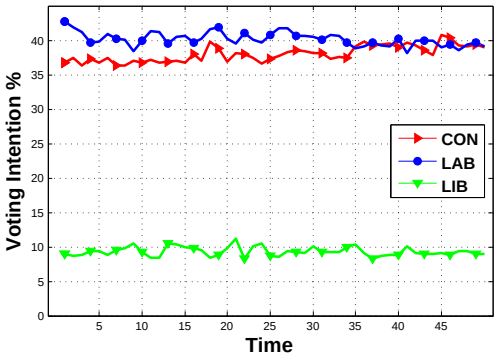
improve predictive performance.

[9] Pearson's linear correlation averaged across the four Austrian parties.

[10] Voting intention polls were plotted separately to allow a better presentation.



(a) Ground Truth (**polls**) **(b) BEN**



(c) BGL

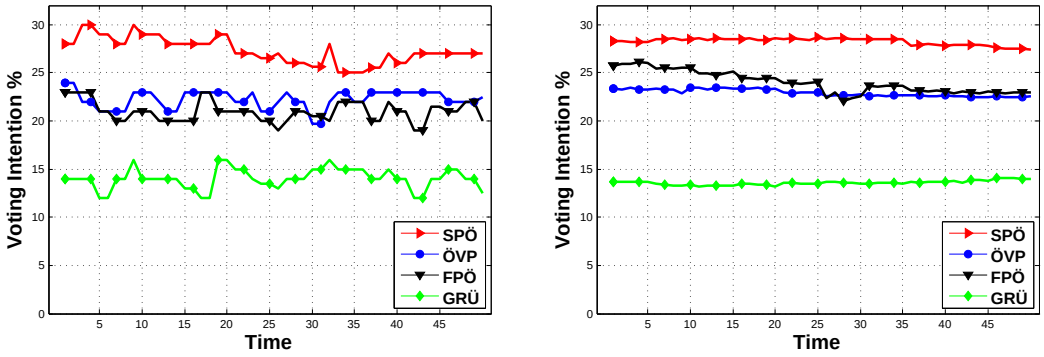
Figure 2.3: UK case study — Voting intention inference results (50 polls, 3 parties). Sub-figure 2.3a is a plot of ground truth as presented in voting intention polls (Fig. 2.1a).

more accurate model.

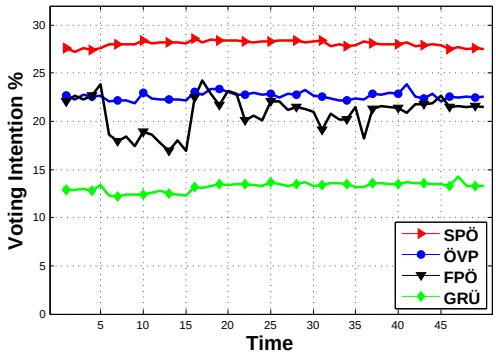
2.3.3 WP2 Integration: Using sentiment word lists

In this section we describe two experiments that demonstrate a way for integrating sentiment features extracted by WP2 to our methods for text regression. In the following experiments we use the MPQA opinion corpus and in particular its subjectivity lexicon [43, 44]. We consider four groups of terms based on all combinations of their polarity (positive or negative) and subjectivity (strong or weak). The number of keywords per group is: 1062 (weak subjectivity, negative polarity), 899 (weak subjectivity, positive polarity), 1299 (strong subjectivity, positive polarity) and 2234 (strong subjectivity, negative polarity).

The experiments follow the same scenario as in the previous sections, *i.e.*



(a) Ground Truth (**polls**) **(b) BEN**



(c) BGL

Figure 2.4: Austrian case study — Voting intention inference results (50 polls, 4 parties). Sub-figure 2.4a is a plot of ground truth as presented in voting intention polls (Fig. 2.1b).

the inference target are voting intention polls; we have only carried out experiments on the UK data set (C_{uk}). The first experiment is the application of BEN on the MPQA terms as whole (without considering any grouping based on polarity or subjectivity), whereas the second is the application of BGLEN on the four groups of sentiment terms. The aim of those experiments is two-fold: a) investigate whether a predictive value is present in those sentiment terms and b) see whether a structured learning function (structure is based on the polarity/subjectivity groups) can improve further this inference value.

The performance results (with the baselines as well as the performance of BEN when applied on the entire Twitter vocabulary for an easy comparison) are presented on Table 2.3. First of all, we see that the enforcement of a structure (based on the sentiment word list properties) enhances the inference performance, that is BGLEN (MPQA) performs better than BEN (MPQA). We also observe that both models are not able to outperform the application of BEN on the entire Twitter lexicon. Still, we believe that by combining the

	CON	LAB	LBD	μ
$\mathbf{B}_{\mu}$	2.272	1.663	1.136	1.69
$\mathbf{B}_{\mathbf{last}}$	2	2.074	1.095	1.723
BEN	**1.939**	1.644	1.136	**1.573**
BEN (MPQA)	2.13	1.713	1.487	1.777
BGLEN (MPQA)	2.238	**1.629**	**1.126**	1.664

Table 2.3: UK case study — Inference performance based on the use of MPQA sentiment lexicon and the corresponding methodologies; comparison with previous results.

	CON	LAB	LBD	μ
$\mathbf{B}_{\mu}$	2.272	1.663	1.136	1.69
$\mathbf{B}_{\mathbf{last}}$	2	2.074	**1.095**	1.723
LEN	3.845	2.912	2.445	3.067
BEN	1.939	1.644	1.136	1.573
EN (IE)	1.969	1.584	1.096	**1.55**
GL (IE)	**1.916**	**1.422**	1.355	1.564

Table 2.4: UK case study — Inference performance based on the use of Wikipedia entities (representing events, people, locations, etc.) and the corresponding methodologies; comparison with previous results.

two lexicons (Twitter vocabulary index or clustered subsets of it as well as groups of sentiment word lists) we can possibly improve the performance of the former; this is something that we will experiment with in the next months.

2.3.4 WP2 Integration: Predictions using entities/events

In this section two additional WP2 integration experiments are presented. We are using a method developed by WP2 (see Deliverable **D2.2.1** and [9]) which aims to extract entities from text by exploiting linked data. In our case, the entities are extracted from the content of tweets and each entity corresponds to a 'term' described in a Wikipedia page. Of course, this information extraction (**IE**) method cannot be 100% accurate and therefore some noise is expected in the extracted features, *i.e.* some Wikipedia pages are going to be incorrect references for the words included in a tweet. However, we assume that our learning algorithm will filter most of the noise (the results from the experimental process seem to confirm that). Note that the Wikipedia-based features

can account for several types of entities including events, people, phenomena, locations, etc.

In those preliminary experiments we do not model users (as we did in all previous experiments); the experimentation in both word and user spaces will be carried out in the near future. Two models for voting intention prediction are trained and tested using the data set of UK tweets (C_{uk}); the only distinction here that the actual data set used in the experiments is a subset of the original one (it only contains a 10% of the original 42K users) and thus, the inference results will not be directly comparable with the original ones. The first model is a pure application of Elastic Net (**EN**), whereas the second one performs multi-task learning and multi-output regression using the Group ℓ_1/ℓ_2 regulariser (**GL**).

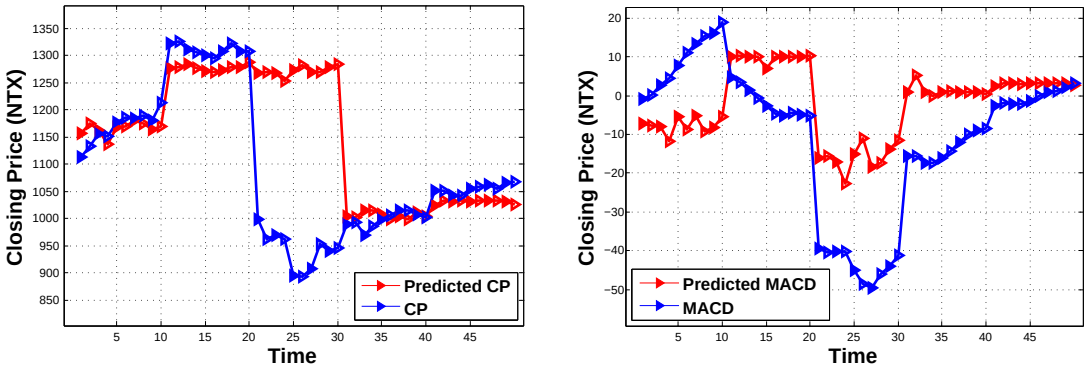
WP2's IE method extracted a set of approx. 250,000 Wikipedia entities from C_{uk}. By filtering the ones that had a small frequency in our data, we ended up with approx. 25,000 entities, which defined the dimensionality of our task. Table 2.4 holds the performance results of EN and GL in comparison to the previously applied schemes on the entire Twitter vocabulary and the considered set of users. Interestingly, the Wikipedia entities perform quite well when used as features; they outperform the baselines as well as BEN (not BGL though, see Table 2.1). Similarly to the previous section, these good performance indicators encourage the application of learning schemes which combine all features in one model.

2.3.5 Preliminary experiments on financial time series

In this section, we describe a set of preliminary experiments which apply BEN in an effort to predict financial indicators. The experiments yield interesting results which firstly indicate that data coming from Social Media do contain valuable information for the domain of finance and secondly demonstrate how the proposed models can be directly applied to other domains (apart from the political one). For the following experiments we are using the data set with tweets from Austria (C_{au}); the response variables (inference targets) are the closing price and MACD[11] of the NTX Index.[12]

The evaluation process is similar to the one used in the previous experiments in that we train on previous values and try to predict the ones in the future, though this time we are using 5 folds where the number of training points is the same for each fold (set equal to 4 months of data) and then try

[11] Moving Average Convergence/Divergence, `http://en.wikipedia.org/wiki/MACD`.
[12] NTX Index, `http://en.indices.cc/indices/details/ntx/facts/`.



