VILNIUS TECH
Vilnius Gediminas Technical University

Journal of Civil Engineering and Management
ISSN 1392-3730 / eISSN 1822-3605
2022 Volume 28 Issue 5: 408–421
https://doi.org/10.3846/jcem.2022.16012

MACHINE LEARNING FOR TEXT CLASSIFICATION IN BUILDING MANAGEMENT SYSTEMS

Jose Joaquin MESA-JIMÉNEZ[1,2], Lee STOKES[3], QingPing YANG[1], Valerie N. LIVINA[2*]

[1]*Brunel University London, Kingston Lane, Uxbridge, UK*
[2]*National Physical Laboratory, Hampton Road, Teddington, UK*
[3]*Mace Group Ltd, London, UK*

Received 28 June 2021; accepted 22 October 2021

Abstract. In building management systems (BMS), a medium building may have between 200 and 1000 sensor points. Their labels need to be translated into a naming standard so they can be automatically recognised by the BMS platform. The current industrial practices often manually translate these points into labels (this is known as the tagging process), which takes around 8 hours for every 100 points. We introduce an AI-based multi-stage text classification that translates BMS points into formatted BMS labels. After comparing five different techniques for text classification (logistic regression, random forests, XGBoost, multinomial Naive Bayes and linear support vector classification), we demonstrate that XGBoost is the top performer with 90.29% of true positives, and use the prediction confidence to filter out false positives. This approach can be applied in sensors networks in various applications, where manual free-text data pre-processing remains cumbersome.

Keywords: free-text classification, building management systems, Haystack data standard, sensor tagging.

Nomenclature

BMS – Building Management Systems;
CNN – Convolutional Neural Network;
EM – Expectation Maximisation;
HSLE – Hierarchical Label Set Expansion;
HVAC – Heating Ventilation and Air Conditioning;
ID3 – Iterative Dichotomiser 3;
MaxEnt – Maximum-Entropy Classication;
MLE – Maximum Likelihood Estimation;
NB – Naive Bayes;
NLP – Natural Language Processing;
PR-curve – Precision-Recall curve;
SVM – Support Vector Machine;
TF-IDF – Term Frequency-Inverse Document Frequency;
TSVM – Transductive Support Vector Machine;
XMTC – Extreme Multi-label Text Classication.

Introduction

When working with building management systems, there are two main problems to face. First, each BMS manufacturer has proprietary data structures and architecture. Second, there is no standard for the naming conventions that would be automatically applied to the sensors or equipment within these structures. Translating building sensor points into a naming standard is a very time-consuming task that requires highly skilled engineering knowledge. This should be done using each items name in the BMS (points) into the clean structured format (labels). The necessity of this manual step in the process is slowing the proliferation of IoT integration with existing BMS and causing large costs to companies during the BMS roll out. The purpose of this paper is to apply several machine learning methods for text classification in the context of Building Management Systems (BMSs). The mobilisation of a site implies the translation of all the different elements that are used in analysis platforms with the purposes of detecting failures of internal systems (heating, cooling), along with controlling areas of major electricity consumption and potential savings.

Most medium-to-large buildings have installed BMS which can provide valuable data to any IoT implementation. This data includes the operational states of existing

*Corresponding author. E-mail: *valerie.livina@npl.co.uk*

equipment in the building and occupancy comfort parameters for the installed sensors. In the context of BMS point tagging, we need to engage the machine learning techniques for text classification. Automatic text classification is usually done by extracting features from the text document. In this text classification problem, in the first stage the classes are different tags, which are pre-defined. The problem further departs from pre-defined classes and becomes consisting of several stages, in each of which the predicted classes are used for the next stage; thus, this becomes a semi-supervised machine learning problem. In this work we follow a generic strategy for text classification as defined in Dalal and Zaveri (2011), which has the following steps: training set of text documents, preprocessing, feature extraction, machine learning model selection, train classifier and test classifier.

