Abstract

One of the visions for the smart grid is the creation of a dynamic energy market with multiple active buyers and sellers of energy. Renewables will be more widespread and generated energy may be traded in small to large energy markets. This entails a systems that rely on the buying and selling of energy at the right time in any amount. These systems benefit from having accurate predictions of future energy use.

This thesis proposes a system that uses historical data to predict electrical load on the network using several machine learning algorithms: Decision tree learning, Artificial Neural Networks and Support Vector Machines. An attempt to predict energy usage by mining social media is explored and is shown to be unreliable. The results of the application are made available by a web service that provides data in JSON or XML format.

Contents

1 Introduction

The Smart Grid promises dynamic pricing of energy where different providers and consumers will be able to buy and sell energy. Consumers of energy might also be producers of energy. In the smart grid energy prices will depend on supply and demand. The smart grid is designed with a larger role for renewables in mind. Renewables like solar power and wind energy increase the variation in the supply more drastically. This means buying and selling on the right time will be important in saving costs or even earning an income.

A system to consume energy at the right time has already been built in an office environment at the University of Groningen [13]. The system sets policies for the different devices at the office depending on the price of energy at different times. The system uses a scheduler tries to minimize the the cost energy used.

1.1 Problem

The above system can be improved if predictions of future energy usage are available. The scheduler could anticipate drops and rises in price with access to predictions of drops and rises in demand. Tennet, the energy grid operator in the Netherlands already makes predictions for the total load on the network. High and low loads on the network indicate a high or low demand for energy. This thesis will propose a system to predict electrical load using two different methods.

The first method is to use past data and make predictions using machine learning techniques. Several techniques for predicting energy consumption and electrical load on the network have been used before. These techniques often use past data e.g.: is today a holiday?, what holiday is it?, the day of the week, the hour of the day, the day of the year. Weather data is also commonly used as an predictor of energy loads.

The second method predicts energy usage based on the micro blogging service Twitter. We will use the wisdom of the crowd to predict future energy usage. Although machine learning techniques in combination with social media has not been used to predict electrical load before, similar techniques have been effective in predicting box office revenues [7], movement in stock markets [9], or even detecting events happening around the world [14].

All of these techniques will be implemented in a web service with a small web application so it can easily be integrated into other systems designed for the smart grid.

This thesis will try to answer the following questions:

1. Is it possible to predict electrical load using past data?
2. Is it possible to predict electrical load social media?
3. Is it possible to predict electrical load in a reliable way?

1.2 Organization

Chapter 2 starts with discussing the background information required to answer the research questions. The thesis continues with work done on data mining using social media and the prediction of electrical load in chapter 3.

The design of the software running the web-service is discussed in Chapter 4. Chapter 5 discusses implementation of the software design discussed in the chapter. The following chapter discusses the different experiments. Chapter 7 chapter concludes the results of the experiment and results the software built. The last chapter, chapter 8 discusses the outcome of the thesis and future improvements.

Appendix A contains a manual which describes how to install software built for this thesis. More detailed results of the experiments in chapter 6 can be found in appendixes B and C. Appendix B contains a table with more details about predictions of energy usage using past data. Appendix C shows the calculated correlations between word usage and electrical load.

2 Background

This chapter focuses on the technology and methods used to predict electricity load. The load prediction consists of two important components: retrieving and analysing data from social media and machine learning.

2.1 Social Media

Social media is software that allows people to communicate online. People send each other messages, post messages about what happens in their daily lives, share pictures, chat or play games with others. Most social media offer the possibility of choosing between sharing all this data with your friends or sharing it with everyone. Most users of social networks choose to only share with friends . Social media is very popular in the Netherlands, eight out of ten Dutch people use one ore more forms of social media [18].

2.2 Twitter

Twitter is a social networking service sometimes also referred to as a micro blogging service. The term micro blogging service is sometimes used because Twitter only allows users to share short messages of 140 characters. It does not allow customers to play games or other advanced features other social networks sometimes offer.

Unlike other social networks, Twitter accounts are public by default and it is very common for users to keep their account public. Users are able to become *followers* of other users, when a user follows other users the user sees their messages called *Tweets* on their Twitter home page. Twitter is used by a lot of public figures like politicians and celebrities. It is quite common to follow these public figures, this is different from other social networks where people mainly connect with real friends and family.

Like other social networks Twitter is quite popular in the Netherlands. According to a study from 2012 there are about 5 million Dutch Tweets per day [16]. Another study showed about 3.3 million Dutch people use Twitter [18], this is less then half of the Social Network with the most users Facebook [18]. Although Facebook has more users it is not very common for Facebook users to share their account with the public.

Twitter has an Application programming interface (API) where applications can request JavaScript Object Notation (JSON) formatted Tweets. This API can be used for data mining or building regular applications. This API requires a Twitter account to log in using OAuth. OAuth is discussed in more detail in section 2.3.

2.3 OAUTH

```
  POST /1/statuses/update.json?include_entities=true HTTP/1.1
Accept: */*
Connection: close
User-Agent: OAuth gem v0.4.4
Content-Type: application/x-www-form-urlencoded
Authorization:
        OAuth oauth_consumer_key="xvz1evFS4wEEPTGEFPHBog",
              oauth_nonce="kYjzVBB8Y0ZFabxSWbWovY3uYSQ2pTgmZeNu2VS4cg",
              oauth_signature="tnnArxj06cWHq44gCs1OSKk%2FjLY%3D",
              oauth_signature_method="HMAC-SHA1",
              oauth_timestamp="1318622958",
              oauth_token="370773112-GmHxMAgYyLbNEtIKZeRNFsMKPR9EyMZeS9weJAEb",
              oauth_version="1.0"
Content-Length: 76
Host: api.twitter.com

status=Hello%20Ladies%20%2b%20Gentlemen%2c%20a%20signed%20OAuth%20request%21
```

Figure 2.1: An example of a OAuth request

OAUTH is an authentication framework over HTTP. It allows users to give applications access to their data without sharing their username and password. This prevents the spreading of the username and password to multiple parties. This means when one application gets compromised the username and password are not spread to the attacker. Users often use the same password for multiple websites. A study that matched passwords and email addresses from two hacked databases showed that 76% of the email addresses that where in both databases used the same password [19]. This means that if data leaks from one application, accounts on multiple services are compromised. Another reason for the use of OAuth is that Twitter is able to identify the application. For this reason they also require OAuth authentication when you request public data.

An OAuth request contains unique keys from both the user and the application. The application generates an oauth_consumer_key, this key identifies the application trying to make the request. The key can be created using a Twitter developer account. The application must also request a token from the user called oauth_token. Twitter has several convenient methods of requesting such a token. The simplest way is to generate one on the developer page. Then the application creates a timestamp and a nonce. The nonce is a unique key generated for every request the application makes. The Request also requires the version and the signature method. These are 1.0 and HMAC-SHA1. HMAC-SHA1 is a hashing algorithm, it is commonly used, safe and available in most programming languages. All the keys and request data are combined to sign the request using the signature method to create the oauth_signature. All this data is included in a HTTP request as a header. An example of such a request is shown in Figure 2.1

2.4 Machine Learning/ Data Mining

Data Mining is the extraction of information from large quantities of data. A classical way to mine large quantities of data is to use statistics. Another way is to use machine learning. Machine learning is the programming of computers to optimize a performance criterion using example data or past experience [6].

Because we want to do a large number of experiments we need algorithms that are able to train within 30 minutes at most. This thesis uses three machine learning techniques: M5' Learning trees, a Multilayer Neural Network and a Support Vector Machine (SVM). All of these techniques are described in the sections below.

The Feed Forward Neural Network and SVM are chosen because they are both very popular neural network related machine learning techniques. In practice they have different results when used on different data sets, but both models are popular for predicting electricity load. The M5' model was chosen because it is reasonably fast to train. Other decision tree algorithms that are faster to train exist, but they are specifically designed for speed and have worse predictions than M5' learning trees. Random Forests is another decision tree model commonly used for data mining. Although Random Forests usually have better results, they may take a couple of hours to train. Because time is important we will not use Random Forests.

2.4.1 Decision Tree Learning

Decision tree learning is a technique often used for both classification and regression. Decision tree learning automatically grows a decision tree by splitting each node on an optimal input variable. An example is shown in figure 2.4.1 The algorithm stops splitting the nodes when no significant gain is acquired. Because there are only a few examples left in the leaf nodes decision trees tend to become very specific. This causes noise and outliers have a lot of influence on the trained model, this is called over fitting. Over fitting can be prevented by pruning the trees. Pruning means removing nodes so the tree becomes less specific.

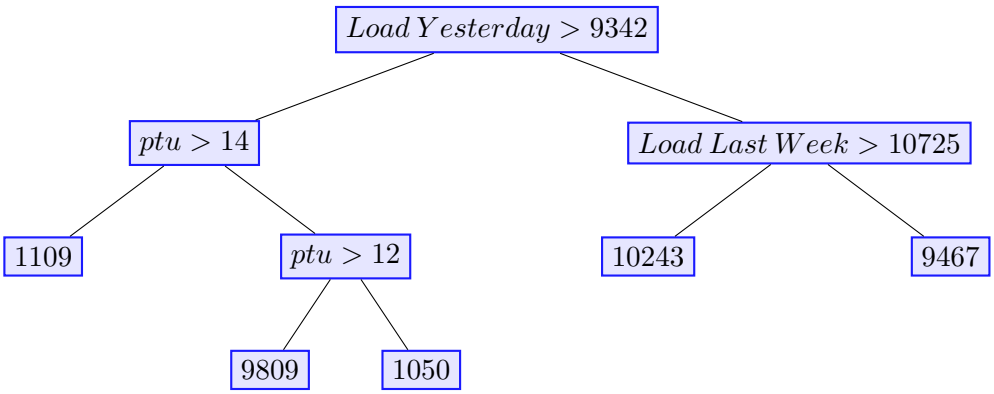

Figure 2.2: Example of a trained Decision Tree

The M5' Decision Tree Learning method is an improvement and reconstruction of the M5 method [21]. The original M5 paper left a lot of data unspecified and did not offer any solution for dealing with missing data and enumerated attributes.

The M5' paper specifies a very detailed picture of the algorithm. Below is a short summary of

its techniques. M5' splits nodes by treating the standard deviation of the values that reach a node as a measure of error and chooses a splitting criterion that minimizes that error. Pruning is done by minimizing the average absolute error. The average will be underestimated because only a few nodes reach the leafs, therefore a compensation relative to the independent variables and nodes reaching the node is made. The pruning stops when the error stops decreasing. M5' also uses a smoothing technique which smooths the differences between the different leafs, this increases the accuracy for most data sets [21].

2.4.2 Multi Layered Feed Forward Neural Network

Multi Layered Feed Forward Neural Networks are the most commonly used Artificial Neural Network (ANN). ANNs are inspired by the brain, real brains contain a large complicated network of neurons. Each neuron influences other neurons with their outgoing connections and is influenced by incoming connections from other neurons. In a real brain these connections can go in every direction or even loop back into themselves. Although such ANNs exist, they are not used a lot in practice.