(a) Closing Price (NTX) **(b)** MACD (NTX)

Figure 2.5: Prediction of financial indicators (closing price and MACD of NTX Index) based on Twitter data (Austria).

to predict the financial indices in the following 10 days (*i.e.* overall we have 50 test points). Table 2.5 and Figure 2.5 present the inference performance results which are encouraging given that we have not embedded any time series analysis technique so far in our methodology (for example B_{last} could be included as an additional feature or a simple autoregressive model could be incorporated).

	Closing Price	**MACD**
$\mathbf{B_{\mu}}$	160.306	18.286
$\mathbf{B_{last}}$	**87.151**	**16.502**
LEN	151.805	18.853
BEN	149.582	17.345

Table 2.5: Regression performance (RMSE) in predicting financial indicators (Austria).

2.3.6 Qualitative Analysis

In this section, we refer to features that have been selected and weighted as significant by our bilinear learning functions. Based on the weights for the word and the user spaces that we retrieve after the application of BGL in the last step of the evaluation process (see the previous section), we compute a score (weighted sum) for each tweet in our training data sets for both C_{uk} and C_{au}. Table 2.6 shows examples of interesting tweets amongst the top weighted ones (positively as well as negatively) per party. Together with their text

Party	Tweet	Score	Author
CON	PM in friendly chat with top EU mate, Sweden's Fredrik Reinfeldt, before family photo	1.334	Journalist
	Have Liberal Democrats broken electoral rules? Blog on Labour complaint to cabinet secretary	−0.991	Journalist
LAB	Blog Post Liverpool: City of Radicals Website now Live <*link*> #liverpool #art	1.954	Art Fanzine
	I am so pleased to hear Paul Savage who worked for the Labour group has been Appointed the Marketing manager for the baths hall GREAT NEWS	−0.552	Politician (Labour)
LBD	RT *@user*: Must be awful for TV bosses to keep getting knocked back by all the women they ask to host election night (via *@user*)	0.874	LibDem MP
	Blog Post Liverpool: City of Radicals 2011 – More Details Announced #liverpool #art	−0.521	Art Fanzine
SPÖ	Inflationsrate in Ö. im Juli leicht gesunken: von 2,2 auf 2,1%. Teurer wurde Wohnen, Wasser, Energie.	0.745	Journalist
	Hans Rauscher zu Felix #Baumgartner "A klaner Hitler" <*link*>	−1.711	Journalist
ÖVP	#IchPirat setze mich dafür ein, dass eine große Koalition _mathematisch_ verhindert wird! 1.Geige: #Gruene + #FPOe + #OeVP	4.953	User
	kann das buch "res publica" von johannes #voggenhuber wirklich empfehlen! so zum nachdenken und so... #europa #demokratie	−2.323	User
FPÖ	Neue Kampagne der #Krone zur #Wehrpflicht: "GIB BELLO EINE STIMME!"	7.44	Political satire
	Kampagne der Wiener SPÖ "zum Zusammenleben" spielt Rechtspopulisten in die Hände <*link*>	−3.44	Human Rights
GRÜ	Protestsong gegen die Abschaffung des Bachelor-Studiums Internationale Entwicklung: <*link*> #IEbleibt #unibrennt #uniwut	1.45	Student Union
	Pilz "ich will in dieser Republik weder kriminelle Asylwerber, noch kriminelle orange Politiker" - BZÖ-Abschiebung ok, aber wohin? #amPunkt	−2.172	User

Table 2.6: Examples of tweets with top positive and negative scores per party for both $\mathbf{C}_{\text{uk}}$ and $\mathbf{C}_{\text{au}}$ data sets.

(anonymised for privacy reasons) and scores, we also provide an attribute for the author (if present). In the displayed tweets for the UK study, the only possible outlier is the '*Art Fanzine*'; still, it seems to register a consistent behaviour (positive towards LAB, negative towards LBD) and, of course, hidden, indirect relationships may exist between political opinion and art. The Austrian case study revealed even more interesting tweets since training was conducted on data from a very active pre-election period. For a better interpretation of the presented tweets, it may be useful to know that '*Johannes Voggenhuber*' (who receives a positive comment for his book) and '*Peter Pilz*' (whose comment is questioned) are members of GRÜ, '*Krone*' (or Kronen Zeitung) is the major newspaper in Austria[13] and that FPÖ is labelled as a far right party, something that may cause various reactions from '*Human Rights*' organisations.

[13] "Accused of abusing its near monopoly to manipulate public opinion in Austria", Wikipedia, 19/02/2013, `http://en.wikipedia.org/wiki/Kronen_Zeitung`.

2.4 Related Work

The topic of political opinion mining from Social Media has been the focus of various recent research works. Several papers have presented methods that aim to predict the result of an election [42, 6] or to model voting intention and other kinds of socio-political polls [31, 19]. Their common feature is a methodology based on a meta-analysis of word frequencies using off-the-shelf sentiment tools such as LIWC [33] or Senti-WordNet [10]. Moreover, the proposed techniques tend to incorporate posting volume figures as well as hand-crafted lists of words relevant to the task (*e.g.*, names of politicians or parties) in order to filter the content successfully.

Such papers have been criticised as their methods do not generalise when applied on different data sets. According to the work in [13], the methods presented in [42] and [31] failed to predict the result of US congressional elections in 2009. We disagree with the arguments supporting the statement "you cannot predict elections with Twitter" [12], as many times in the past actual voting intention polls have also failed to predict election outcomes, but we agree that most methods that have been proposed so far were not entirely generic. It is a fact that the majority of sentiment analysis tools are English-specific (or even American English) and, most importantly, political word lists (or ontologies) change in time, per country and per party; hence, generalisable methods should make an effort to limit reliance from such tools.

Furthermore, our work – indirectly – meets the guidelines proposed in [28] as we have developed a framework of "well-defined" algorithms that are "Social Web aware" (since the bilinear approach aims to improve noise filtering) and that have been tested on two evaluation scenarios with distinct characteristics.

Chapter 3

Online Biconvex Learning

3.1 Introduction

The bilinear model described in the previous chapter allows the exploration of textual streams from both the perspective of the words expressed and the people expressing those words. This predictive model is constructed through a supervised learning scheme against arbitrary real valued data streams. Therefore, using this model it is possible to go beyond the identification of words and users which portray general importance, and go towards the identification of important words and users *for a particular task*.

A primary application envisaged for this bilinear model is the analysis of social text streams. Concretely, we define this as the analysis of textual corpora to which multiple users are contributing incrementally over time. Within this domain the bilinear model could be used to identify leading players for a given task, important keywords of a given topic or domain, various other summarisation of social event activities via ranking as well as the identification of novel words or users associated with particular events occurring within a given domain.

The Web Social Media platforms which house such activity are inherently dynamic, readily expressed as asynchronous streams of novel textual artefacts often generated by extremely large and diverse user bases. This is in direct contrast to traditional machine learning datasets usually comprised of static data sources with fixed users and vocabularies. This challenging streaming context means novel terms and novel users are a rule rather than an exception. In its current state the biconvex learning algorithm outlined in the previous chapter cannot analyse changing vocabularies, changing user sets or indeed incorporate novel information once the model is constructed. One way to

extend the model to deal with streaming sources is the construction of fixed data windows from the stream. This approach would necessitate a re-training of the model from scratch against these changing data windows.

A more elegant approach is the application of an online, one pass learning scheme to construct the bilinear model directly from the stream. An appropriate technique must be able to accurately construct the model incrementally, both in terms of data but also in terms of model dimensions (i.e. users and words). Furthermore, the technique chosen must be able to selectively forget data as it becomes irrelevant, allowing the model to efficiently change over time and therefore capture non-stationary components of the underlying information space.

Over the past 5 years such online learning techniques have been given a great deal of attention. As is often the case, the interest is born out of a novel problem and a promising family of solutions which become relevant as they seem to address that problem. In the case of online learning, its increased interest can be attributed to a mixture of: increased availability of large datasets including large scale streaming data sources; and the improved performance of techniques which have the capacity and are simple enough to use large quantities of these novel data sources. It has been shown that, in practice, the size of the dataset is the more important factor in terms of the performance of machine learning models [14]. Studies have shown that a simple models trained over an enormous dataset will outperform a sophisticated model trained on a smaller dataset. Interesting examples of this exist in both the text analysis domain [4], as well as image retrieval [41].

This desire to handle extremely large datasets has lead to the exploration of online techniques which can create a reasonable model in an iterative, one-item at a time setting. Stochastic gradient descent is a common approach to learning in an online manner [27, 8]. For example, Lin and Kolcz [24] present Twitter's solution to large scale machine learning including an online stochastic gradient descent model learner [8] which trains a two class logistic regression based classifier. In this section we present and explore our solution to training a bilinear model in an online fashion which is in fact a biconvex stochastic gradient descent scheme.

In the rest of this chapter we describe the approach we have taken to learn the bilinear model in an online, one pass, setting. We then describe a series of experiments in which we demonstrate the online learner's ability improve its predictions over time, maintain sublinear regret increase[1] and an ability to forget historic data and therefore deal with non-stationary distributions more

[1] against a batch learning approach

effectively than a batch learner with access to the whole stream simultaneously.

3.2 Online Biconvex Learning

To learn the bilinear model described in the previous chapter, we have devised an online biconvex learning scheme which includes an ability to learn the weightings of novel words, novel users, individual or batches of novel data and also the integration of various regularisers including the l1,l2 regulariser mentioned in the previous chapter, all working in an online, one pass setting.