1. Background

1.1. Text classification methodologies

Bayesian methodology is widely applied in text classification. Models for Naive Bayes (NB) text classification are compared in Singh et al. (2019), and finds that multi-variate Bernoulli model performs well with small vocabulary sizes, while the multinomial model performs better at larger vocabulary sizes. Similarly, Liu et al. (2002) combines the Expectation Maximisation (EM) algorithm with the NB classification method using only partial information, one class of labeled documents and a set of mixed documents, showing extremely accurate results under certain class restrictions. In research by Chai et al. (2002), Bayesian online perceptron and Gaussian processes have been implemented and tested, showing that their performance is comparable to that of Support Vector Machines (SVMs). Parallel naive Bayes algorithm is used by Liu et al. (2019) for large-scale Chinese text classification. More recent examples for this algorithm for text classification can be seen, for example in Venkatesh Ranjitha and Venkatesh Prasad (2020), where Naive Bayes is used for text classification, which lead to an advantage in terms of characteristic dialect processing. Another recent work can be found in Le et al. (2019), where this technique is also combined with sentiment lexicon. The effectiveness of this technique is further enhanced with the use of a dictionary as an input source and a document preparation process which improves the accuracy to 98.2%.

Decision trees also play an important role in text classification. In research by Hasanli and Rustamov (2019) decision trees are applied to text categorisation and classification. Random forest is an ensemble learning version of decision trees, as it constructs a multitude of decision trees at training time and outputs the class that is made of the classes. In terms of performance, Ali et al. (2012) show that the random forest gives better results than decision trees for the same number of attributes in large medical datasets. With respect to text classification, Akinyelu and Adewumi (2014) use random forests for content-based phishing detection, which yields a very high classification accuracy. Similarly, Xu et al. (2012) present an improved random forest algorithm by simultaneously employing a new feature weighting method and the tree selection methods to categorise text documents. As a result, the algorithm can effectively reduce the upper bound of the generalisation error and improve classification performance.

SVMs are very popular for text classification. As an example, Joachims (1998, 2001) shows that SVMs are appropriate for this task, and it outperforms other algorithms. Also, Tong and Koller (2001) introduce an algorithm for performing active learning with SVMs, i.e., an algorithm for choosing which instances of data to request next for the training stage. In a comparative study, Alsaleem (2011) shows that SVMs outperforms NB. In Sun et al. (2009), a comparative study on the strategies addressing imbalanced text classification using SVM classifiers is described. They evaluated 10 methods on 3 benchmark datasets using area under the PR-curve as the performance metric, finding that the standard SVM learnt the best decision surface in most test cases. In more recent studies on SVM for text classification, Gopi et al. (2020) classify tweets data based on polarity using improved RBF kernel on SVM, which outperforms other SVM-RBF classifier and models. User comments are classified using Word2Vec embedding and SVM classifier in Kurnia et al. (2020). They classify comments from social media about mobile networks applications, achieving a 79.5% accuracy. Another recent methodology on SVM for text classification can be seen in Zhang et al. (2019), where text is represented mathematically by vector space model, and the classifier is trained to classify the text based on the principle of SVM. The framework of the SVM for this classification system can be seen in Figure 1. Another interesting work by Wang et al. (2019) combines Char Convolutional Neural Networks with SVM, to obtain the emotional tendencies of users reviews. In research by Chatterjee et al. (2019) multi-class classification is performed using SVM and one-vs-rest, which divides a multi-class classification problem into one binary classification per class. On top of that, this is enhanced by using multi-threading and CUDA.

Logistic regression also provides good results in text classification. The study of Genkin et al. (2007) uses lasso logistic regression, which provides state-of-the-art text

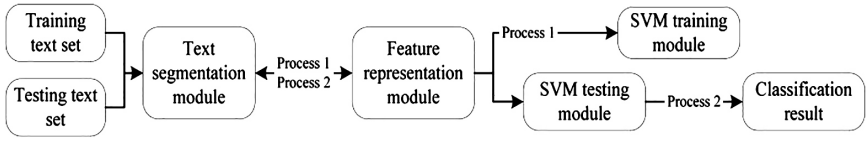

Figure 1. Framework of SVM Chinese text classification system by Zhang et al. (2019)