A Multi Layered Feed Forward Neural Network or Multi Layer Perceptron (MLP) is a lot simpler, all the connections move in one direction towards the output node(s). The layers of neurons in between the input and output layer are called hidden layers. In most applications only a single hidden layer is enough [1] to approximate most functions.

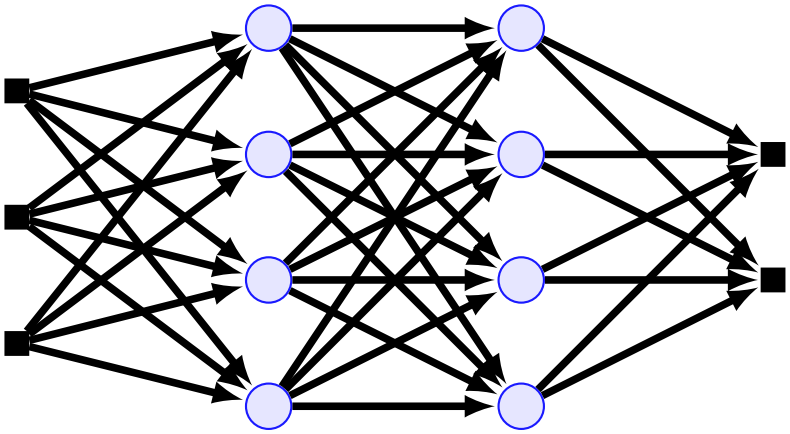

Figure 2.3: A Simple Neural Network

The value of a node at a point in the network is based on the combined weighted value of the input. The network learns by using back propagation, this method slowly changes the weights while moving back through the layers of the network [2].

2.4.3 Support Vector Machines

Support Vector Machines are machine learning models that can be used for both classification and regression. When used for classification linear SVMs create a hyperplane which separate two classes of data [12]. An example is shown in figure 2.4.

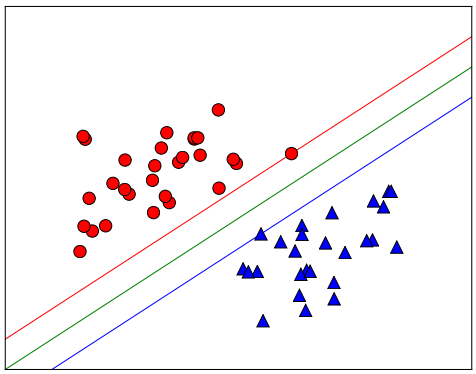

Figure 2.4: 2D example of a support vector machine

The algorithm chooses two support vectors closest to the separating plane. The data in Figure 2.4 is perfectly linearly separable but the algorithm can be modified using an error function to allow for data that is not perfectly separable. By altering the error function SVMs can also be used for regression [12]. An example is shown in figure 2.5.

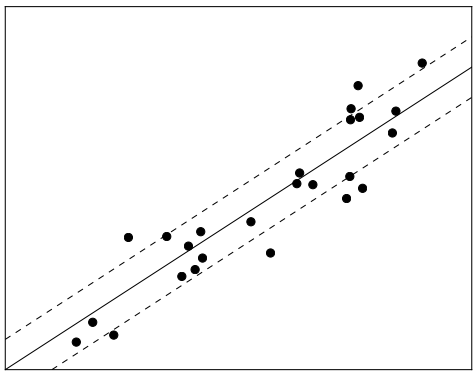

Figure 2.5: 2D example of a regression support vector machine

There are a lot of different techniques for linear regression and classification. What makes SVMs interesting is that they can also be used on non-linear data by using a *Kernel Trick* [12]. The Kernel Trick uses a function called a kernel function to map the data to a higher dimensional space making the data linearly separable. An example of this mapping is shown in figure 2.6. The example has a kernel that gives a higher value for points closer to the center and a lower value to values further away form the center. The points are now separable by a plane. This is just an example kernel and kernels used in practice can be quite complicated. The kernel trick separation technique can be used for both classification and regression.

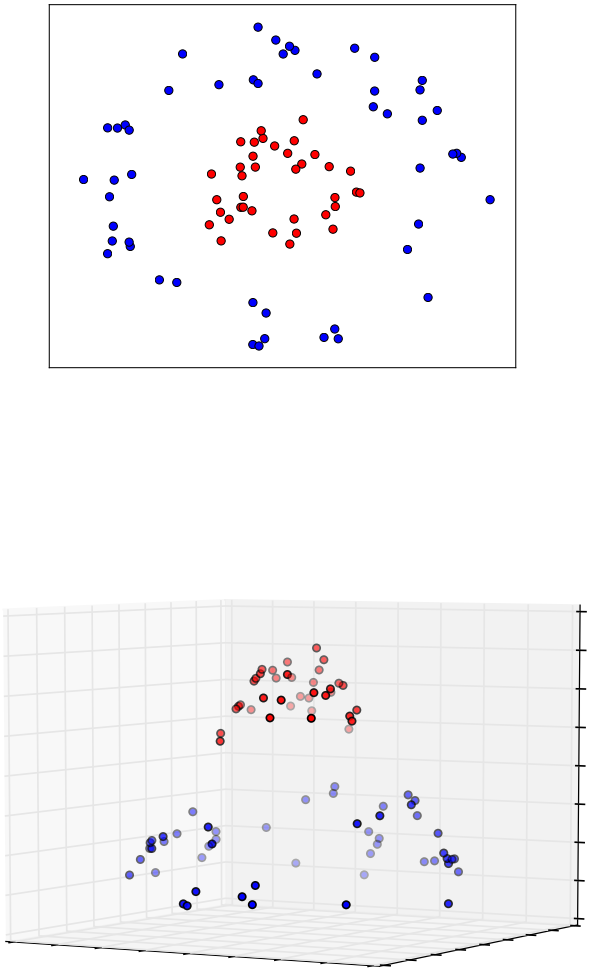

Figure 2.6: Non Linearly separable data mapped to a higher dimensional space

3 State of the Art

Prediction of energy consumption using Twitter consists of two parts. The prediction of Energy Consumption and the prediction of data using Twitter. This chapter discusses the advances in both of those topics.

3.1 Data Prediction Using Social Media

Predicting future outcomes using social media has been successfully used in a couple of applications. Asur et al. [7] showed that it is possible to predict box office revenues of movies. They used a regression model to relate the tweet-rate per hour to box office revenues. They outperformed market based predictors of the Hollywood Stock Exchange. They did this by extracting tweets by performing a search on keywords in the movie title using the Twitter search API. They also performed sentiment analysis on the movies. Sentiment analysis labels words as negative, neutral or positive [11]. They were able to use sentiment analysis to improve their predictions.

A study at the university of Indiana used sentiment analysis to predict movement in the stock market. They measured mood in 6 dimensions Calm, Alert, Sure, Vital, Kind, and Happy. They used the mood dimensions to train a Self-Organizing Fuzzy Neural Network and performed a Granger causality analysis [9]. The study found that these mood predictions match shifts in the Dow Jones Industrial Average (DJIA) 3-4 days later. They had an accuracy of 86.7% and reduced the mean squared error of predictions by 6%.

Becker et al. showed that it is possible to detect events via clustering and similarity techniques [14]. This paper does not try to predict the future using social media but it does show that it is possible to use machine learning techniques to extract information from large quantities of messages produced by users of social media.

3.2 Energy Prediction by Tennet

Tennet, the High-voltage grid operator of the Netherlands, makes its own predictions of the load on their network. Tennet receives transmission projections from all power plants and Transmission System Operators (TSOs) that cooperate at the Union for the Coordination of the Transmission of Electricity (UCTE) level [4]. These models are used for safety calculations [4] and published for implementing the obligations resulting from the directive of the European Commission (Vo. 1228/2003/EG) of increasing Transparancy in the electrical market [5].

Unfortunately the models for all these predictions are not publicly available. The data provided by Tennet suggests that they use the load on the same time last week and the load on the same time the previous day. This is suggested because the XML file containing the loads and forecasts provides these values in the same record as the load.

3.3 Energy Consumption Prediction using Machine Learning

Multiple studies have been done that use machine learning to analyse and predict energy usage. Some of these studies use external data, other studies use historical data. Both types of research relate to this thesis because we will combine predictions based on historical data and predictions based on external factors. The external factors will be extracted from Twitter.

A study in Hong Kong was successful in using machine learning techniques to predict energy usage, by using external factors as data [20]. The external factors are: used household type, household characteristics and appliance ownership to predict the energy consumption of a family. The study also forced the families to record appliance usage in a diary. This study showed that Decision Trees and Artificial Neural Networks (ANNs) were viable alternatives to stepwise regression, which is a more common way of predicting energy usage. The Decision Tree outperformed the ANN in this study.

Several methods for predicting load forecasting have been used in the past. One modern method of load prediction is using a neural network to predict the load on a network several hours ahead [17]. This approach also includes weather data as input for the neural network. It also selects similar days by calculating the euclidean norm based on past load data.

Ortiz-Arroyo et al.[10] showed that using an ANN on the simple inputs: month of the year, day of the week, holiday and week number (Week number of the month) could also be used to predict the load. This model was tested in the 2001 EUNITE competition, it ended in third place and outperformed more complex models.

Chen et al. were the winners of the 2001 EUNITE competition trained a SVM using temperature data, the type of day (Holiday, day of the week) and past data [8]. The study also concluded that temperature data is not easy to use for load prediction because temperature predictions might be inaccurate.

Regression trees have been used for analyzing factors influencing electricity load [15]. GÅĆadysz et al. used temperature data, hour of the day, type of day (weekday, weekend, holiday.. etc) to build their regression tree. This study used learning trees to build a model interpretable by humans to draw conclusions about which factors influence the load. The study showed that certain weather data greatly influence the load. The study used actual weather data instead of forecasts the day before, this means we do not know if weather predictions will help with predicting load. The study also used the type of day (holiday, day of the week) to predict the electricity load.

For an automated system the most promising techniques seem the ones based on past data. Using weather data to train is not easy to do because weather predictions are different from the actual weather. It is also hard for an automated system to rely on data that has to manually entered by humans as used in Hong Kong.

4 Software Design

The results of the experiments in chapter 6 will be made available using a simple web application. This chapter discusses the requirements and high level architecture of the application.

4.1 Requirements

The end user application should make the research results available for client applications. It has to do this by providing a simple interface for client applications and be easy to configure and use. This results in the following high level requirements for the end user application. When the requirements refer to the admin they refer the person controlling the server. Clients are defined as applications requesting predictions.

1. Clients should be able to select a classifier.
2. Clients should be able to retrieve predictions.
3. Predictions should be made automatically.
4. Data from Tennet should be retrieved automatically.
5. Data from Twitter should be retrieved automatically.
6. Data should only be available to clients that have the right credentials.
7. The admin should be able to view the Tennet data stored.
8. The admin should be able to view the Twitter data stored.
9. The admin should be able to view the Prediction data stored.
10. The application should be able to predict the electrical load one day ahead.