Our technique extends stochastic gradient descent methods with proximal updates for regularisation to work in a biconvex context. Given a training pair of features x_i and the real valued tasks to be predicted y_i, Algorithm 1 addresses the learning problem outlined in Equation 2.9 and updates the bilinear model. For each training pair, the algorithm alternate between (sub-)gradient steps with respect to some loss function and proximal steps with respect to some regulariser. These steps are repeated alternately for the model's word and user weightings and the model's bias. More concretely, we update the user weights keeping the words fixed, followed by updating the words keeping the users fixed, and finally update the bias keeping both users and words fixed. This update procedure is then iterated for a single training pair until some stopping condition is reached. The process is then repeated when a novel training pair arrives.

For comparison, it might be useful to consider this algorithm to be a stochastic gradient descent implementation of the biconvex update scheme suggested in section 2.2.1. The key difference being that the batch scheme optimises both $\boldsymbol{w}$ and $\boldsymbol{u}$ completely across all data items available, where this scheme is explicitly formulated to update both $\boldsymbol{w}$ and $\boldsymbol{u}$ for each item of data in isolation.

3.2.1 The Online Learning Algorithm

The first input to Algorithm 1 is a stream of training pairs $\{\langle x_1, y_1\rangle, ..., \langle x_n, y_n\rangle\}$ where x_i is a matrix whose columns are words and whose rows are users and y_i is a vector containing the values whose columns are the real valued results of a number of tasks being predicted and modelled. The concatenation of all x_i instances across the whole of this stream (or some partial subwindow) would form the $\mathcal{Q}$ tensor described in Section 2.2.1. The second input of the algo-

Algorithm 1 Online Biconvex Proximal Gradient Algorithm

$\mathcal{Q} \leftarrow \{\langle x_1, y_1\rangle, ..., \langle x_n, y_n\rangle\}$ ▷ A potentially unbounded set of training pairs
$\boldsymbol{\theta} \leftarrow \{\eta_0, \lambda_w, \lambda_u\}$ ▷ Gradient steps, proximal parameters etc.
procedure OnlineBiconvexLearner($\mathcal{Q}$,$\boldsymbol{\theta}$)
    initialise $\boldsymbol{w}$ ▷ Initialise word weights
    initialise $\boldsymbol{u}$ ▷ Initialise user weights
    initialise β ▷ Initialise Bias
    while $\mathcal{Q}$ hasItem **do**
        while !StoppingCondition($\boldsymbol{\theta}$;$\boldsymbol{w}, \boldsymbol{u}, \beta$) **do**
            $\mathcal{D}_i \leftarrow x_i\boldsymbol{w} + \beta$ ▷ The user-scores matrix
            $\boldsymbol{u} \leftarrow \boldsymbol{u} - \Delta L(\boldsymbol{\theta}; \mathcal{D}_i, \boldsymbol{u}, y_i)$ ▷ User Gradient Step
            $\boldsymbol{u} = prox(\boldsymbol{u})$ ▷ User Proximal Step
            $\mathcal{V}^i \leftarrow \boldsymbol{u}^T x^i + \beta$ ▷ The word-scores matrix
            $\boldsymbol{w} \leftarrow \boldsymbol{w} - \Delta L(\boldsymbol{\theta}; \mathcal{V}_i, \boldsymbol{w}, y_i)$ ▷ Word Gradient Step
            $\boldsymbol{w} = prox(\boldsymbol{w})$ ▷ Word Proximal Step
            $\mathcal{B}_i \leftarrow \boldsymbol{u}^{\mathrm{T}} x_i \boldsymbol{w} + \beta$ ▷ Current estimate with bias
            $\beta \leftarrow \beta - \Delta L(\boldsymbol{\theta}; \mathbb{I}, \mathcal{B}_i, y_i)$ ▷ Bias Gradient Update
        end while
    end while
end procedure

rithm are the various parameters which govern initialisation, learning rates and regulariser parameters etc. The parameters and their purpose will be highlighted as they are used in the rest of this algorithm description and in more detail in the implementation details in Section 3.4.

The first stage of the algorithm, before any data is seen, is the initialisation of the weightings matricies and the bias. Experimentally we found there was a great deal of sensitivity in these initial values given the gradient calculation nature of the stochastic update scheme. Complete 0 initialisation would result in a 0 update in the loss-gradient update. A random initialisation heavily slows the rate of learning. Eventually we found a reasonable setup was to initialise $\boldsymbol{w}$ uniformly with low values in the range of the data input and to keep β and $\boldsymbol{u}$ at 0, allowing the first update to be completely data driven.

Once initialised, the algorithm proceeds on a per training-pair basis. Though we show a monolithic while-loop in Algorithm 1, the actual implementation more closely resembles a processing pipeline where current state of the weightings vectors are held[2] and updated via calls to an update function. The exact implementation is described in further detail in Section 3.4. For

[2] either in memory or serialised on disk

each training pair, an iterative bi-convex optimisation scheme is followed. The stages of the bi-convex optimisation can be further broken down into a loss function sub-gradient steps followed by a proximal regulariser sub-gradient steps adjustment for each of the words, users and bias matricies.

Though any loss function with a calculable sub-gradient can be used, for the sake of an example we show here the procedure for a L2 loss function. The loss function for a single training pair $\langle x_i, y_i \rangle$ is:

$$L(\boldsymbol{\theta}; x_i, \{\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{\beta}\}, y_i) = \sum_{t=1}^{\tau} \left(\boldsymbol{u}_t^{\mathrm{T}} x_i \boldsymbol{w}_t + \beta_t - y_{ti}\right)^2 \tag{3.1}$$

In the case of calculating the update on $\boldsymbol{u}$, firstly $\mathcal{D}_i$ is calculated, thus keeping $\boldsymbol{w}$ and $\boldsymbol{\beta}$ fixed. The gradient of this loss function is used to update $\boldsymbol{u}$:

$$\begin{aligned} \mathcal{D}_i &= x_i \boldsymbol{w} \\ L(\boldsymbol{\theta}; \mathcal{D}_i, \{\boldsymbol{u}, \boldsymbol{\beta}\}, y_i) &= \sum_{t=1}^{\tau} \left(\boldsymbol{u}_t^{\mathrm{T}} \mathcal{D}_{it} + \beta - y_{ti}\right)^2 \\ \Delta L(\boldsymbol{\theta}; x_i, \mathcal{D}_i, y_i) &= \sum_{t=1}^{\tau} 2\mathcal{D}_{it} \left(\boldsymbol{u}_t^{\mathrm{T}} \mathcal{D}_{it} + \beta - y_{ti}\right) \end{aligned} \tag{3.2}$$

It can be shown that this cost function is convex which means following its gradient is guaranteed to lead to a global minima. The amount which this gradient is followed is one of the parameters held in $\boldsymbol{\theta}$. A similar loss function gradient update can be calculated for $\boldsymbol{w}$ and β with slight adjustment for the calculation of the β gradient which takes the form:

$$\begin{aligned} \mathcal{B}_i &= \boldsymbol{u}^{\mathrm{T}} x_i \boldsymbol{w} + \beta \\ L(\boldsymbol{\theta}; \mathcal{B}_i, \mathbb{I}, y_i) &= \sum_{t=1}^{\tau} \left(\mathcal{B}_{it} - y_{ti}\right)^2 \\ \Delta L(\boldsymbol{\theta}; \mathcal{B}_i, \mathbb{I}, y_i) &= \sum_{t=1}^{\tau} 2 \left(\mathcal{B}_{it} - y_{ti}\right) \end{aligned} \tag{3.3}$$

As with the loss function, any regulariser with a calculable sub-gradient proximal update step can be used in our procedure. The proximal update step to achieve a multi-task ℓ_1/ℓ_2 regulariser discussed in Section 2.2.3 for the m^{th} word or user is:

Algorithm 2 Dampening of Weightings

1: $\boldsymbol{w}$ ▷ Current value of word-task weightings
2: $\boldsymbol{u}$ ▷ Current value of user-task weightings
3: β ▷ Current value of bias
4: γ ▷ Some dampening factor
5: **procedure** DampenWeightings($\boldsymbol{w}, \boldsymbol{u}, \beta, \gamma$)
6: $\quad \boldsymbol{w} \leftarrow \boldsymbol{w}(1-\gamma)$ ▷ Dampen word weightings
7: $\quad \boldsymbol{u} \leftarrow \boldsymbol{u}(1-\gamma)$ ▷ Dampen user weightings
8: $\quad \beta \leftarrow \beta(1-\gamma)$ ▷ Dampen bias
9: **end procedure**

$$\text{prox}_{\theta_d}(\boldsymbol{w})_m = \begin{cases} 0 & \text{if } \|\boldsymbol{w}\|_2 \leq d_m \\ \frac{\|\boldsymbol{w}\|_2 - d_m}{\|\boldsymbol{w}\|_2} \boldsymbol{w}_m & \text{otherwise.} \end{cases} \tag{3.4}$$

Whose derivation and background can be found in Martins [27]'s work. As described in the previous chapter, this regulariser favours sparsity between rows of the weighting matrices (i.e. sparsity between users and words) but density between columns of a non-zero row (i.e. tasks). For example, this means that when a word is seen to be useful for the prediction of one task, it is expected to be important for the prediction of other tasks. This assumption works well when the various tasks modelled in the y_i vector are thought to share a domain. Alternative regularisers can also be supported many of whose implementations already exist in the SPAMS toolkit[3].

The updates to each weightings matrix is iterated a number of times for each training pair depending on a given stopping criteria. Experimentally we have found that stopping after a certain number of minimum iterations up to a certain maximum is prudent. Furthermore, we can stop early if we notice that the values of the weightings matrices have become stable and therefore don't change very much relatively within a single update step. Other approaches could be the tuning of the weightings matrices for the optimal calculation of some left out validation set. Such stopping criteria will be investigated in future work.

3.2.2 Non-Stationary Distributions

In Algorithm 1, Line 7, after a new training pair is received but before any alteration is made to the model weightings, it is possible to inject the actions of

[3] `http://spams-devel.gforge.inria.fr`

Algorithm 2 and dampen the past model weightings. The main purpose of this algorithm is to deal with the potentially non-stationary nature of the training pairs being consumed during model construction. Both the word-user dependent matrix and the task value independent matrix are likely to have means, variances and co-variances that change over time. Though non-stationary distributions can occur for a variety of reasons, one way in which non-stationary distributions are generated is through a random walks like process wherein recent values are better indicators of future values than historic values.