categorisation while producing sparse and thus efficient models. In the same way, Ifrim et al. (2008) present a coordinate-wise gradient ascent technique for learning logistic regression in the space of all n-gram sequences (contiguous sequence of n items from a given sample of text or speech) in the training data. They use several datasets, interestingly including a Chinese language dataset among them. A modified logistic regression for positive and unlabeled learning is applied by Jaskie et al. (2019), who introduce a new modified logistic regression with a variable upper bound that provides a better theoretical solution for the proposed problem. A comparison between Bayes classification and logistic regression is studied in tweets categorisation in Prabhat and Khullar (2017), showing after training that logistic regression gives a 10.1% more accurate and 4.34% more precise than the Bayes algorithm. Logistic regression and its variations are popular for sentiment analysis, as can be seen in recent studies such as Ramadhan et al. (2017), which studies tweets sentiment analysis by extracting the features first, then transforming the list of features into binary form and transformed again used Tf-idf method before being classified using logistic regression. This is very relevant for our work, as tweets have a character limitation, so this would prove that logistic regression is suitable for classification using text that is shorter than usual. Following a similar line of work, Rane and Kumar (2018) compare several methodologies, including logistic regression, SVM, Naive Bayes, AdaBoost, among others, for sentiment classification of Tweeter data for an US airline service analysis. Results show that logistic regression shows a good score with a F-measure of 81.9%, but it is outperformed by random forests in the first place, with an 86.5% F- measure. This creates a solid base for the BMS text classification case explored in this Thesis, as several of these methods are used in this work for short text classification. Also, for another work of tweet sentiment analysis, Hasanli and Rustamov (2019) compared logistic regression, Naive Bayes and SVM to detect sentiment polarity. Logistic regression and SVM show a better performance if bag-of-words is used for the pre-processing, and Naive Bayes performs better if term frequency - inverse document frequency is used.

Deep learning has been increasingly gaining popularity and the literature provides examples of using some of these methods for text classification as well. An example of this is Liu et al. (2017), which presents the first attempt at applying deep learning to extreme multi-label text classification (XMTC), with a family of new Convolutional Neural Network (CNN) models which are tailored for multi-label classification. Several large-scale datasets are constructed in Zhang et al. (2015) to show that character-level convolutional networks could achieve state-of-the-art or competitive results for text classification. Lai et al. (2015) introduce a recurrent convolutional neural network for text classification without human-designed features, showing that the proposed method outperforms the state-of-the-art methods on several datasets, particularly on document-level datasets. More recent works on text classification can be seen for example in Elnagar et al. (2020). Here, the authors perform Arabic text classification using deep learning models. Results show that attention-Gated Recurrent Units (GRUs) achieves a top performance of 96.94% by using the dataset NADiA, which is the largest dataset of Arabic documents. Another example is given by Yao et al. (2019), where a novel text classification method termed text graph convolutional networks is used with the purpose of text classification. Results of this work shows that this promising methodology outperforms other state-of-the-art deep NNs such as CNN, LSTM and others that are not NNs, such as logistic regression. The algorithms are run in various datasets, such as news or movie reviews, with a large corpus. Also, Gargiulo et al. (2019) use DNNs for hierarchical extreme multi-label text classification. They describe a methodology named Hierarchical Label Set Expansion (HLSE) used to regularize the data labels, evaluating the methodologies on the PubMed scientific articles collection, proving the usefulness of the proposed HLSE methodology. The graphical representation of the DNN model used for this work is shown in Figure 2. Deep learning methodologies proved unmistakably useful for text classification and they should be considered, as the literature suggests, for large corpus text where a lot of features need to be processed to perform the classification. For the purpose of this problem, whose length of the text is the real challenge (11 words maximum per label), DNNs are not considered. Although they can be considered for further research.

There is a growing number of interesting publications in the field of text classification: Onan et al. (2016) presented a multiobjective weighted voting ensemble classifier, which is based on a differential evolution algorithm for text sentiment classification, and it proved to be better than conventional ensemble learning methods. In study by Onan and Korukolu (2017), a feature selection model is presented based on genetic rank aggregation for text sentiment classification. In another work related to sentiment classification, Onan (2020) studies sentiment analysis based on weighted word embeddings and DNNs. Other related works in Natural Language Processing (NLP) for sentiment classification can be found in Onan (2021), Tocoglu and Onan (2020), including sarcasm identification in Onan (2019), Onan and Tocoglu (2021) and satire identification in Onan and Tocoglu (2020).

Ensemble methods are also interesting in text clssification, as they allow multiple learning algorithms to be used to obtain a better predictive performance. Onan (2018) includes an ensemble scheme based on language function analysis and feature engineering for text genre classification. Similarly, Onan (2017) uses a hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification.