The last requirement is dictated by the training data used for training the different classifiers. The classifiers use energy load data from the previous day. More information about the training data can be found in chapter 6.

The client application does not need access to predictions made by the classifiers based on Twitter data. These predictions are worse than guessing the average and therefore not included. For testing the Twitter prediction the application had the following requirements:

1. Data from Twitter should be retrieved automatically.
2. Data should only be from Dutch Twitter users.

4.2 High Level Architecture

The client application is made up of five of basic components, these components are described here. Except for the Twitter Miner all components are dictated by the requirements. The Twitter components were built because they were needed for running the experiments. They have been left in for future developers to expand on.

4.2.1 Data Storage

The application needs to be able to store Twitter users, followers, tweets and load data from Tennet. This data will all be accessed using Database Access Objects (DAOs). DAOs are abstract objects for inserting, updating and deleting objects into a database. The implementation details of these objects are not important for other objects using them.

4.2.2 Data Retrieval from Twitter and Tennet

The Twitter Miner is responsible for checking Twitter for available data. To make the chance of storing non Dutch users as small of possible while getting as much users as possible, the admin should be able to specify a list of users who's followers should be mined for tweets. In this way the admin can include celebrities in which only Dutch people are interested in following. The Twitter Miner has access to information required to authenticate using OAuth.

The Tennet miner periodically checks the Tennet website for new data. Whenever it finds new data it stores it in the database. Unlike the Twitter Miner it does not require authentication data. After retrieving the data it starts the prediction algorithms to predict future loads.

4.2.3 Web service

The web service allows a user to fetch predictions made as either XML or JSON. It also checks credentials of the client before making data available.

URL

The web service is able to respond to requests using urls of the following type: `<adressoftheserver>/predictions/<type>/<start>_<stop>`. This request allows the client to specify which type of data the client wants to receive. It also allows the client to specify between which hours he wants to receive predictions. Valid values for `<type>` are XML or JSON to get the xml or json output. The `<start>` and `<stop>` field are in the format yyyy-mm-dd-hh, the start and stop field are both inclusive to the data. An example URL is `/predictions/xml/2013-08-01-01_2013-08-02-16`. This request should return all predictions between 2013-08-01 01:00 and 2013-08-02 16:00.

Authentication

For authentication the web service uses basic http authentication. Basic http authentication sends the username and password as BASE64 encoded strings in the Authorisation header. An example of a string to be encoded is admin:adminpassword. This authorisation allows for network snooping when using basic http, but is safe when using https. This version of the software will only have one password for all clients. If the service needs to be deployed for a large number of clients it will be necessary to implement the creation of multiple client accounts in future versions.

4.2.4 Website

The website allows the administrator to configure and view settings and the view data contained in the database. The different settings and data screens will be described in the following sections.

Configuration

The settings screen allows the admin to change settings and train classifiers. A screenshot of the settings screen is shown in figure 4.1

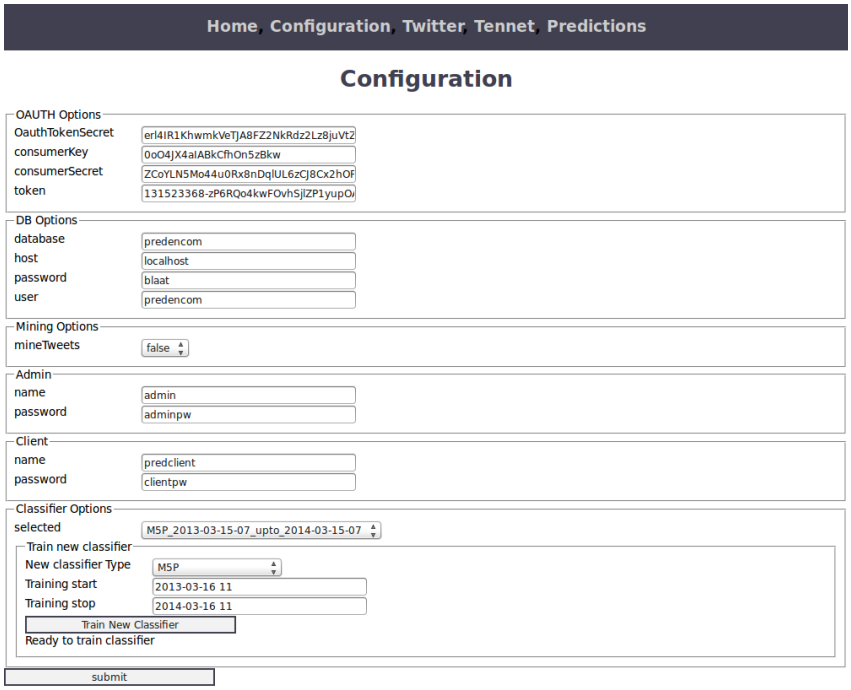

Figure 4.1: Configuration Screen

`OAUTH : consumer key, token, consumer secret, token secret`

These are the settings required for connecting with Twitter, they are discussed in detail in chapter 2.

`Database: host, database, username, password`

These are the settings required for connecting to the database. When the database does not exist yet the application is able to create it if the database user has rights to create new databases. For more information check the PostgreSQL manual.

`Mining Options : minetweets`

This option allows the user to turn Twitter mining on and off. It is off by default because Twitter based predictions have failed.

`admin : name, password`

Username and password for the administrator, this username and password are required to edit settings and view data.

`client : name, password`

Username and password for the client. Client applications need to use this name and password to connect.

`classifier : selected, new classifier type, training start, training stop`

These are the settings for the classifier that will be used. The selected option specifies the trained classifier that is currently used for predicting. Before you can select a classifier you need to train one. The user needs to specify a start date, stop date and type of classifier (M5P, MultilayerPerceptron or SMOreg) afther this he can press the train "classifier new button." When the classifier is finished training it will be stored on disk. The user will be notified that training of the classifier is finished and be able to specify it as the *selected* option. It is possible to have multiple classifiers stored on disk and switch between them.

Data Screens

The Administrator is able to view some of the data stored in the database on the website. There are multiple screens available to the admin.

The Twitter screen allows the admin to view the rate limit status for the different parts of the Twitter api used, it also contains a list of the users added by the admin. A screenshot of the twitter screen is shown in figure 4.2. The bottom of the page contains a field where the admin can add twitter ids of users to add to the database. A screenshot is shown in figure 4.4. When the admin clicks on one of the users added, the admin is able to view his followers and tweets. A screenshot of a screen for a individual user is shown in figure 4.3. The screenshot of the individual user screen shows a the followers and tweets of the prime minister Mark Rutte. As you can see he has 25115 followers and is a very good example of how many Dutch speaking users you can find from one popular Twitter user.

The Tennet screen allows the admin to view data from Tennet. It also allows the admin to manually upload XML files retrieved from Tennet. A screenshot of the Tennet screen is shown in figure 4.5.

The predictions screen shows predictions made for the hours specified by the user. When actual data is available for that hour it is also displayed in the table. It has a similar endpoint as the webservice that returns XML and json. To view XML or JSON predictions the user can simply replace "HTML" in the address bar by "XML" or "JSON". The user will be asked for the client credentials when doing this. A screenshot of the predictions page is shown in figure 4.6

Home, Configuration, Twitter, Tennet, Predictions

Twitter

Miner	Stutus
Tweet Miner	Stopped
Follower Miner	Stopped
To-Parse Miner	Stopped

Rate Limiting

Bucket	Rate Limit Limit	Limit Remaining	Limit Reset
/application/rate_limit_status	180	174	2014-03-15 08:27:02
/followers/ids	15	15	2014-03-15 08:38:00
/statuses/user_timeline	180	180	2014-03-15 08:38:00
/trends/place	15	15	2014-03-15 08:38:00
/users/lookup	180	180	2014-03-15 08:38:00

Users

Id	screen Name	Description
25845631	CharlyLownoise	DJ, producer, Zenteacher, WPIP
155507136	MinPres	Minister-president van NL / Redactie door RVD / Reacties zijn welkom / Alles wordt gelezen, maar niet (direct) beantwoord / Vragen? Ga naar onze website!
3103511	trouw	Opgericht in WO2, in 2005 eerste NL-krant op tabloid, oplage 100K site 1M. Zie ook trouw.nl/groen, trouw.nl/rf, trouw.nl/onderwijs en trouw.nl/indebuurt
2902821	telegraaf	LET OP dit is de officiële account! Het laatste binnenland, buitenland, sport, privé en financieel nieuws
42641124	erikdezwart	Media ondernemer, stationvoice Radio Veronica, voorzitter St. Nederlandse Top 40, voorzitter St. Mat54, bestuurslid Nationaal Pop Museum http://t.co/eOcYyfGY
16708690	APechtold	Fractievoorzitter D66 Tweede Kamer
18976324	diederiksamsom	Fractievoorzitter van de Partij van de Arbeid in de Tweede Kamer.
96975923	Dolfjansen	Hardloper, grootspraker, dichter en grapmaker
39486854	andriesgknevel	Echtgenoot, vader, opa van Bibi, Maxe, Victor en Jasper, televisiepresentator, schrijver, preker en spreker.
14215482	dweesie	Manusje van alles bij PowNed

1 2

Figure 4.2: Twitter Screen

Home, Configuration, Twitter, Tennet, Predictions

Id	screen Name	Description
118738430	LaagLand_Media	Full service Webdesign en Grafisch Ontwerp bureau. Webdesign | Grafisch Ontwerp | Marketing & Advies
1245583831	MrBraams	Werkzaam bij de overheid, inline skaten in de zomermaandjes, nu vooral bezig met zelfstudie photoshop en nog allerlei Adobe-producten.
1051345027	wdegroot92	
1261466215	E_Esveld	
224350812	henkplas	Sport - Dorps Evenementen - Incidenten - Activiteiten - Natuur- Feestpartijen ..Fotografie. Voor opdrachten graag DM
324314748	man_d89	Starter BBA CE Marketing . running; swimming; online marketing; Psychology; Economy; Creative; The Netherlands
1277755813	MWoering	Father of Emma, Sophie & Thomas / Married to Jennifer / Oracle / Program Manager EMEA Systems Support / Business Engagement / Quality Mgt / Opinions are my own!
969994081	aramdjiawan	
269350741	grove3050	kraanmachinist vest. kampen
1291858806	gradusmiedema	