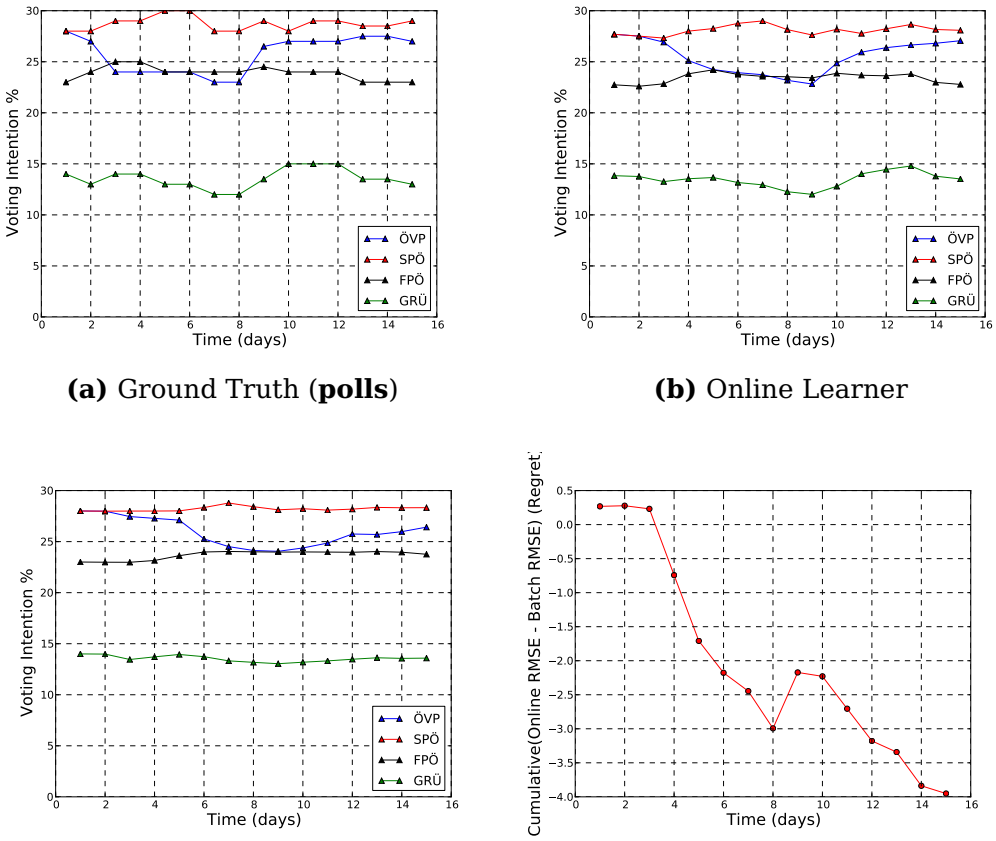
For this reason, upon the addition of novel data, old weightings could be dampened by some proportion. This would result in the effect of early training pairs on the model's weightings disappearing entirely as novel items are added. The amount of dampening might control exactly how quickly the past is forgotten. If the distribution of independent and dependent variables is truly non-stationary, then predictions of future events should improve as a consequence of forgetting unhelpful old pairs and paying more attention to recent pairs. Furthermore, as the pairs being used to predict some future event occur closer in time to that future event's time, more aggressive dampening should force the past to be forgotten more rapidly and therefore improve predictions.

3.3 Experiments

In this section we outline experiments and present results testing the online biconvex learner used to train a bilinear model. The first experiment explores the online learner's performance as compared to a batch learner. The second experiment explores dampening and its effect on performance given the potentially non-stationary nature of the tweet and political poll problem space being modelled. All experiments mentioned here use datasets discussed in Section 2.1 and some experiments outlined in Section 2.3 in an online context.

3.3.1 Regret analysis

In this experiment we aim to demonstrate the capabilities of the online learner by comparing it to the batch learner in the context of a common problem born of streaming data. Our experiment uses the Austrian political intention data set described in Section 2.1 and proceeds as follows: Given some initial model, the x_i component of a training pair is used to predict the pair's y_i component. An error is calculated between this prediction and the correct value in the form of the Root Mean Sum Square Error (RMSE). Once complete, the $\langle x_i, y_i \rangle$



(a) Ground Truth (**polls**)

(b) Online Learner

(c) Batch Learner

(d) Overall regret across days

Figure 3.1: Stream Learning case study for Austrian data

pair is used to continue training the model. This experiment mirrors a situation wherein novel information is constantly available, and perhaps novel independent variables (the x_i components) are made available before the correct dependent variable (the y_i components). In such a situation it is often desirable to predict the likely dependent variable. We perform this experiment with our online learner as well as the batch learner. The batch learner simulates this scenario by being trained on N days to predict the $N+1^{\text{th}}$ day. This is the same data setting as the online learner which is instead handed the first N days before being handed the $N+1^{\text{th}}$ which it uses to perform a prediction, calculate RMSE, and then update the model ready for the next day. By calculating the difference between the batch learner's RMSE and the online learner's RMSE per day, we can calculate a cumulative measure of *regret* [38].

In this experiment we initialise the β of both the batch and the online learner as the first day's y_i value. In both the batch and online learners, the initial values of the $\boldsymbol{w}$ and $\boldsymbol{u}$ weighting matrices are set to a uniform 0.1 and

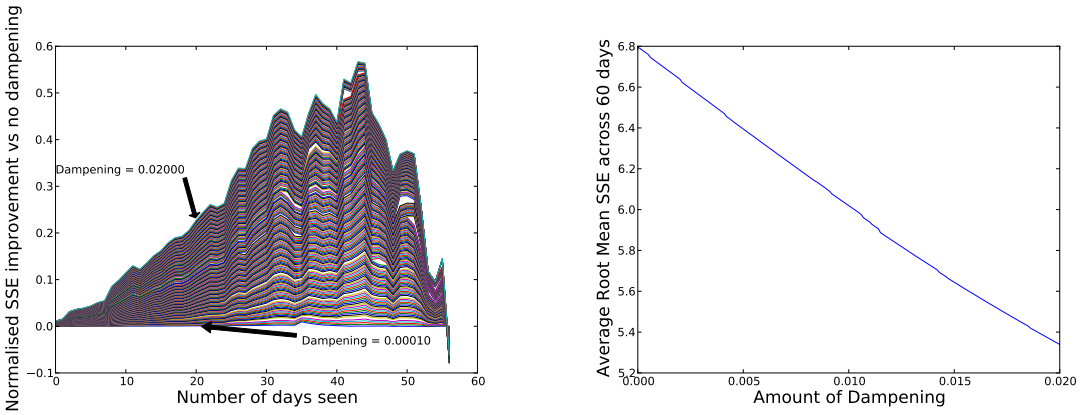
0 respectively. The other parameters used for learners can be seen in Appendix B.1. Figure 3.1 shows the results of this experiment. Figure 3.1a shows the ground truth, 3.1b shows the results of the online learner, 3.1c shows the batch learner's predictions and 3.1d shows the regret between the online and batch learners.

We start by looking at the overall shape of the voting intention predictions where we note that the online learner is correctly updating the bilinear model and thus follows ground truth voting intentions visibly well. Key characteristics visible in the ground truth Figure 3.1a are reflected by the model learnt using the online learner in Figure 3.1b such as the overall ordering of all 4 parties. The online learner also successfully predicts the notable event between FPÖ and ÖVP during days 2 and 8 where there was a swapping of voting intention order. In fact, the online learner was able to capture this switch along with the general shape of the other curves more effectively than the batch learner and therefore achieved lower RMSE scores.

This lower RMSE score can be seen in the form of the negative values in the regret between the batch and online learner shown in 3.1d. Negative regret is generally impossible in regret analysis where multi-pass online algorithms are often expected to perform as well or worse than their batch counterparts when faced with a large static dataset [38]. However, the potential non-stationary nature of this voting intention prediction problem is evidently better handled by an online learner. In this experiment, the online learner was dampened (by 0.02 per day in this experiment) and therefore forgets uninformative old information, favouring newer information. Assuming some variety of random walk is the underlying mechanism for the problem of voter intention, novel training pairs are of higher value for prediction of future events than older training pairs. When compared to the batch learner, for which no special measures have been taken to window the data and thus forget the past, it can be understood that the past might weight the model incorrectly, resulting in the negative regret measured. Another explanation might be inappropriate experimental parameters. As will be discussed in more detail in Section 3.4, the parameters of both learners are numerous and with some adjustment the batch learner's performance on this task might be improved.

3.3.2 Non stationary components

To investigate the premise of dealing with non-stationary data by dampening past models (outlined in Section 3.2.2), we aim to investigate whether our political poll prediction tasks display non-stationary properties. Mimicking the



(a) The difference in Root Mean SSE to no dampening against days)

(b) The average Root Mean SSE across all days against dampening

Figure 3.2: The effect of daily weightings matrix dampening on the root Mean SSE of predicting days 63 through 68 using days 0 to 62.

experiments in Section 2.3 we use days 0 to 62 of the austrian political intention dataset (see Section 2.1) to predict the voting intention outcome of days 63 to 68. Each day is processed by the model sequentially, and after each day an attempt is made to perform the prediction task; the root mean SSE of predicting days 63 to 68 is then measured after each day is consumed. In Figure 3.2a we see the difference in Root Mean SSE when applying a dampening between 0.0001 and 0.02 as compared to no dampening after each day is processed. In Figure 3.2b we show the average absolute Root Mean SSE across all 63 training days for each level of dampening. The first figure shows that as dampening increases, the improved RMSE score beyond the un-dampened model increases as the days processed approach the days to be predicted. This result is evidence for the political intentions distribution being a non-stationary distribution as the results would be expected if the distant-past was a worse predictor of the test-time period than the near-past.