As this is a multi-stage text classification problem, it is worth comparing it with similar works such as Montieri et al. (2019), which is related to the goal of this paper.

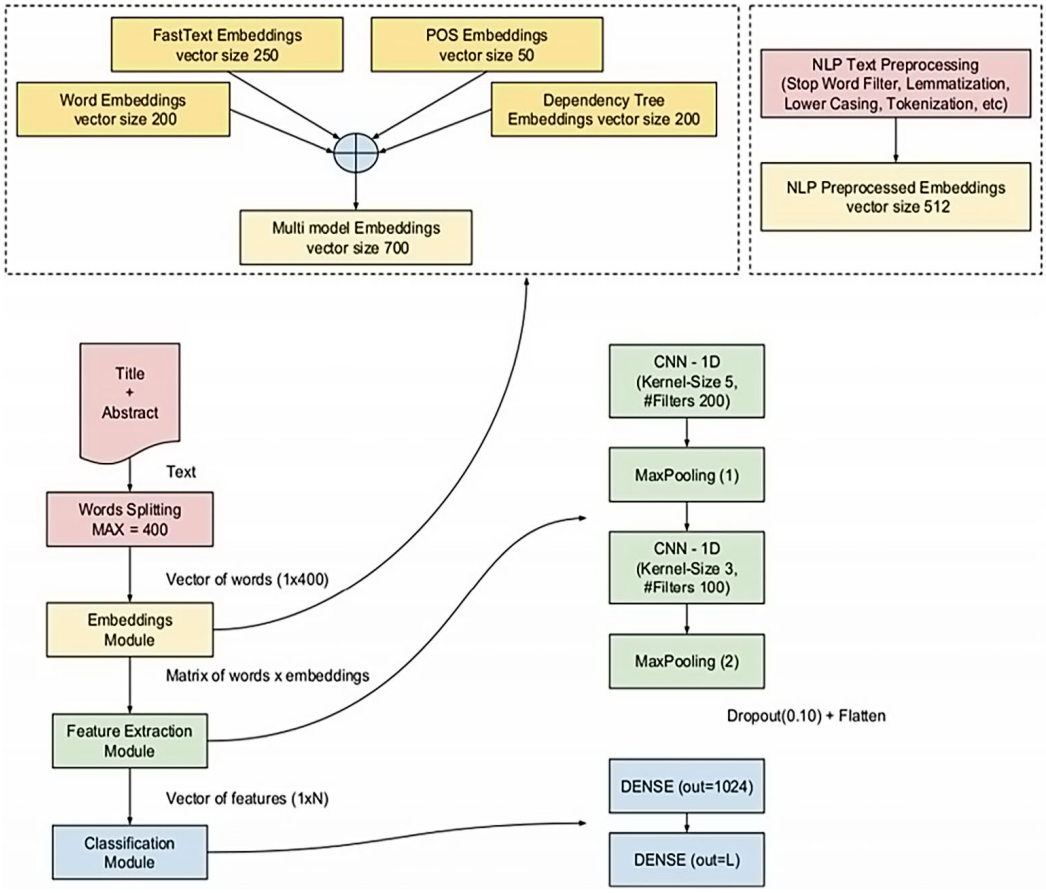

Figure 2. Graphical representation of the DNN models used by Gargiulo et al. (2019)

They use a hierarchical approach for traffic classification in the deep web. This paper first uses the flat approach in a first experiment to determine what the best performing algorithm is, and the best performing approach is then used in the multi-stage text classification problem, from the simplest to the most complex categories. On each classification level, they use several classification algorithms to select the best performing one for the next level. The final result proves that the hierarchical approach is better than the flat approach.

A summary comparing the methodologies considered during the literature review for text classification are presented in Table 1.

1.2. Text processing and categorisation

In recent years, there has been a lot of progress in natural language modelling and representation. NLP is of major interest in research as it represents the core business of Internet companies today. Language modelling is defined in Goodman (2001) as the art of determining the probability of a sequence of words and introduces the N-grams, which computes the probability of a word sequence, but if it only depends on the N previous words. Assuming that similar words appear in similar contexts, Brown et al. (1992) used counts of classes, which leads to generalisation, therefore better performance on novel data. Bag-of-words model is one of the most popular representation methods, whose statistical framework is explained in Zhang et al. (2010). It consists of the sum of one-hot codes, ignoring the order of the words, but it can be extended to bag-of-Ngrams to capture local ordering of words. Term Frequency-Inverse Document Frequency (TF-IDF) is another common technique that evaluates how important a word is to a document in a collection of corpus proportionally to the number of times it appears in the document.