1 2 3 ... 25116

time	message
2010-10-28 06:55:59	Via dit kanaal zal de RVD regelmatig verslag doen van de activiteiten van de minister-president.
2010-10-28 06:56:58	Bekijk of lees hier de Regeringsverklaring van de minister-president http://ow.ly/30G9D
2010-10-28 08:20:05	Op rijksoverheid.nl staat een samenvatting van de kabinetsplannen: http://ow.ly/30Hj9
2010-10-28 13:13:56	MP Rutte is vanavond en morgen bij de Europese Raad. Hoofdonderwerp: meer begrotingsdiscipline en afstemming van macro-economisch beleid.
2010-10-29 11:43:03	MP Rutte na afloop Europese Raad. Goed resultaat: stevige begrotingsregels, strikte voorwaarden aan hulp bij eurocrisis en slot op EU-budget
2010-11-01 15:03:07	MP Rutte arriveert maandagmiddag op Paleis Huis ten Bosch voor het wekelijks gesprek met de Koningin. http://ow.ly/32yRt
2010-11-02 11:32:58	Dank voor alle reacties op de lancering van @MinPres. Deze worden niet direct beantwoord, wel gelezen. Nu en dan worden FAQ's beantwoord.
2010-11-03 19:19:48	President Obama heeft zojuist MP Rutte gebeld om hem te feliciteren met zijn aantreden als minister-president. Foto: http://is.gd/gGaAH
2010-11-04 16:10:03	MP Rutte bereidt met zijn medewerkers de ministerraad van morgen voor. Foto: http://is.gd/gIs3K
2010-11-05 14:36:31	Bekijk live de persconferentie van MP Rutte na de wekelijkse ministerraad. Te verwachten vanaf 16.30 uur. http://is.gd/gKpua

1 2 3 ... 77

Figure 4.3: Twitter User Screen

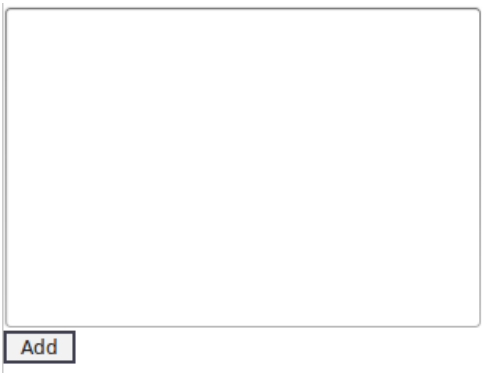

Figure 4.4: Twitter Add User Screen

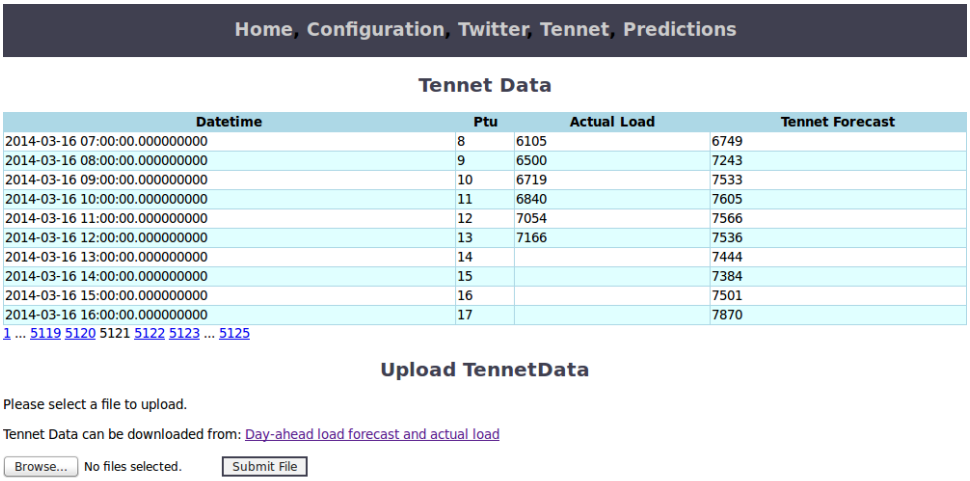

Figure 4.5: Tennet Screen

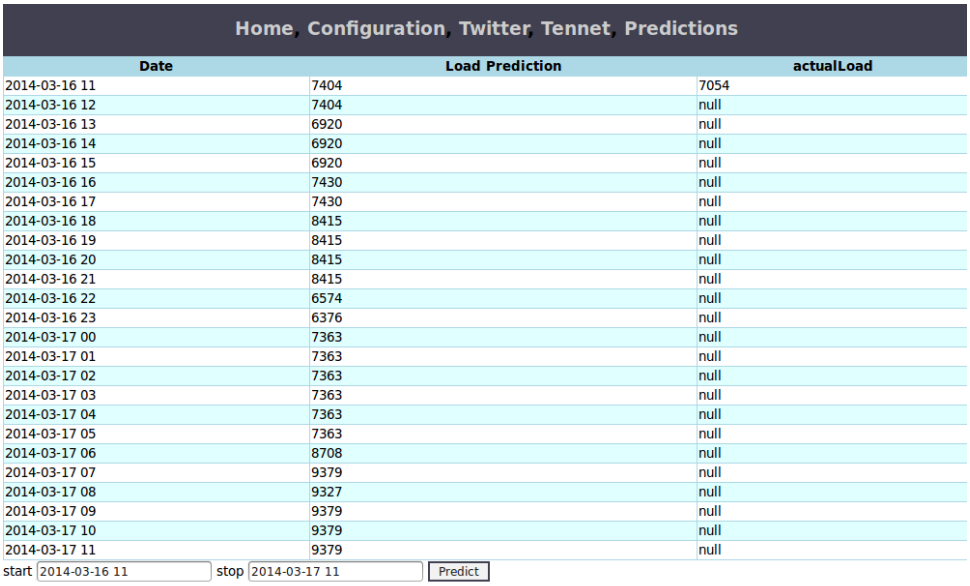

Figure 4.6: Predictions Screen

5 Implementation

This chapter describes the implementation details of the application. It starts with describing the external software used to build the application and ends with class diagrams describing the different parts of the application.

5.1 External Software

This section describes external software used by the application. The application uses external software for implementing machine learning algorithms, running a web server and storing data. When choosing external software, commonly used tools within the Java community were preferred.

5.1.1 Java and Weka

When choosing a machine learning library a number of different platforms stand out. Matlab and R are very common platforms for machine learning research. The upside of these platforms is that they have tools for quickly plotting and interpreting data and are therefore very easy to use for prototyping. The downside is that they have not been built for creating general purpose software like websites and web services.

C/C++ also has a lot of machine learning algorithms available. The different libraries are spread over many different libraries and the tooling around it is very minimal. Although it is possible to build web applications in C/C++, it takes quite a bit of effort and is not very common.

Weka is a machine learning library from The University of Waikato for Java, it contains most commonly used machine learning algorithms. It is commonly used just like Matlab and R. The downside is that the collection of machine learning algorithms is smaller than the ones contained in Matlab and R. Next to being a library, Weka has a collection of graphical tools to experiment with. This makes it almost as easy to quickly interpret data as Matlab and R. Because Weka is written for Java, it is easy to build general purpose applications around it. Java is also commonly used for building web applications, which makes it very suitable for this project.

Weka has a GPL licence which requires the source code to be shared with everyone who owns a copy of the program. GPL is copy left which means every application that links to it also becomes GPL. Linking in this case does not include people who use the web service, linking only includes applications that use the binary. If the application needs a different license in the future a different license can be purchased from The University of Waikato.

5.1.2 Tomcat

Tomcat is a Java web server with limited supports for JavaEE applications. It supports JSP and Servlets. Both the web service and the web site run on the Tomcat service. Tomcat is

lightweight and requires a small amount of memory and CPU usage. This makes it an ideal server because it leaves memory and CPU time for training classifiers using Weka. Tomcat is also easy to configure and allows the addition of other JavaEE libraries when required. Should a full JavaEE stack be required in the future there is a full stack Tomcat server available called TomEE. Existing applications can be deployed without needing to port or reconfigure them.

5.1.3 Apache Ant

The software was built using the Integrated development environment (IDE) eclipse. To make build process independent of the IDE Apache Ant was used. Ant is a standard in the Java community and supported on multiple platforms. It also allows for easy deployment to Tomcat servers. Unlike other build systems like Make, it does not require a special text format relying on white space and uses XML files in stead.

5.1.4 PostgreSQL

The application stores a lot of data, the data is stored in two separate ways: on the file system and in a database. When choosing between different database systems, MYSQL and PostgreSQL were considered. Both databases work on multiple platforms and are common choices when developing applications. Both systems have java JDBC driver support for java application development. Both applications have good performance and a big feature set.

PostgresSQL was chosen because of its more permissive license and stricter default settings. MYSQL is GPL licensed and an application that can only connect to MYSQL is considered a derivative work requiring it to also be GPL licensed. It is however possible to purchase other licenses from Oracle. PostgreSQL has a MIT style license making it possible to use for whatever type of application required.

The default MYSQL settings are not very strict and silently convert null values to empty strings or zero. This is known to cause integrity problems for applications using MYSQL. It is possible to configure MYSQL to be more strict. This would make deployment more difficult however. PostgreSQL has strict server settings when using the default installation.

5.2 Data Storage

Figure 5.1 shows the relations between the different classes required for storage in the database. The DBConnector class provides a low level interface for interacting with the database. It allows the retrieval and storing of data. Database transactions can be executed for single rows or as large transactions manipulating rows in multiple tables and mutiple rows as a single transaction. The DBConnector class uses objects of the ResultsSetHandler interface to convert database rows to Java objects and the StateMentPreparer interface to convert java objects to database rows.

The DAOs provide a more abstract interface to store and retrieve data. The Database object is responsive for creating the DAOs and providing it with the low level DBConnector object. The database object also contains all the information required to connect to the database, like the port, password, username and the database name.