These results show that an effective way of improving the bilinear model's predictive ability in such a non-stationary system is to dampen past weightings. Furthermore we show that, in this particular dataset, higher levels of dampening are preferential, though we expect there to be some limit. This is because if dampening were set to 1 (i.e. all past weightings completely forgotten as training pairs were consumed), then weightings would be incorrectly set to 0 and the model learning would collapse. The appropriate amount of weighting is likely to be affected by various factors including the rate of change of underlying random walk, the rate at which data is added to the model, the optimal or desired window of influence and so on. The exact effect

of this kind of dampening and how it can be wielded to control the portions of past data which effect the model will be addressed in future work.

3.4 The Tools

After initial prototype versions were written in matlab, the learning schemes described in Chapters 3 and 2 were implemented as reusable libraries in both Python and Java. The projects which contain these implementations have been made available as open source projects under the BSD licence, both in java[4] and python[5]. The python project implements both the batch version of the learning described in Section 2[6] while the java implementation currently supports the streaming implementation. Both have been written such that they can be easily used within a streaming context with sequential data updates, the batch learner achieves this by holding previously seen data in memory.

In the rest of this section we describe the structure of the learner libraries including the manner in which they should be used, design decisions and the various parameters and options controlling the behaviour of both libraries support.

A common theme between the implementations below is a focus on the separation of concerns. Learners understand and hold all the functionality of the actual learning procedure, but more general components such as the initialisation of parameters, the loss functions used, the procedure for gradient step update etc. are handled by separate modules which can be specified as parameters. Implementations mentioned here strive for flexibility, allowing the easy side by side comparison of novel replacements for all proposed techniques as they arise.

3.4.1 Batch Implementation: Python

Batch Biconvex Learner

The main class in our python batch biconvex learner implementation is the module `bivariate.learner.batchbivariate`. The main class in this module is an `bivariate.learner.onlinelearner` whose children must provide implementations of `process(X,Y)` and `predict(X)`. The `process` accepts an

[4] `http://github.com/sinjax/trendminer-java`

[5] `http://github.com/sinjax/trendminer-python`

[6] A streaming version was originally written in python but must be re-written for release

individual $X_i \in \mathbb{R}^{p \times m}$ where p is the number of users and m is the number of words and $Y_i \in \mathbb{R}^{1 \times \tau}$ a row of responses for each τ task. `process` also accepts a n long list of X_i instances and a matrix $Y \in \mathbb{R}^{n \times \tau}$. In both cases the input is dealt with cumulatively with previous inputs and must therefore match it's dimensions in terms of users and words. In future implementations we aim to support adaptability with respect to users and words as the streaming implementation does in the next section. Once the model parameters of a given learner are trained, the `predict(X)` function may be called, thus generating a prediction for a given X.

The batch python version of the learner is backed by the SPAMS optimisation library. SPAMS (SPArse Modeling Software) provides a set of implementations of the F-ISTA [5] stochastic gradient descent based loss function and proximal update regulariser optimisation scheme. The SPAMS library supports various problem forms; the form which is of direct interest to our purpose is the multi-task style problems. Given a training set $\{\boldsymbol{x}^i, \boldsymbol{y}_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $\boldsymbol{y}_i \in \mathbb{R}^\tau$, τ is the number of tasks, and p is the number of features we are interested in finding $W \in\in \mathbb{R}^{p \times t}$. The formulation of the problem with a square loss function, no bias and arbitrary regulariser can be written as:

$$\underset{W \in \mathbb{R}^{p \times t}}{\arg\min} \frac{1}{2} ||\mathbf{Y} - \mathbf{X}\mathbf{W}||_2^2 + \lambda \psi(\mathbf{W}) \tag{3.5}$$

Where the rows of $\mathbf{X}$ and $\mathbf{Y}$ hold the n training samples. The problem shown in Equation 3.5 matches closely the two sub-parts of the iterative biconvex optimisation scheme described in Section 2.2.1, namely Equations 2.5 and 2.7 and more importantly the expansion to the independent parameter form found in Equation 2.8. However, before SPAMS can be used, certain discrepancies must be addressed, namely the number of rows in the task-expanded term-weighting and task-expanded user-weighting matrices $\mathcal{D}'$ and $\mathcal{V}'$ and how they relate to the X matrix expected by SPAMS in Equation 3.5.

We re-iterate that $\mathcal{D}' \in \mathbb{R}^{\tau n \times p}$ and is created as a result of stacking $\mathcal{D}_t \in \mathbb{R}^{\tau n \times p}$ where $t \in \{1, \ldots, \tau\}$. The issue which arises can be seen when comparing the number of rows in $\mathcal{D}'$, equalling τn, and the number of rows in the SPAMS formulation $\mathbf{X}\mathbf{W}$ and therefore $\mathbf{Y}$, equalling n. To deal with this discrepancy the values held in each row of y_i in equation 2.8, i.e. the response for each task for a given training pair, can be placed in the matrix $\mathbf{Y}'$ which has the shape $\mathbf{Y}' \in \mathbb{R}^{\tau n \times t}$. We construct $\mathbf{Y}'$ as $\mathbf{Y}' = [\mathbf{Y}_1, \ldots, \mathbf{Y}_\tau]^{\mathrm{T}}$ where $\mathbf{Y}_i$ is a matrix whose shape is $\mathbf{Y}_i \in \mathbb{R}^{n \times \tau}$ and whose i^{th} column contains the response of the i^{th} task of all n samples.

Each $\mathbf{Y}_i$ matrix corresponds with with a $\mathcal{D}_i$ matrix stacked inside $\mathcal{D}'$. De-

constructing the calculation $\mathcal{D}_i\mathbf{U}$ we note that the column created by the multiplication of $\mathcal{D}_i$ with the i^{th} column of $\mathbf{U}$ correctly corresponds with the non-empty i^{th} column of $\mathbf{Y}_i$. All other columns must be ignored by the optimisation process as they correspond to nonsensical pairings of tasks[7] . This is achieved in SPAMS by setting non-i columns of $\mathbf{Y}_i$ to *NaN*. More details of this expansion procedure can be seen in the batch python implementation of our biconvex learner using SPAMS.

Beyond this core consideration, our implementation of the batch biconvex learner includes various options and parameters which could be adjusted for a given task. These include all options which can be passed to the underlying SPAMS optimiser[8]. Beyond these we also allow for the various parameters unique to our biconvex optimisation scheme, including various parameters for controlling the iterative biconvex procedure. These are described in detail both in the code and in Appendix A.1

Other Library Features

Beyond the learner, the python implementation provides various `bivariate.generator` instances, each providing various methods with which to generate data which can feed learners as well as `bivariate.evaluators` which given a learner perform a given evaluation against a test set. Currently the *SSE*, *MSSE* and *RMSSE* evaluators are implemented. Beyond this functionality for building learning systems, we also provide the various experiments used to generate the experimental results found in the previous section. These can be found along with documentation in `bivariate.experiment`.

3.4.2 Online Implementation: Java

With respect to the seperation of regularisers and loss functions, our online biconvex learner implemented in java follows a similar strategy to SPAMS. Loss functions and regularisers are interacted with generically and interchangeably, both for the evaluation of gradient and proximal updates, as well as for the calculation of the cost function by various evaluators. The online learner is a part of the OpenIMAJ [15] multimedia library, more specifically a TrendMiner branch of OpenIMAJ, in the machine learning subproject[9].

[7] e.g. the 2^{nd} row of $\mathcal{D}_i$ and the 1^{st} column of $\mathbf{U}^{\text{T}}$ would represent the 2^{nd} task's user-weights weighted by the 1^{st} task's user parameter, a meaningless calculation.

[8] Those are extensively documented here: `http://bit.ly/Zi2TOt`

[9] `https://github.com/sinjax/trendminer-java`

Simple Biconvex Learner

The first and most simple Online learning biconvex learner in our Java implementation is `org.openimaj.ml.linear.learner.BilinearSparseOnlineLearner` which is an implementation of the `OnlineLearner` interface in the same package, therefore providing implementations of a `process(X,Y)` and `predict(X)` as in the python batch implementation. While the python implementation accepts as input either a single training pair or a batch, this Java implementation accepts only single training pairs at a time. Extensions which we discuss below go beyond this limiting factor.

When the first item is seen, the W, U and optionally the β bias are initialised. Unlike the python implementations the initialisations of these matrices can be controlled separately. Currently the W is initialised to a uniform 0.1, and the β bias and the U value zeros. However, any initialisation strategy which is an extension of `org.openimaj.ml.linear.learner.init.InitStrategy` may be used. Currently various sparse and non sparse init strategies are provided.

As explained with the python implementation, the value of $\mathbf{Y}'$ is constructed from Y_i. With the limitation of one item at a time in this basic implementation, the size of $\mathbf{Y}' \in \mathbb{R}^{\tau \times \tau}$. In the iterative phase of the online learner (see Section 3.2) $\mathcal{D}_i$ and $\mathcal{V}_i$ are calculated for the individual X_i instance handed to the process method. At this point, the gradient of the loss function (extracted from the parameters) is handed the value of $\mathbf{Y}'$, the current value of the parameter being calculated (either W or U) and finally the value of $\mathcal{D}_i$ or $\mathcal{V}_i$. Any loss function implemented against the `org.openimaj.ml.linear.learner.loss.LossFunction` may be used meaning any function with a calculable sub-gradient. Currently only `SquareLossFunction` is provided. The loss for either W or U is added and the paramter is regularised. Again any regulariser implemented against `org.openimaj.ml.linear.learner.regul.Regulariser` which can be any regulariser which can provide a proximal update given a parameter matrix and a λ regularisation paramter. Currently the `L1Regulariser` and the group-task `L1L2Regulariser` are provided. In fact, any regulariser which is expressible as a proximal update can be implemented and used as part of our learning procedure. In future releases we plan to provide a regulariser which understands nested groups and performs the appropriate proximal update.