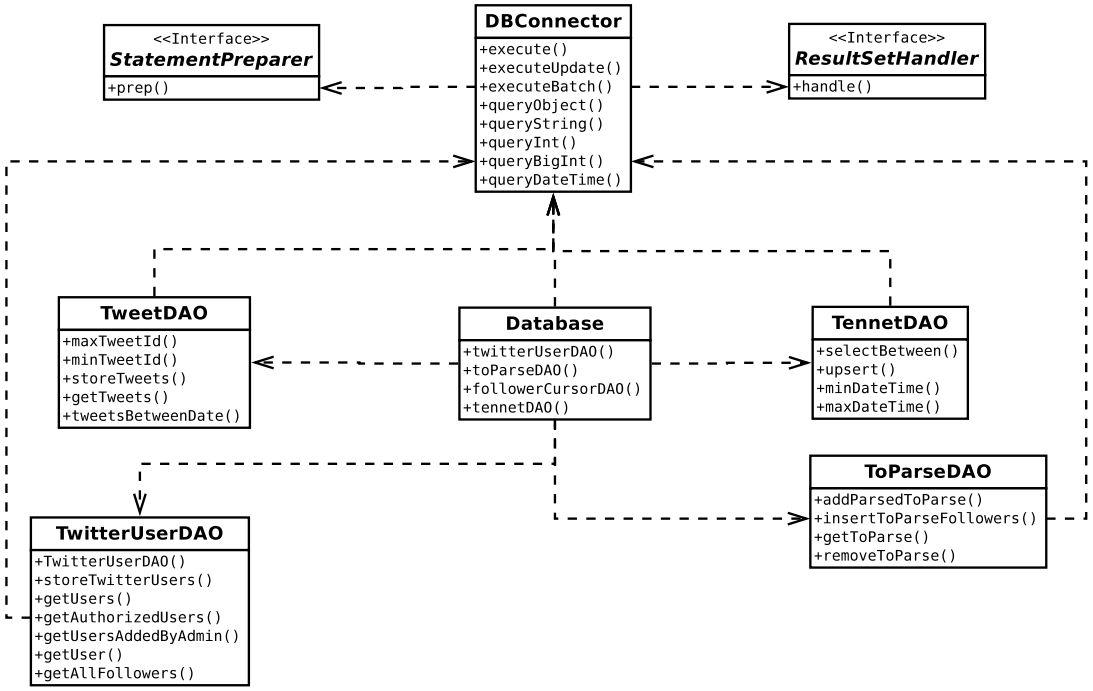

Figure 5.1: Data Storage Class Diagram

5.3 Web Data Retrieval

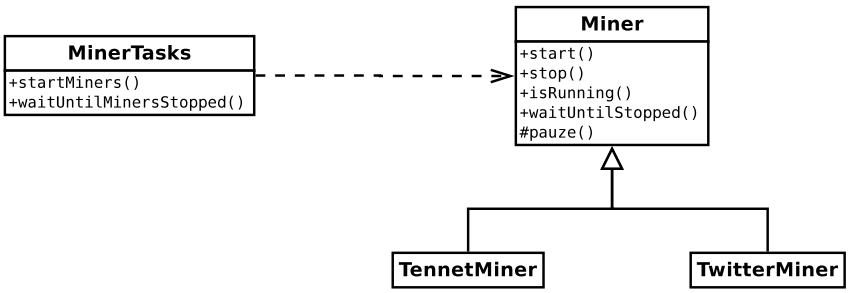

Figure 5.2: Miners Class Diagram

The application is capable of retrieving data from both Twitter and Tennet. A MinerTasks object controls the starting and stopping of all the miner tasks. The Miner objects all run as separate threads, they have a pause function to safely pause themselves when they have nothing to do. The class relationships are shown in figure 5.2

5.3.1 Tennet Data Retrieval

The TennetMiner Checks for updates every five minutes, it checks for available data that is older than the data already stored in the database. It checks for new data every 5 minutes. Between the different intervals the TennetMiner pauses. To know which records to fetch, it needs to know the maximum date-time stored in the database. If recent load data contains null it will try to find new data for those records. The relationships are displayed in figure 5.3

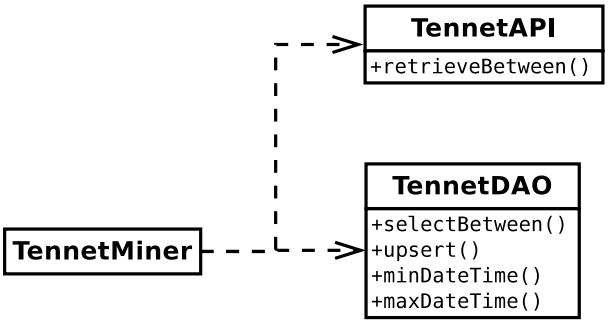

Figure 5.3: Tennet Miner Class Diagram

5.3.2 Twitter Data retrieval

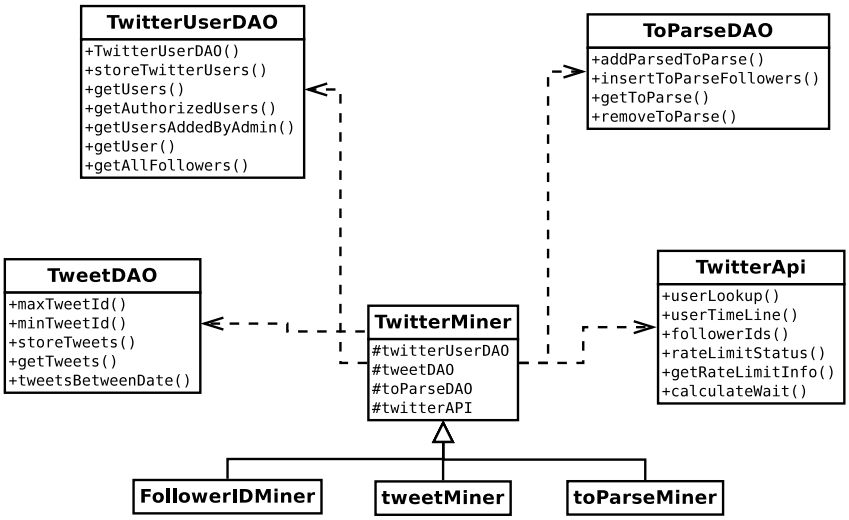

Figure 5.4: Twitter Miner Class Diagram

Figure 5.4 shows the relationship between the different data and miner classes. The FollowerIdMiner searches the database for users added by the admin, for those users it makes requests to the Twitter API to retrieve the followers of those users. It adds the ids in the database for the ToParseMiner to use.

The ToParseMiner searches for users it needs to parse in the database and retrieves them using the Twitter API. The users are stored in the database with a flag indicating that they were not added by the administrator. This way the followers of the followers will not be retrieved by the FollowerIdminer.

The TweetMiner searches the Twitter API for tweets not yet stored in the database. It searches for tweets off all the users stored in the database including the followers of the users added by the admin.

The Twitter Miners have different end points in the Twitter API, all those end points have different rate limits. When a rate limit is reached the miner pauses until the rate limit resets, which is every 15 minutes.

5.4 Website and Web Service

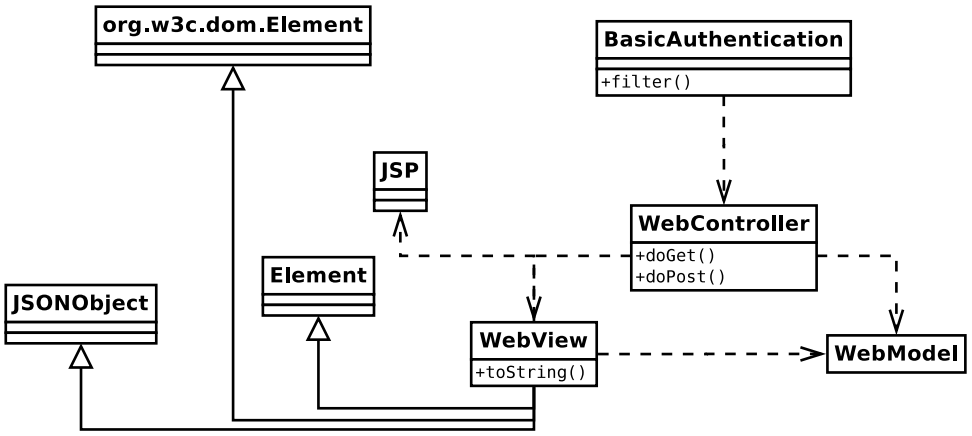

Figure 5.5: Web MVC Diagram

The web site and web service use a web variant of the Model View Controller pattern. The controller receives get or post httpRequests, it parses parameters specified by the user and passes them to the model. The models are passed to the views who represent them as HTML. This is the basic implementation for every page and for the web service endpoint. The controllers are implemented as Java Servlets. The implementation of the pattern is shown in figure 5.5.

Basic HTTP authentication is implemented using a Fiter from the servlet specification. Every request reaching the web application has to go through the Filter, it checks for the admin username and password for admin pages and checks for the client username and password for the Webservice pages.

The MVC Pattern makes it easy to create multiple ways to display the same data. The prediction endpoint demonstrates this property of MVC the best, it uses the view to display multiple types of output : HTML, XML and JSON. The Controller class parses the user input, the model fetches data and makes predictions. The view displays predictions, or an error when there is a problem with the user input or when the system is not properly configured. The model class does not care which type of output needs to be given. It makes predictions and returns the results. The model class does not have a reference to the view and controller classes, this means it does not need to be updated when the view and controller classes change. The prediction controller uses an extra view in the form of a JSP page to insert the table created in the PredictionView class. The implementation is show in figure 5.6.

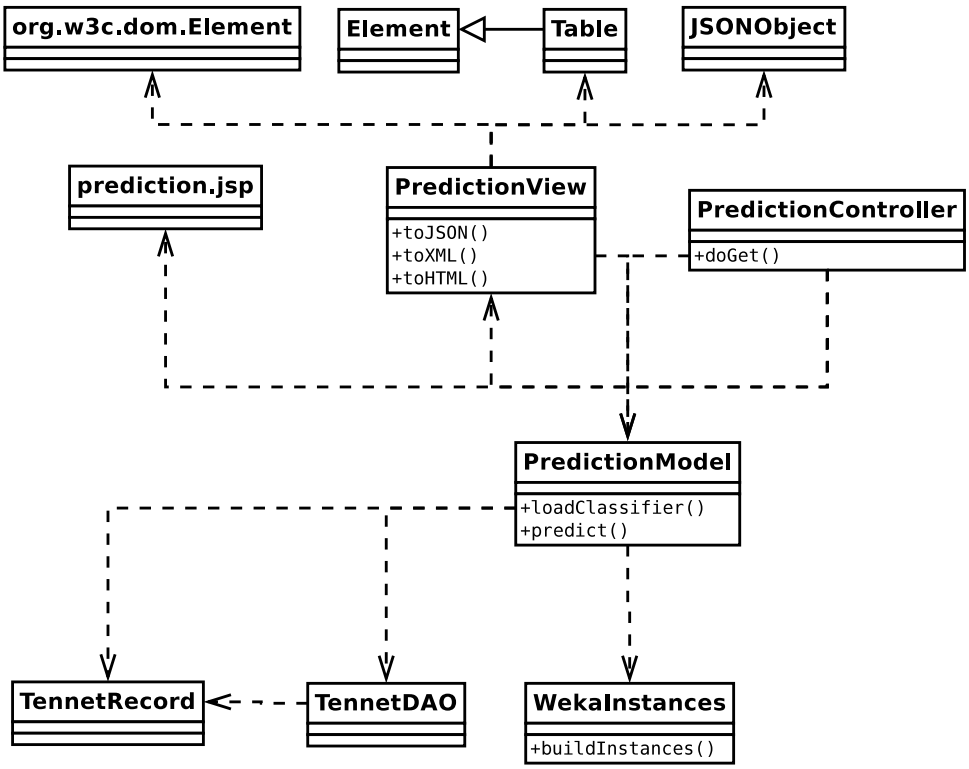

Figure 5.6: Prediction MVC

6 Experiment

This chapter discusses the different experiments, it starts with the experiment without Twitter data in section 6.1 and ends with the experiments including Twitter data in sections 6.2 and 6.3.

6.1 Energy Prediction Using Past Data

To train the different classifiers the attributes in table 6.1 were used.

historicalDay	The electrical load same time yesterday
historicalWeekDay	The electrical load same time last week.
dayOfYear	The day of the year.
dayOfMonth	The day of the month.
dayOfWeek	The day of the week.
ptu	The hour of the day. This value is important because a day can have 23 or 25 hours because of daylight saving time.
holiday	The current holiday, the holidays included are : New Year's Day, Good Friday, 1st Easter Day, 2nd Easter Day, Queensday, Feast of the Ascension, 1st Pentecost Day, 2nd Pentecost Day, First Christmas Day, Second Christmas Day, New Year's Eve. or "No holiday" when there is no holiday.
actualLoad	The load of the current hour, the value the predictor is actually trying to predict

Table 6.1: Attributes used for training

HistoricalDay and historicalWeekday where chosen because they were included in the Tennet XML data. The assumption was made that this data was included because it is useful for predicting the load. Models in other studies also used similar data. The HistoricalDay limits the classifiers to only be able to predict a day ahead of the last data stored in the database.

Holiday data can be represented as two different types of data: an enum type containing each specific holiday or a boolean indicating if the day is a holiday or not. The experiment is repeated for both of those types of holiday values. The different classifiers are trained by using one or two years of data and testing it on the next year. The results of the different classifiers are verified by comparing the Root Mean Squared Errors (RMSEs). The correlations are also given to show the dependence between the predictions and actual load on that day. Another criterion for selecting the best classifier is the training time.

6.1.1 Classifier Parameters

The M5P' decision tree classifier does not require any parameters to be set. The configuration of the MLP and SVM are described in the following sections.

Multi Layer Perceptron

Two important parameters to select with the MLP are the number of hidden layers and the number of nodes in each layer. The number of hidden layers influences the types of functions the neural network can approximate [3]. After trying one hidden layer and then two hidden layers, two hidden layers were chosen because they performed better. More than two layers are never necessary because two layers can approximate any decision boundary [3].

A guideline for choosing the number of neurons in each layer is 2/3 of the size of the input layer plus the size of the output layer [3]. The neural network implementation of Weka uses an input for each entry in the enum. The best way to represent a boolean in Weka is an enum with the values true and false. This means 14 or 8 neurons in each hidden layer are needed when using enum values or boolean values.

Support Vector Machines

For the support vector machine it is important to select the right kernel. When training the MLP it is necessary two use two hidden layers so any decision boundary could be learned. This means a kernel with the same properties is necessary for the support vector machine. A Radial Basis Kernel was selected because it can approximate the same types of functions as a two layer MLP.

6.1.2 Results

The results of the different classifiers are shown in table 6.2. The first column: Classifier, indicates the classifier used. The trained column indicates if one or two years were used for training the data. Boolean holidays indicates if a boolean or enum was used as holiday data. The correlation is the Pearson product-moment correlation coefficient, it is a value between -1 and 1 and gives the degree of linear dependence between the predictions and the actual loads. A value of 0 means no correlation a value of 1 means perfect correlation, and a value of -1 means perfect opposit correlation. A value of 1 is just as good as -1. To turn a correlation of -1 to 1 invert your results. The next column is the most important and gives the Root Mean Squared Error (RMSE) in megawatt per hour. A low value indicates a low margin of error. The last column is the training time, this is also an important factor because it makes training of a new classifier easier to do.

The classifier that comes out best is the MLP trained on one year of data. The RMSE in megawatt per hour of the MLP is 96.19% of the RMSE of M5P. The training time of M5P is significantly faster, but all the classifiers can be used for an entire year making training time less important. The difference in prediction is significantly better than the predictions made by Tennet. The RMSE is 79.25 % of the RMSE of the predictions made by Tennet. The difference between using boolean or enum holidays is not significant, this means it is better to use boolean holiday values because they are faster to train.

Average Performance					
Classifier	**Trained**	**Boolean Holidays**	**Correlation**	**RMSE**	**Time**
MultilayerPerceptron	Year	false	0.9325	687.1685	01:50:690
MultilayerPerceptron	Year	true	0.9373	687.8613	00:37:060
M5P	MultiYear	true	0.9232	714.3827	00:05:461
M5P	MultiYear	false	0.9231	714.8732	00:08:237
MultilayerPerceptron	MultiYear	false	0.9359	732.7809	03:34:748
MultilayerPerceptron	MultiYear	true	0.9369	747.4010	01:08:093
SMOreg	MultiYear	false	0.9125	751.4962	21:19:317
SMOreg	MultiYear	true	0.9117	754.3739	20:37:406
SMOreg	Year	false	0.9111	757.1660	00:27:245
SMOreg	Year	true	0.9104	759.7767	00:29:444
M5P	Year	true	0.9135	774.3595	00:01:957
M5P	Year	false	0.9134	774.7928	00:03:292
TennetForecast	unknown	false	0.8840	867.0842	unknown

Table 6.2: Test Results of Trained Classifiers

6.2 Prediction Using the Mean Load on Word Occurrence

A way of trying to correlate tweets and electrical loads is by correlating a load on a given hour to a word. To predict the load x hours away the loads must be correlated to words that were found x hours back. The algorithm does not use a list of predefined words. Anything that is separated by white space or punctuation is counted as a word. This is done because new words pop up in social media all the time.

First, all the words are counted in a certain hour, for a given time period. The words with a count below a certain threshold are thrown away. Then the loads for the number of hours ahead are kept in a list, together with the word count. Then a weighted average and weighted variance are calculated using the word count. We end up with a table containing the weighted averages and variances for each word. Because some words might not have a specific correlation to any load the words with the highest variance are thrown away.

For the actual prediction all occurring words in an hour are counted, then the algorithm searches the table containing the average load for a word and adds that load and current word count to a new table. We use the counts and words table to compute a new weighted average, which is the prediction for the number of hours ahead.

6.2.1 Example

This section is a simple example of only two words above the minimum word count and low variance.

Hour two had a load of 6500 and hour three had a load of 8000. The word "hello" was found 2000 times in hour one and 1000 times in hour two (3000 in total). The word "world" was found 1000 times in hour one and 1000 times in hour two (2000 in total). The average load one hour later for the word "hello" is $(2000 * 6500 + 1000 * 8000)/3000 = 7000$ and the average load for the word "world" one hour later is $(1000 * 6500 + 1000 * 8000)/2000 = 7250$.

We want to predict the load for hour four. The word "hello" was found 1000 times in hour three and the word "world" 2000 times. To calculate the expected load we do $(1000 * 7000 + 2000 * 7250)/3000 = 7166.67$.

6.2.2 Results

The algorithm was used to predict the load upto six hours ahead. Minimum word counts of 300, 600 and 900 and top percentages of lowest variance of 0.2, 0.4, 0.6, 0.8 and 1.0 where used. The RMSE of the algorithm was compared with RMSE of the mean value of the same period. A table of results can be found in Appendix C

The RMSE was usualy worse than that of the mean. This means that using the mean as a predictor is better than using prediction based on word count. The parameters do not have a significant impact on the performance of the algorithm. This means that the algorithm basically gives output unrelated to the actual load. This is confirmed when looking at the table in Appendix C the correlation between the prediction and the actual load is around 0.2. This means the algorithm can not be used to predict electrical load.

6.3 Smiley Based Mood prediction

A simple form of mood prediction is to count the occurrence of smilies in an hour, and correlate this to the load a specified number of hours ahead. To test this classifier an M5P' tree and MLP were trained on data containing the count of happy, laugh, sad or cry and the load an hour later. Table 6.3 shows the different smilies used for happy, laugh, sad or cry.

Happy	:) , :-)
laugh	:D , :-D , xD , x-D
sad	:(, :-(
cry	:'(, :'-(

Table 6.3: Smilies used for training

6.3.1 Results

The RMSE of both predictors was very high: 1881 for the MLP and 1896 for M5P. When comparing them to predictions done by Tennet the RMSEs are 216.93 % and 218.66 % higher.

Another problem was M5P classifier kept predicting the same value. This means it could not find any useful correlation between the different instances and it built a one node decision tree. This method can not be used for the prediction of the electrical load.

7 Conclusion

This chapter answers the questions asked at the beginning of this thesis.

7.1 Research Questions

1. Is it possible to predict electrical load social media?

The attempts at using social media for predicting the load on the electrical network were unsuccessful. Both the word to load correlation and smiley prediction were not better than using an average load as a predictor over the same period.

2. Is it possible to predict electrical load using past data?

Yes, predicting energy consumption using past data was successful and outperformed predictions made by Tennet. From all of the different learners the MLP trained on one year of data performed the best. Both SVM and learning trees performed best when trained on two years of data. The difference between using boolean values or enum values as holiday data is not significant. This means using boolean values are used in the end application because they are faster to train.

3.Is it possible to predict electrical load in a reliable way?

It is possible to predict energy consumption in a reliable way. Not only are the predictions reliable, they are also better than the ones currently made by Tennet. The experiment succeeded in predicting energy consumption using passed data and makes this data available using a web application.

7.2 Application

A web application was built to make the results available for client applications. The web application is a simple prototype that can be used for small scale deployment. If the application needs to be deployed to a large number of clients it is recommended to implement separate credentials for each client. The application should be easy to update because it is built on top of a number of standard components from the Java community.

The application includes functionality for mining Twitter and viewing the data. This functionality could be updated in the future to try other algorithms that were not presented in this thesis.

8 Discussion

Although it is not possible to predict electrical load using social media with the techniques described in this paper. It is definitely possible to have build reliable prediction algorithms using past data. The web application built for this thesis is able to give reliable load predictions and is ready to be integrated into applications for the smart grid.

Even though the algorithms in this paper do not work for predicting electrical load using social media, there are still possibilities open for future work. Future work could try to focus on more advanced ways of doing mood prediction by basing the mood on the words used in a tweet. However, this is not an easy task [11], and a problem on its own without including the prediction of electrical load.

There are also ways to improve the prediction based on past data. One way is changing the algorithm to allow it to predict further ahead by not using data of the day before. This method does have the risk of making the data less accurate. But trade off of knowing predictions more in advance might be worth it. Another way might be to find a reliable way to include weather predictions. Because training on real weather data might be a problem getting access to a database of old predictions to train a classifier on might be more useful.

A Manual

This chapter contains a manual for setting up the Energy prediction service web-service and web-application on a development machine. It deals with installing the required dependencies and building and running the software. The manual is written for windows and Linux. The Linux distributions chosen are Debian and Ubuntu, for other distributions the user must search for the right packages in the package manager or manually install the required software.

The service has three external dependencies that need to be installed on the development machine. Tomcat (webserver), Postrges (database) and Ant (build system). When moving the service to production use the documentation of Tomcat, Postgres and Ant to set them up in a secure way.

A.1 Postgres

This section deals helps you getting a postgresql server running on your development machine. The method shown here requires you to manually start the postgres server. For starting Postgres as a refference the postgres documentation.

A.1.1 Linux Installation

This section deals with installing Postgres on a Ubuntu or Debian using the package manager and by building from source. Building from Source is required when you do not have root access on a machine.