As with the python implementation this iterative phases proceed for each training pair until some stopping condition is reached. The various parameters of the stopping condition as well as the parameters which control the stages

```
X_i = {
        "steve": {
                "cheese": 1,
                "is": 1,
                "really": 10,
                "nice": 1
        },
        "alan": {
                "ok": 1
        }
}
Y_i = {
        "cheesecompany": 10.4,
        "othercompany":3.4
}
```

Figure 3.3: Example input as JSON to the process function of the adaptive biconvex learner

described above are discussed in more detail in Appendix A.2

Adaptive Biconvex Learner

Another online learner available is the `IncrementalBilinearSparseOnlineLearner`. This learner aims to not only deal with the adaption of a model with the appearance of novel data, but is adaptive with respect to novel words and users. It's `process` function consumes as X_i instances `Map<String,Map<String,Double>>`, a map of users to maps of words to counts. The function consumes as Y_i a `Map<String,Double>`, namely a map of tasks to values. An example of such an input presented as JSON can be seen in Figure 3.3.

This adaptive implementation is backed by a simple `BilinearSparseOnlineLearner`. When a new training pair is received it is firstly checked whether there are any novel words and users in the X_i component of the pair. For each novel word and user, the underlying learner's W and U matrices are updated with empty rows, initialised using the provided initialiser. Alternatively, a special initialiser can be specified which initialises novel word or user parameter rows for each task seen so far to some value aggregated from the parameter of values seen so far. Such flexibility has had little experimentation but may provide for more flexibility in future iterations

of the library.

The index of the new rows added for novel words and users are held by the adaptive learner such that each word and user seen is associated with a unique index. Once this check is made, a new X_i matrix is constructed which is of the dimensions of the number of words and users seen so far, along with a Y_i matrix constructed in a similar way. The two matrices are populated sparsely with the values of the X_i and Y_i maps and the constructed matrices are passed to the underlying online bilinear learner instance. In this way, the weights of novel words and users are initialised and learnt as they arrive from the stream, but when a word or user has been seen before it will correctly contribute towards the value of the parameter matching that word or user.

Other Library Features

As with the python batch implementation, the Java library provides various `org.openimaj.ml.linear.data.DataGenerator` instances, each providing various methods with which to generate data which can feed learners. These include generators which are a type of random data simulator, primarily used for testing, but also includes generator implementations for the datasets used in the experiments discussed in Section 3.3.Also made available are various `org.openimaj.ml.linear.evaluation.BilinearEvaluator` implementations which, given a learner, perform a given evaluation against a test set. Currently the *SSE*, *MSSE* and *RMSSE* evaluators are implemented. Beyond this functionality for building learning systems, we also provide the various experiments used to generate the experimental results found in the previous section. These can be found along with documentation in `org.openimaj.ml.linear.experiments`.

Chapter 4

Conclusions and Future Work

In this section we outline the major components discussed in this deliverable and outline the direction of future work suggested by our work.

4.1 Conclusions

We have presented a novel method for text regression that exploits both word and user spaces by solving a bilinear optimisation task, and extensions that apply multi-task learning for multi-output inference. Our approaches perform feature selection – hence, noise filtering – on large-scale user-generated inputs automatically, generalise across two languages without manual adaptations and deliver a significant inference performance over strong baselines ($< 1.5\%$ error when predicting polls). Our methodologies have been extended to accommodate outputs from WP2 such as sentiment taxonomies or entities. Moreover, we have provided a preliminary demonstration about their applicability on the financial domain.

We have also presented a learner capable of learning the bilinear text regression model in an online setting. We showed that this learner was capable of achieving predictive performances similar to that of the model learnt in a batch setting. Furthermore, we have demonstrated capability of modelling data originating from a non-stationary distribution.

Finally, we have also delivered a set of open source implementations of both our batch and online bilinear model learners. We discussed the key functionality of these tools as well as discussing how the tools could be extended to allow for novel loss functions, regularisers and how they could be integrated as part of a stream processing pipeline.

4.2 Future Work

In this section we will highlight some key directions of future work which have been suggested and lead logically from the work presented in this deliverable. Firstly, in Appendix C we present an example of more sophisticated online learning schemes which could be used to train our bilinear model, or indeed any other model more quickly than a standard stochastic gradient decent.

On the subject of the online model, we believe an exploration into its application to a financial domain would be an interesting area of research. We have shown in earlier deliverables that the bag of words type representation of tweets during a time interval carries some information that correlates with financial data. We have also worked on online learning algorithms suitable for non-stationary data. In immediate future work we expect to achieve convergence between these two strands. A particular focus will be on integrating work on predictions with volatility in the financial markets, rather than direction of movement of the stock prices.

Another potential avenue for research is the joint modelling of several domains in a single model by applying multi-task learning techniques. For example, knowing that 10-year bond prices (financial domain) characterise the internal economy's status for a country, they should also reflect on the political domain. Hence, combining voting intentions together with bond prices in one model may produce better and more generic solutions as well as interesting socio-political insights.

A second line of exploration will be the link between work carried out in WP2 on extracting ontologies and data driven regularization we have been working on in WP3. In this deliverable we have mostly used the outputs of WP2 as rich features and achieved some interesting results shown in Section 2.3.3. In future work we could leverage the structures suggested by the ontologies of WP2 for a more sophisticated integration of WP2 and WP3. This might involve forcing large parts of the model weights to be zero, based on prior knowledge derived from ontologies via well understood hierarchical regularisation techniques.

Appendices

Appendix A

Biconvex Learner Parameters

A.1 Python Batch Learner

Table A.1 shows a set of extra parameters understood by the batch biconvex learner in the biconvex learning library created by trendminer. These parameters are subject to a great deal of change and this document should be used for advisory purposes only. Please see the code available on the repository[1]

Parameter	Values	Purpose
bivar_it0	$0 \dots N$	The minimum number of iterations of the biconvex scheme which must pass before any other checks of biconvex convergence are made
bivar_max_it	$0 \dots N$	The maximum number of iterations of the biconvex learning scheme
bivar_tol	$0 \dots N$	The minimum amount of loss improvement against some loss function and some validation set which an iteration of the biconvex scheme must achieve. If it is not achieved the biconvex scheme ends
init_strat	*lambda* function	Given a shape, this lambda function must initialise a matrix. This is used to initialise the W and U matrices when the first data pair is processed

[1] `http://github.com/sinjax/trendminer-python`

intercept	*True/False*	*(SPAMS parameter)* If true, an extra column of ones must be added to $\mathcal{V}'$ and the current value of the bias (initialised as 1) must be added as a row to $\mathbf{W_0}$ and $\mathbf{U_0}$ before they are handed to SPAMS, and then removed when SPAMS gives its answers for $\mathbf{W}$ and $\mathbf{U}$. This is the bias.
loss	various, see SPAMS docs	*(SPAMS parameter)* If a "non-missing" loss function is requested, it's "missing" version is used instead. This is to account for the required formulation of $\mathbf{Y}'$.

Table A.1: Parameters for the python biconvex learner (includes some SPAMS parameters also used by the learner)

A.2 Java Online Learner

Table A.2 shows a set of parameters understood by the streaming biconvex learner in the biconvex learning library created by trendminer. These parameters are subject to a great deal of change and this document should be used for advisory purposes only. All parameters used by each learner are documented in an extention of the `org.openimaj.ml.linear.learner.LearningParameters` instance. For example, the simple biconvex learner's paramters can be found in `org.openimaj.ml.linear.learner.BilinearLearnerParamters`. Please see the code available on the repository[2]

Parameter	**Values**	**Purpose**
BIAS	true/false	whether a bias component is added to w and u. Default is false.
SEED	$1, \ldots, N$	The random seed of any randomised components of this learner (usually initialisation). Defaults to -1 (i.e. no seed)
WINITSTRAT	`InitStrategy`	The initialisation strategy for W. Defaults to a SparseRandomInitStrategy with sparcity set to 0.5

[2] `http://github.com/sinjax/trendminer-java`

UINITSTRAT	`InitStrategy`	The initialisation strategy for U. Defaults to a SparseRandomInitStrategy with sparcity set to 0.5
BIAS INIT-STRAT	`InitStrategy`	The initialisation strategy for BIAS. Defaults to a SparseZerosInitStrategy
BICONVEX _MAXITER	$1, \ldots, N$	The maximum number of iterations in the biconvex iterative stage. defaults to 3
BICONVEX _TOL	float < 1	The threshold of the ratio between the $\frac{(\|\|w_{\text{new}}-w_{\text{old}}\|\|_1+\|\|u_{\text{new}}-u_{\text{old}}\|\|_1+\|\|\beta_{\text{new}}-\beta_{\text{old}}\|\|_1)}{(\|\|w_{\text{old}}\|\|_1+\|\|u_{\text{old}}\|\|_1+\|\|\beta_{\text{old}}\|\|_1)}$ i.e. some notion of normalised changed of the paramters. Defaults to 0.01
LAMBDA	float	The parameter of the regulariser, defaults to 0.001
ETA0_U	float	The weighting of the subgradient of U, weighted down each ETASTEPS number of iterations of the biconvex scheme, defaults to 0.05
ETA0_W	float	The weighting of the subgradient of W, weighted down each ETASTEPS number of iterations of the biconvex scheme, defaults to 0.05
ETA0_BIAS	float	The weighting of the subgradient of BIAS, weighted down each ETASTEPS number of iterations of the biconvex scheme, defaults to eta0 (0.05)
LOSS	`LossFunction`	The loss function, defaults to SquareMissingLossFunction
REGUL	`Regulariser`	The regularisation function, defaults to L1L2Regulariser
ETASTEPS	integer	The steps at which point the eta parameter is reduced, defaults to 3
FORCE _SPARCITY	true/false	Should all parameter matricies be held SparseMatrix instances and therefore remain sparse. Forces a copy but could save a lot.
DAMPENING	float < 1	The value of w, u and beta are updated each time data is added s.t. w = w * (1.0 - DAMPENING). The default value is 0

Table A.2: Parameters for the Java online biconvex learner

Appendix B

Online Learning Experiment Parameters

B.1 Regret Experiment Parameters

In Figure B.1 we note the parameters used to achieve the results reported for the experiment in Section 3.3.1. Any parameters not explicitly mentioned take their default values as noted in Appendix A.