Installation using the package manager

Debian and Ubuntu have postgres in the repositories of their package managers. type `sudo apt-get install` to install postgres using the package manager.

Building From Source

When you don't have root access on a machine you might have to build from source. Download the source from : `http://www.postgresql.org/ftp/source/` and run the following commands in the source directory (replace useraccount with your own user account).

```
./configure --prefix=/home/useraccount/postgresql-9.2
make
make install
```

Posgresql should now be installed in you're home directory. Postgresql needs a data directory to work

```
mkdir ~/psqldata
cd postgresql-9.2/bin
initdb -D ~/psqldata
```

Start postgresql server:

```
postmaster -D ~/psqldata/
```

A.1.2 Command line Client

To enable connections open /etc/postgresql/9.2/main/postgresql.conf (/psqldata/postgresql.conf on a local install) for editing and uncomment the following lines.

```
listen_addresses = 'localhost'
password_encryption = on
```

This will allow local connections and encrypt passwords. You can add other IP-adress by following the instructions in the comments. Then use the following command to restart the server.

```
sudo /etc/init.d/postgresql restart
```

Create a user with a password (The user will at least need database creation rights):

```
sudo -u postgres createuser --interactive predencom -P
```

Start the command client using.

```
psql -h localhost -U <username> <dbname>
```

A.1.3 Windows Installation

Postgres is available for windows using a installer that can be downloaded from the postgres website. It contains a GUI to configure it any way you like without using config files.

A.2 Tomcat

This section explains setting up a Tomcat 7 server.

A.2.1 Installation

Download tomcat 7 from `http://tomcat.apache.org/`. make sure you download version 7 The application has been developed for Tomcat 7 and has not been tested on Tomcat 8.

A.2.2 Configuration

Tomcat needs an environment variable `CATALINA_HOME` that points to the lcoation of the extracted tomcat folder. To enable the deployment of applications a couple of Users must be defined in

CATALINA_HOME/conf/tomcat-users.xml. The following example shows a user with the required privileges to add applications using scripts or via the manager application that comes with tomcat.

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>
    <role rolename="manager-script"/>
    <role rolename="manager-gui"/>
    <user password="predenmanpw"
    roles="manager-script,manager-gui,admin"
    username="predenmager"/>
</tomcat-users>
```

A.2.3 Starting and Stopping the Server

For starting and stopping the bin directory of contains the startup.sh and shutdown.sh to start and stop the server under Linux. Windows users can use startup.bat and shutdown.bat

A.2.4 Debugging

If you want to attach a debugger to Tomcat, tomcat has to be started with debugging enabled. This is done by setting some options in an environment variable and starting tomcat using a specific command.

```
JPDA_OPTS="-agentlib:jdwp=transport=dt_socket,
address=8000,server=y,suspend=n"

./catalina.sh jpda start
```

These settings allow you to attach a debugger on port 8000. If you want the server to wait for a debugger to attach change the option suspend=n to suspend=y. Both Eclipse and Netbeans are able to attach a debugger via this mechanism.

A.3 Ant

This section deals with installing and using the Ant build system. Ant is used to build and deploy the application.

A.3.1 Installation

Use the package manager to download Apache Ant sudo apt-get install ant Alternatively Ant can be downloaded from http://ant.apache.org/. Make sure to add the bin directory to your path.

A.3.2 Configuration

The ant script requires a number of variables that differ from machine from machine. These variables can be set in a build.properties file. This file should be in the same directory as the build.xml file. An example of this file is shown in listing A.3.2

Listing A.1: Ant properties file