B.2 Non-Stationary Experiment Parameters

In Figure B.2 we note the parameters used to achieve the results reported for the experiment in Section 3.3.2. Any parameters not explicitly mentioned take their default values as noted in Appendix A.

```
Streaming Parameters = {
        dampening: 0.02,
        loss: SquareLossFunction,
        biaseta0: 0.5,
        seed: -1,
        regul: L1L2Regulariser,
        biasinitstrat: HardCodedInitStrat // Hard coded to y_0,
        biconvex_maxiter: 10,
        winitstrat: SingleValueInitStrat // Set to 0.1,
        forcesparcity: true,
        uinitstrat: SparseZerosInitStrategy,
        lambda: 0.0010,
        biconvex_tol: 0.01,
        eta0u: 0.02,
        eta0w: 0.02,
        etasteps: 3,
        bias: true
}
Batch Parameters = {
        lambda1: 0.0001,
        bivar_it0: 3,
        bivar_max_it: 10,
        max_it:500, // Maximum number of FISTA iterations
        tol:1e-3, // Tolerence for FISTA
        intercept:True,
}
```

Figure B.1: Parameters used for the online and batch learners in the regret experiments

```
Streaming Parameters = {
        dampening: {0:0.0001:0.02}, // from 0 to 0.02 in steps of
            0.0001
        loss: SquareLossFunction,
        biaseta0: 0.5,
        seed: -1,
        regul: L1L2Regulariser,
        biasinitstrat: HardCodedInitStrat // Hard coded to y_0,
        biconvex_maxiter: 10,
        winitstrat: SingleValueInitStrat // Set to 0.1,
        forcesparcity: true,
        uinitstrat: SparseZerosInitStrategy,
        lambda: 0.0010,
        biconvex_tol: 0.01,
        eta0u: 0.01,
        eta0w: 0.01,
        etasteps: 3,
        bias: true
}
```

Figure B.2: Parameters used for the dampening experiments

Appendix C

Sequential Nonstationarity Estimation

Sequential updates for parameter estimation, when cast in a Bayesian setting, can be used to propagate uncertainties about estimated parameters and inference. In the sequential setting, the Kalman filter for linear Gaussian models, and extensions of it, offer the framework for this propagation. This Appendix summarises work being done in this direction. Algorithms have been developed and tested on synthetic data and some difficulties identified. These are now being applied to financial time series data.

C.1 State Space Models

A state state space model, formulated for sequential estimation of parameter vector $\boldsymbol{\theta}$ from observations $\boldsymbol{y}$ is defined by a dynamical state equation and an observation equation.

$$\begin{aligned} \boldsymbol{\theta}(n) &= f\left(\boldsymbol{\theta}(n-1)\right) + \boldsymbol{u}(n) \\ \boldsymbol{y}(n) &= g\left(\boldsymbol{\theta}(n), \boldsymbol{x}(n)\right) + \boldsymbol{v}(n) \end{aligned}$$

A simplified version of this, where one assumes a random walk model for state dynamics, scalar observations, linear output model and Gaussian noise in the process and observation stages is:

$$\begin{aligned} \boldsymbol{\theta}(n) &= \theta(n-1) + \boldsymbol{u}(n) \\ y(n) &= \boldsymbol{\theta}^t(n)\boldsymbol{x}(n) + v(n) \end{aligned}$$

Sequentially estimating $\boldsymbol{\theta}$ from observations y involves the following steps of the Kalman filter:

$$\begin{aligned}
\boldsymbol{\theta}\,(n|n-1) &= \boldsymbol{\theta}\,(n-1|n-1) \\
\boldsymbol{P}\,(n|n-1) &= P\,(n|n-1) + \boldsymbol{Q} \\
e(n) &= y(n) - \boldsymbol{\theta}^t\,\boldsymbol{x}(n) \\
\boldsymbol{k}(n) &= \frac{P\,(n|n-1)\;\boldsymbol{x}(n)}{R + \boldsymbol{x}^t(n)\boldsymbol{P}(n|n-1)\boldsymbol{x}(n)} \\
\boldsymbol{\theta}\,(n|n) &= \boldsymbol{\theta}\,(n|n-1) + \boldsymbol{k}\,(n)\,e(n) \\
\boldsymbol{P}\,(n|n) &= \boldsymbol{P}\,(n|n-1) - \boldsymbol{k}(n)\boldsymbol{x}^t(n)\boldsymbol{P}\,(n|n-1)
\end{aligned}$$

In the notation above $\boldsymbol{\theta}\,(n|n-1)$ should be read as the estimate of the parameter vector $\boldsymbol{\theta}$ at time point n, using all the data upto time $n-1$. $\boldsymbol{Q}$ and R are the covariance matrix of the process noise and the variance of the scalar observations noise respectively.

The regression model we consider for predicting direction of financial market movements from a bag-of-words representation of tweets is a logistic, defined as

$$\begin{aligned}
y &= g\left(\boldsymbol{\theta}^t\boldsymbol{x}\right), \text{where} \\
g(s) &= \frac{1}{1+\exp(-s)}
\end{aligned}$$

We assume that predictions are made from tweets collected from within a time window (daily, hourly etc), preprocessed to generate a vector of bag-of-words. For the non-linear logistic model, the Kalman filter estimation is no longer optimal and a number of methods for approximate propagation of estimates and their uncertainties exist. The extended Kalman filter linearizes the observations aboout the operating point and uses gradients in its re-estimations. The linearization, and subsequent update can be iterated till convergence to derive a recursive update [30] for the parameters of the logistic and the error covariance matrix (uncertainty) on it:

$$\begin{aligned}
\left[\boldsymbol{P}(n|n)\right]^{-1} &= \left[\boldsymbol{P}(n|n-1)\right]^{-1} + g\,(1-g)\,\boldsymbol{x}(n)\boldsymbol{x}^t(n) \\
\boldsymbol{\theta}(n|n) &= \boldsymbol{\theta}(n|n-1) + (y(n)-g)\,\boldsymbol{P}\,[n|n-1]\,\boldsymbol{x}(n)
\end{aligned}$$

These update equations can also be derived directly from a Bayesian perspective as shown by [39].

In the above formulation, the denominator of the update term in the Kalman filter, *i.e.* $R + \boldsymbol{x}^t(n)\boldsymbol{P}(n|n-1)\boldsymbol{x}(n)$, is known as the innovation term and contains the new information in data $\{\boldsymbol{x}(n), y(n)\}$. This has is an indicator

of the degree of novelty in the new data point at time n and closely relates to what is known as the *evidence* term in Bayesian inference literature [26], [18]. Our expectation in **WP3** is that an estimate of this quantity can be used as a robust detector of nonstationarity.

The difficulty in directly implementing the above formulation is the relative dimensions of the data setting. When the covariates $\boldsymbol{x}$ are from a bag-of-words representation, their dimensionality is very high, often much higher than any temporal window which we analyze.

Sparsity inducing regularizers is one way of dealing with the dimensionality issue. We have explored an algorithm that alternates between a Kalman filter update and a convex optimization step of an l_1 norm regularized regression in this context. The regularizer minizes the following cost function:

$$\boldsymbol{\theta}(n) \;=\; \arg\min_{\boldsymbol{\theta}} \; \left\{||y(n) - \boldsymbol{\theta}^t(n)\boldsymbol{x}(n)||_2^2\right\} \;+\; \tau\left\{||\boldsymbol{\theta} - \boldsymbol{\theta}(n-1)||_1\right\}$$

The regularization here constrains the number of changes that the steps between time $n-1$ and time n. A similar idea has been used in [1] for inference of slowly time varying networks in social and biological problems.

At the time of this deliverable implementation of the above has been completed and is being tested on financial data.

Bibliography

[1] A. Ahmed and E.P. Xing. “Recovering time-varying networks of dependencies in social and biological studies”. In: *Proceedings of the National Academy of Sciences* 106.29 (2009), pp. 11878–11883.

[2] Faiz A Al-Khayyal and James E Falk. “Jointly Constrained Biconvex Programming”. In: *Mathematics of Operations Research* 8.2 (1983), pp. 273–286. issn: 0364765X. doi: 10.1287/moor.8.2.273. url: http://mor.journal.informs.org/cgi/doi/10.1287/moor.8.2.273.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. “Convex multi-task feature learning”. In: *Machine Learning* 73.3 (Jan. 2008), pp. 243–272. issn: 0885-6125. doi: 10.1007/s10994-007-5040-8. url: http://www.springerlink.com/index/10.1007/s10994-007-5040-8.

[4] Michele Banko and Eric Brill. “Scaling to very very large corpora for natural language disambiguation”. In: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. ACL ’01. Toulouse, France: Association for Computational Linguistics, 2001, pp. 26–33. doi: 10.3115/1073012.1073017. url: http://dx.doi.org/10.3115/1073012.1073017.

[5] Amir Beck and Marc Teboulle. “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems”. In: *SIAM J. Img. Sci.* 2.1 (Mar. 2009), pp. 183–202. issn: 1936-4954. doi: 10.1137/080716542. url: http://dx.doi.org/10.1137/080716542.

[6] Adam Bermingham and Alan F Smeaton. “On using Twitter to monitor political sentiment and predict election results”. en. In: *Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011)*. Nov. 2011, pp. 2–10. url: http://doras.dcu.ie/16670/1/saaip2011.pdf.