```
manager.url=http://localhost:8080/manager/text
manager.username=predenmager
manager.password=predenmanpw
junitpath=/usr/share/java/junit4.jar
```

The manager.url option points to the url of the manager application of Tomcat. The address is `http://<domain:port>/manager/text`. When working on a development machine it is usually the same as in the example. The manager.username and manager.password variables are the same as specified in the `tomcat-users.xml` file.

The junitpath variable contains the location of `junit4.jar` which is used for unit testing. It can be installed using `sudo apt-get install junit`. Or it can be downloaded from `junit.org/`.

A.3.3 Building and Deployment

Navigate to the directory containing the build.xml file. The general syntax for building is `ant <target>` where target can be any of the following : compile, dist, deploy undeploy, list and test. Some targets depend on other targets, ant calls all dependencies before calling the actual target.

The compile target compiles all the java source files and copies all required files to the build directory. The dist target creates a war file that can be deployed to a tomcat server. The dist target depends on the compile target. The war file is put in the dist directory.

The deploy and undeploy targets deploy and undeploy the application on the tomcat server. The deploy target depends on the dist target. Typically you only need to call `ant deploy`, all the other dependencies will be called automatically.

The list target lists the applications installed on the Tomcat server. The test target runs all unit tests. Output of the unit tests is put in the `testreports` directory.

B Year Tables

This appendix gives the results per classifier per specific year.

Test Year : 2010					
Classifier	**Trained**	**Boolean Holidays**	**Correlation**	**RMSE**	**Time**
MultilayerPerceptron	2008 - 2009	false	0.9289	722.9248	03:34:812
M5P	2008 - 2009	true	0.9229	737.5203	00:07:739
M5P	2008 - 2009	false	0.9229	737.7199	00:10:479
SMOreg	2008 - 2009	true	0.9149	754.0789	19:57:480
SMOreg	2008 - 2009	false	0.9147	755.6083	24:14:800
SMOreg	2009	true	0.9143	757.6077	00:25:856
SMOreg	2009	false	0.9131	762.5642	00:24:557
MultilayerPerceptron	2009	false	0.9230	784.8000	01:53:084
MultilayerPerceptron	2009	true	0.9264	806.9679	00:36:739
MultilayerPerceptron	2008 - 2009	true	0.9156	818.5187	01:08:201
TennetForecast	Unknown	false	0.8919	864.3927	unknown
M5P	2009	false	0.9013	886.4517	00:03:330
M5P	2009	true	0.9014	886.4622	00:02:005

Test Year : 2011					
Classifier	**Trained**	**Boolean Holidays**	**Correlation**	**RMSE**	**Time**
MultilayerPerceptron	2010	true	0.9453	606.0937	00:37:297
MultilayerPerceptron	2009 - 2010	true	0.9469	629.5492	01:09:053
MultilayerPerceptron	2010	false	0.9374	630.6996	01:48:108
MultilayerPerceptron	2009 - 2010	false	0.9396	702.6959	03:33:640
M5P	2009 - 2010	true	0.9214	703.4674	00:04:351
M5P	2009 - 2010	false	0.9214	703.4738	00:07:148
M5P	2010	false	0.9165	719.7817	00:03:337
M5P	2010	true	0.9165	719.7817	00:01:958
SMOreg	2009 - 2010	true	0.9131	732.7822	24:49:712
SMOreg	2009 - 2010	false	0.9129	733.2890	19:29:195
SMOreg	2010	false	0.9113	739.1888	00:25:348
SMOreg	2010	true	0.9104	742.6221	00:31:501
TennetForecast	Unknown	false	0.8704	890.6052	unknown

Test Year : 2012					
Classifier	**Trained**	**Boolean Holidays**	**Correlation**	**RMSE**	**Time**
MultilayerPerceptron	2011	false	0.9371	646.0060	01:50:878
MultilayerPerceptron	2011	true	0.9401	650.5224	00:37:144
M5P	2010 - 2011	true	0.9252	702.1604	00:04:294
M5P	2010 - 2011	false	0.9249	703.4258	00:07:085
M5P	2011	true	0.9227	716.8347	00:01:910
M5P	2011	false	0.9224	718.1449	00:03:210
SMOreg	2010 - 2011	false	0.9099	765.5912	20:13:957
SMOreg	2011	false	0.9090	769.7450	00:31:832
MultilayerPerceptron	2010 - 2011	false	0.9393	772.7221	03:35:794
SMOreg	2010 - 2011	true	0.9072	776.2606	17:05:026
SMOreg	2011	true	0.9065	779.1002	00:30:977
MultilayerPerceptron	2010 - 2011	true	0.9482	794.1352	01:07:025
TennetForecast	Unknown	false	0.8896	846.2548	unknown

C Word Correlation Evaluation Data

Trained = Last Training Hour
Hours = Number of Hours Predicted Ahead
Top = Top Variance Percentage
W RMSE = Word Predictor Root Mean Squared Error
M RMSE = Mean Predictor Root Mean Squared Error
WC = Word Predictor Correlation
MC = Mean Predictor Correlation

Trained	Hours	Top	Count	W RMSE	M RMSE	Time	WC	MC
2012-02-29 23	4	0.2	300	1982.563	1993.470	03:53:140	0.266	0.210
2012-02-29 24	5	0.2	300	1983.666	1996.707	13:45:095	0.234	0.204
2012-02-29 25	5	0.4	300	1990.059	1996.707	18:16:074	0.220	0.204
2012-02-29 26	5	1	300	1992.347	1996.707	21:16:726	0.214	0.204
2012-02-29 27	5	0.8	300	1993.896	1996.707	25:06:250	0.211	0.204
2012-02-29 28	5	0.6	300	1994.365	1996.707	18:24:027	0.211	0.204
2012-02-29 29	5	1	600	1994.987	1996.707	11:26:316	0.209	0.204
2012-02-29 30	5	1	900	1995.061	1996.707	09:28:775	0.208	0.204
2012-02-29 31	5	0.6	900	1995.771	1996.707	09:23:945	0.208	0.204
2012-02-29 32	5	0.8	600	1995.800	1996.707	11:27:003	0.207	0.204
2012-02-29 33	5	0.8	900	1995.964	1996.707	09:25:458	0.207	0.204
2012-02-29 34	5	0.4	600	1996.391	1996.707	13:52:266	0.209	0.204
2012-02-29 35	5	0.6	600	1996.676	1996.707	13:06:213	0.207	0.204
2012-02-29 36	5	0.2	600	1997.105	1996.707	12:12:840	0.211	0.204
2012-02-29 37	4	1	300	1997.167	1993.470	03:49:898	0.227	0.210
2012-02-29 38	5	0.4	900	1997.331	1996.707	09:24:838	0.207	0.204
2012-02-29 39	5	0.2	900	1997.663	1996.707	09:23:980	0.209	0.204

Trained	Hours	Top	Count	W RMSE	M RMSE	Time	WC	MC
2012-02-29 40	4	0.4	300	2001.221	1993.470	03:49:613	0.233	0.210
2012-02-29 41	4	1	600	2001.277	1993.470	03:16:409	0.220	0.210
2012-02-29 42	4	1	900	2001.387	1993.470	06:23:841	0.220	0.210
2012-02-29 43	4	0.8	300	2001.930	1993.470	03:52:657	0.221	0.210
2012-02-29 44	4	0.6	300	2003.378	1993.470	03:57:077	0.223	0.210
2012-02-29 45	4	0.8	600	2005.362	1993.470	03:14:333	0.216	0.210
2012-02-29 46	4	0.8	900	2005.896	1993.470	06:10:423	0.215	0.210
2012-02-29 47	4	0.6	600	2008.028	1993.470	03:21:383	0.216	0.210
2012-02-29 48	4	0.6	900	2008.485	1993.470	03:06:221	0.215	0.210
2012-02-29 49	4	0.4	900	2010.918	1993.470	02:59:788	0.216	0.210
2012-02-29 50	4	0.4	600	2011.823	1993.470	03:07:150	0.215	0.210
2012-02-29 51	4	0.2	900	2015.266	1993.470	03:03:699	0.212	0.210
2012-02-29 52	4	0.2	600	2015.394	1993.470	03:16:562	0.212	0.210
2012-02-29 53	6	1	300	2021.240	1999.863	18:40:747	0.198	0.198
2012-02-29 54	6	0.8	300	2022.711	1999.863	20:39:722	0.197	0.198
2012-02-29 55	6	1	600	2022.899	1999.863	11:24:518	0.195	0.198
2012-02-29 56	6	1	900	2022.965	1999.863	09:30:784	0.195	0.198
2012-02-29 57	6	0.8	600	2023.641	1999.863	11:35:888	0.196	0.198
2012-02-29 58	6	0.8	900	2023.704	1999.863	09:27:871	0.196	0.198
2012-02-29 59	6	0.6	300	2026.732	1999.863	23:28:424	0.195	0.198
2012-02-29 60	6	0.6	900	2027.139	1999.863	09:26:205	0.195	0.198
2012-02-29 61	6	0.6	600	2027.509	1999.863	11:31:184	0.195	0.198
2012-02-29 62	6	0.4	300	2031.427	1999.863	21:10:427	0.197	0.198
2012-02-29 63	6	0.4	900	2032.334	1999.863	09:25:314	0.196	0.198
2012-02-29 64	6	0.2	600	2032.623	1999.863	11:27:323	0.209	0.198
2012-02-29 65	6	0.4	600	2032.799	1999.863	11:24:202	0.196	0.198
2012-02-29 66	6	0.2	900	2035.608	1999.863	09:25:767	0.207	0.198
2012-02-29 67	6	0.2	300	2036.784	1999.863	19:42:388	0.197	0.198
2012-02-29 68	3	1	300	2041.285	1990.331	03:30:895	0.234	0.216
2012-02-29 69	3	1	600	2047.186	1990.331	02:57:624	0.226	0.216
2012-02-29 70	3	1	900	2047.498	1990.331	03:10:959	0.225	0.216

Trained	Hours	Top	Count	W RMSE	M RMSE	Time	WC	MC
2012-02-29 71	3	0.8	300	2051.038	1990.331	03:31:597	0.226	0.216
2012-02-29 72	3	0.6	300	2055.560	1990.331	03:33:198	0.228	0.216
2012-02-29 73	3	0.8	600	2055.716	1990.331	04:12:590	0.219	0.216
2012-02-29 74	3	0.8	900	2056.294	1990.331	02:51:092	0.219	0.216
2012-02-29 75	3	0.6	600	2060.882	1990.331	02:57:221	0.219	0.216
2012-02-29 76	3	0.6	900	2061.329	1990.331	02:47:752	0.219	0.216
2012-02-29 77	3	0.4	300	2062.383	1990.331	03:41:033	0.232	0.216
2012-02-29 78	3	0.2	300	2062.394	1990.331	03:31:135	0.252	0.216
2012-02-29 79	3	0.4	900	2068.586	1990.331	02:47:017	0.218	0.216
2012-02-29 80	3	0.4	600	2068.987	1990.331	03:05:556	0.218	0.216
2012-02-29 81	3	0.2	600	2079.801	1990.331	03:08:321	0.213	0.216
2012-02-29 82	3	0.2	900	2080.849	1990.331	02:46:296	0.209	0.216
2012-02-29 83	2	1	300	2111.787	1987.501	03:32:205	0.240	0.221
2012-02-29 84	2	1	600	2120.129	1987.501	03:04:813	0.229	0.221
2012-02-29 85	2	1	900	2120.880	1987.501	02:47:190	0.228	0.221
2012-02-29 86	2	0.8	300	2126.909	1987.501	03:32:647	0.228	0.221
2012-02-29 87	2	0.8	600	2132.802	1987.501	03:11:520	0.220	0.221
2012-02-29 88	2	0.8	900	2133.474	1987.501	02:46:352	0.219	0.221
2012-02-29 89	2	0.6	300	2134.774	1987.501	03:49:712	0.228	0.221
2012-02-29 90	2	0.6	600	2139.505	1987.501	03:05:894	0.220	0.221
2012-02-29 91	2	0.6	900	2139.875	1987.501	02:53:170	0.219	0.221
2012-02-29 92	2	0.4	300	2144.294	1987.501	03:47:343	0.232	0.221
2012-02-29 93	2	0.2	300	2144.719	1987.501	03:20:377	0.257	0.221
2012-02-29 94	2	0.4	600	2149.050	1987.501	03:10:495	0.219	0.221
2012-02-29 95	2	0.4	900	2149.789	1987.501	02:48:251	0.217	0.221
2012-02-29 96	2	0.2	600	2162.416	1987.501	02:57:330	0.215	0.221
2012-02-29 97	2	0.2	900	2163.731	1987.501	02:55:496	0.213	0.221
2012-02-29 98	1	1	300	2181.201	1985.251	03:28:291	0.243	0.226
2012-02-29 99	1	1	600	2192.061	1985.251	02:54:490	0.230	0.226
2012-02-29 100	1	1	900	2193.166	1985.251	02:48:623	0.229	0.226
2012-02-29 101	1	0.8	300	2199.008	1985.251	03:29:177	0.229	0.226
2012-02-29 102	1	0.8	600	2205.260	1985.251	02:54:092	0.221	0.226
2012-02-29 103	1	0.8	900	2206.312	1985.251	02:47:601	0.220	0.226
2012-02-29 104	1	0.6	300	2207.123	1985.251	03:28:786	0.228	0.226
2012-02-29 105	1	0.6	600	2212.367	1985.251	02:57:281	0.220	0.226
2012-02-29 106	1	0.6	900	2212.800	1985.251	02:44:708	0.220	0.226
2012-02-29 107	1	0.2	300	2214.334	1985.251	03:44:948	0.257	0.226
2012-02-29 108	1	0.4	300	2218.071	1985.251	03:41:095	0.229	0.226
2012-02-29 109	1	0.4	600	2221.714	1985.251	02:55:582	0.219	0.226
2012-02-29 110	1	0.4	900	2222.199	1985.251	02:46:237	0.218	0.226
2012-02-29 111	1	0.2	600	2230.868	1985.251	02:53:407	0.219	0.226
2012-02-29 112	1	0.2	900	2233.838	1985.251	02:44:792	0.216	0.226

Glossary

ANN Artificial Neural Network. 9, 13
API Application programming interface. 6, 12

DAO Database Access Object. 15, 21
DJIA Dow Jones Industrial Average. 12

Granger causality analysis A test for determining causality between two variables. 12

IDE Integrated development environment. 21

JSON JavaScript Object Notation. 6, 17

MLP Multi Layer Perceptron. 9, 27, 29, 30

OAuth An open protocol for enabling third party authentication 2.3. 6, 7
over fitting A model is to general and also learns the noise [6]. 8

RMSE Root Mean Squared Error. 26, 27, 29

Self-Organizing Fuzzy Neural Network A feed forward neural network making use of Fuzzy Logic. 12
SVM Support Vector Machine. 8–10, 13, 27, 30

Tennet High-voltage grid operator of the Netherlands. 12
TSO Transmission System Operator: A company transporting electricity. 12
Tweet A short 140 character message on twitter. 6

UCTE Union for the Coordination of the Transmission of Electricity: An association Represents all electric TSOs in the EU. 12

Bibliography

[1] Backpropagation neural network. http://www.learnartificialneuralnetworks.com/backpropagation.html. Accessed: 2013-05-13.

[2] Introduction to backpropagation neural networks. http://cortex.snowcron.com/neural_networks.htm. Accessed: 2013-05-13.

[3] The number of hidden layers. http://www.heatonresearch.com/node/707 YEAR = 2008, AUTHOR="Jeff Heaton".

[4] Points of departure underpinning scenarios used. http://www.tennet.org/english/operational_management/transmission_services/Calculated_crossborder_cap/Points_departure_underpinning_scenarios_used.aspx. Accessed: 2013-05-11.

[5] Transparency of the electricity market. http://energieinfo.tennet.org/.

[6] Ethem Alpaydin. *Introduction to Machine Learning 2nd ed.* The MIT Press, 2010.

[7] Sitaram Asur and Bernardo A. Huberman. Predicting the future with social media. 2010.

[8] Ming-Wei Chang Bo-Juen Chen and Chih-Jen Lin. Load forecasting using support vector machines: A study on eunite competition 2001, 2001.

[9] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, pages 1–8, 2011.

[10] Morten K. Skov Daniel Ortiz-Arroyo and Quang Huynh. Accurate electricity load forecasting with artificial neural networks. *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 94 – 99, 2005.

[11] Michelle deHaaff. The number of hidden layers.

[12] Tristan Fletcher. Support vector machines explained, 2009.

[13] Ilče Georgievski, Viktoriya Degeler, Giuliano Andrea Pagani, Tuan Anh Nguyen, Alexander Lazovik, and Marco Aiello. Optimizing offices for the smart grid. 2011.

[14] Hila Becker Mor Naaman Luis Gravano. Learning similarity metrics for event identification in social media. *Proceedings of the third ACM international conference on Web search and data mining*, pages 291–300, 2010.

[15] Barbara GÅĆadysz and Dorota Kuchta. Application of regression trees in the analysis of electricity load. *Operations Research and Decisions*, 4:19–28, 2008.

[16] Bert Kok. http://twittermania.nl/2012/03/5-miljoen-nederlandstalige-tweets-dag/. http://twittermania.nl/2012/03/5-miljoen-nederlandstalige-tweets-dag. Accessed: 2013-05-20.

[17] Paras Mandal, Tomonobu Senjyu, Katsumi Uezato, and Toshihisa Funabashi. Several-hours-ahead electricity price and load forecasting using neural networks. *IEEE*, pages 2146 – 2153, 2005.

[18] NEWCOM. Social media onderzoek 2013. `http://www.newcom.nl/socialmedia`. Accessed: 2013-05-20.

[19] oseph Bonneau. http://www.lightbluetouchpaper.org/2011/02/09/measuring-password-re-use-empirically `http://www.lightbluetouchpaper.org/2011/02/09/measuring-password-re-use-empirically/`. Accessed: 2013-05-23.

[20] Geoffrey K.F. Tso and Kelvin K.W. Yau. Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks. *Elsevier*, pages 1761–1768, 2005.

[21] Young Wang and ian H. Witten. Induction of model trees for predicting continuous classes. *Proc European Conference on Machine Learning Poster Papers*, pages 128–137, 1969.