[7] Johan Bollen, Huina Mao, and Xiaojun Zeng. “Twitter mood predicts the stock market”. In: *Journal of Computational Science* 2.1 (Mar. 2011), pp. 1–8. issn: 18777503. doi: 10.1016/j.jocs.2010.12.007. url: http://dx.doi.org/10.1016/j.jocs.2010.12.007.

[8] Léon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT’2010)*. Ed. by Yves Lechevallier and Gilbert Saporta. Paris, France: Springer, Aug. 2010, pp. 177–187. url: http://leon.bottou.org/papers/bottou-2010.

[9] D. Damljanovic and K. Bontcheva. “Named Entity Disambiguation using Linked Data”. In: *Proceedings of the 9th Extended Semantic Web Conference (ESWC 2012)*. 2012.

[10] Andrea Esuli and Fabrizio Sebastiani. “SentiWordNet: A publicly available lexical resource for opinion mining”. In: *Proceeding of the 5th Conference on Language Resources and Evaluation (LREC)*. 2006, pp. 417–422. url: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.7217\&rep=rep1\&type=pdf.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “A note on the group lasso and a sparse group lasso”. In: *CoRR* (2010). url: http://arxiv.org/abs/1001.0736v1.

[12] Daniel Gayo-Avello. “No, You Cannot Predict Elections with Twitter”. In: *IEEE Internet Computing* 16.6 (Nov. 2012), pp. 91–94. issn: 1089-7801. doi: 10.1109/MIC.2012.137. url: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6355554.

[13] Daniel Gayo-Avello, Panagiotis T Metaxas, and Eni Mustafaraj. “Limits of Electoral Predictions using Twitter”. In: *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM)*. 2011, pp. 490–493. url: http://cs.wellesley.edu/~pmetaxas/ICWSM11-limits_predict_elections.pdf.

[14] Alon Y. Halevy, Peter Norvig, and Fernando Pereira. “The Unreasonable Effectiveness of Data”. In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12.

[15] Jonathon S. Hare, Sina Samangooei, and David P. Dupplaw. “OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images”. In: *Proceedings of the 19th ACM international conference on Multimedia*. MM ’11. Scottsdale, Arizona, USA: ACM, 2011, pp. 691–694. isbn: 978-1-4503-0616-4. doi: 10.1145/2072298.2072421. url: http://doi.acm.org/10.1145/2072298.2072421.

[16] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, 2009, p. 552. isbn: 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7. url: http://www.springerlink.com/index/10.1007/978-0-387-84858-7.

[17] Bernard J Jansen et al. “Twitter power: Tweets as electronic word of mouth”. In: *Journal of the American Society for Information Science and Technology* 60.11 (2009), pp. 2169–2188. issn: 15322882. url: http://doi.wiley.com/10.1002/asi.21149.

[18] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[19] Vasileios Lampos. “On voting intentions inference from Twitter content: a case study on UK 2010 General Election”. In: *CoRR*. arXiv:1204.0423 (Apr. 2012). arXiv:1204.0423. url: http://arxiv.org/abs/1204.0423.

[20] Vasileios Lampos and Nello Cristianini. “Nowcasting Events from the Social Web with Statistical Learning”. In: *ACM Transactions on Intelligent Systems and Technology* (2011). url: https://patterns.enm.bris.ac.uk/files/Nowcasting_Events_from_the_Social_Web_with_Statistical_Learning_Preprint.pdf.

[21] Vasileios Lampos and Nello Cristianini. “Tracking the flu pandemic by monitoring the Social Web”. In: *2nd IAPR Workshop on Cognitive Information Processing*. IEEE Press, 2010, pp. 411–416. doi: http://dx.doi.org/10.1109/CIP.2010.5604088.

[22] Vasileios Lampos, Tijl De Bie, and Nello Cristianini. “Flu Detector - Tracking Epidemics on Twitter”. In: *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Vol. 6323. Lecture Notes in Computer Science. Springer, 2010, pp. 599–602. isbn: 978-3-642-15938-1. doi: 10.1007/978-3-642-15939-8.

[23] Vasileios Lampos, Daniel Preoţiuc-Pietro, and Trevor Cohn. “A user-centric model of voting intention from Social Media”. In: *ACL 2013*. Association for Computational Linguistics, 2013.

[24] Jimmy Lin and Alek Kolcz. “Large-scale machine learning at twitter”. In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’12. Scottsdale, Arizona, USA: ACM, 2012, pp. 793–804. isbn: 978-1-4503-1247-9. doi: 10.1145/2213836.2213958. url: http://doi.acm.org/10.1145/2213836.2213958.

[25] Jun Liu, Shuiwang Ji, and Jieping Ye. “Multi-task feature learning via efficient l2,1-norm minimization”. In: (June 2009), pp. 339–348. url: http://dl.acm.org/citation.cfm?id=1795114.1795154.

[26] D.J.C MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[27] André Filipe Torres Martins. “The Geometry of Constrained Structured Prediction: Applications to Inference and Learning of Natural Language Syntax”. PhD thesis. CARNEGIE MELLON UNIVERSITY, 2012.

[28] Panagiotis T Metaxas, Eni Mustafaraj, and Daniel Gayo-Avello. “How (Not) To Predict Elections”. In: *IEEE 3rd International Conference on Social Computing (SocialCom)*. IEEE Press, 2011, pp. 165 –171. isbn: 9780769545783. url: http://cs.wellesley.edu/~pmetaxas/How-Not-To-Predict-Elections.pdf.

[29] John A Nelder and Robert W M Wedderburn. “Generalized Linear Models”. In: *Journal of the Royal Statistical Society - Series A (General)* 135.3 (1972), p. 370. issn: 00359238. doi: 10.2307/2344614. url: http://www.jstor.org/stable/10.2307/2344614?origin=crossref.

[30] M. Niranjan. “Sequential Bayesian computation of logistic regression models”. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing ICASSP*. IEEE, 1999.

[31] Brendan O’Connor et al. “From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series”. In: *Proceedings of the International AAAI Conference on Weblogs and Social Media*. AAAI Press, 2010, pp. 122–129. url: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/viewPDFInterstitial/1536/1842.

[32] Michael J Paul and Mark Dredze. “You Are What You Tweet: Analyzing Twitter for Public Health”. In: *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media* (2011), pp. 265–272. url: http://www.cs.jhu.edu/~mdredze/publications/twitter_health_icwsm_11.pdf.

[33] James W Pennebaker et al. *The Development and Psychometric Properties of LIWC2007*. Tech. rep. Universities of Texas at Austin & University of Auckland, New Zealand, 2007, p. 22.

[34] Hamed Pirsiavash, Deva Ramanan, and Charless Fowlkes. “Bilinear classifiers for visual recognition”. In: *Advances in Neural Information Processing Systems*. Vol. 22. 2009, pp. 1482–1490.

[35] Daniel Preoţiuc-Pietro et al. “Trendminer: An Architecture for Real Time Analysis of Social Media Text”. In: *Sixth International AAAI Conference on Weblogs and Social Media*. AAAI Press, July 2012, pp. 38–42. url: http://eprints.soton.ac.uk/340056/1/4739-22047-1-PB.pdf.

[36] Ignacio Quesada and Ignacio E Grossmann. “A global optimization algorithm for linear fractional and bilinear programs”. In: *Journal of Global Optimization* 6.1 (Jan. 1995), pp. 39–76. issn: 0925-5001. doi: 10.1007/BF01106605. url: http://link.springer.com/10.1007/BF01106605.

[37] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. “Earthquake shakes Twitter users: real-time event detection by social sensors”. In: *Proceedings of the 19th international conference on World Wide Web (WWW)*. ACM, 2010, pp. 851–860. doi: dx.doi.org/10.1145/1772690.1772777.

[38] Shai Shalev-Shwartz. “Online Learning: Theory, Algorithms, and Applications”. PhD thesis. The Hebrew University of Jerusalem, July 2007.

[39] D. Spiegelhalter and S. Lauritzen. “Sequential updating of conditional probabilities on directed graphical structures”. In: *Networks* 20 (1990), pp. 579–605.

[40] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society - Series B (Methodological)* 58.1 (1996), pp. 267–288.

[41] Antonio Torralba, Rob Fergus, and William T. Freeman. “80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.11 (Nov. 2008), pp. 1958–1970. issn: 0162-8828. doi: 10.1109/TPAMI.2008.128. url: http://dx.doi.org/10.1109/TPAMI.2008.128.

[42] Andranik Tumasjan et al. “Predicting elections with Twitter: What 140 characters reveal about political sentiment”. In: *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*. AAAI, 2010, pp. 178–185.

[43] Janyce Wiebe, Theresa Wilson, and Claire Cardie. “Annotating expressions of opinions and emotions in language”. In: *Language Resources and Evaluation* 39.2-3 (2005), pp. 165–210.

[44] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. “Recognizing contextual polarity in phrase-level sentiment analysis”. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2005, pp. 347–354.

[45] Ming Yuan and Yi Lin. “Model selection and estimation in regression with grouped variables”. In: *Journal of the Royal Statistical Society - Series B: Statistical Methodology* 68.1 (2006), pp. 49–67. issn: 13697412. doi: 10.1111/j.1467-9868.2005.00532.x. url: http://doi.wiley.com/10.1111/j.1467-9868.2005.00532.x.

[46] Peng Zhao and Bin Yu. “On model selection consistency of Lasso”. In: *Journal of Machine Learning Research* 7.11 (2006), pp. 2541–2563. issn: 00189448. doi: 10.1109/TIT.2006.883611. url: http://portal.acm.org/citation.cfm?id=1248637.

[47] Hui Zou and Trevor Hastie. “Regularization and variable selection via the elastic net”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (Apr. 2005), pp. 301–320. issn: 1369-7412. doi: 10.1111/j.1467-9868.2005.00503.x. url: http://doi.wiley.com/10.1111/j.1467-9868.2005.00503.x.