In a more advanced version of text modeling there are word vectors, also known as embedding. Each word is represented by a real valued vector in N-dimensional space (usually $N = 50 – 1000$). These representations manage to capture many degrees of textual similarity. In paper by Mikolov et al. (2013a) it is shown that word vectors capture many linguistic properties (gender, tense, plurality, even semantic concepts). Following the line of work, Mikolov et al. (2013b) present two novel architectures for computing continuous vector representations of words, and they measure the quality of these representations in a word similarity task. This work introduces Word2Vec, which uses a NN model to learn associations from a large corpus of text. The representation of this architecture is shown in Figure 3. In this representation, the CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word. In Mikolov et al. (2013c), the previously

Table 1. Summary of considered text classification methodologies

Methods	Description	Characteristics
Logistic regression	Models the probability of a certain class or event existing such as pass/fail	– Easy implementation – Quick to run – Easily to extend to multiple classes – Works better with simple datasets
Random forests	Ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time	– Good handling missing data – No overfitting – Random subset of features when node split
XGBoost	Boosted tree implementation	– Trees penalization – Automatic feature selection – Trees boosting – It works well in datasets of all sizes
Multinomial NB	It implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification	– Suitable for classification with discrete number of features – Widely used in text classification
Linear SVC	It creates the classification boundary by returning a "best fit" hyperplane that separates the features	– Quick to run – Scales well with increasing number of features – Just one hyperparameter to tune

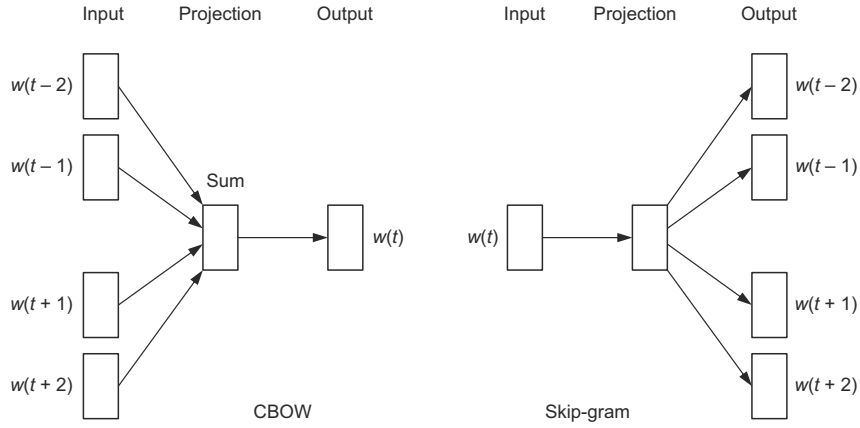

Figure 3. Model representation for vector representation by Mikolov et al. (2013b)

mentioned similarities are explored in languages for translation, reaching above 90% accuracy for the most confident translations. Although for the purpose of this problem such a deep language representation is not needed, this progress in NLP may be useful for future BMS automatic tagging improvements. In a more recent study Miaschi and Della-Orletta (2020) studied the linguistic knowledge implicitly encoded in the internal representations of BERT, a contextual language model (Devlin et al., 2019), in comparison to a contextual-independent one (Word2Vec). The findings reveal that contextual-independent model, the sum works best for obtaining sentence representations and for the contextual-dependent one, the mean works best.

In this paper, we perform text classification to create a system that tags BMS sensor data automatically. We compare several methods for text classification, by following the generic strategy of Dalal and Zaveri (2011) for solving these types of problems. We apply a bag-of-words model for feature extraction prior to the classification. The paper is organised as follows. In Section 2, we describe the problem background and the goals in more detail. In Section 3, we introduce the methods to be used for text classification. In Section 4 all the methods are compared, showing the accuracy per tag type, followed by a second experiment with the complete tagging system. Limitations and further work are provided in Section 5.

2. Problem description

The existing infrastructure uses different naming conventions for sensors and equipment of buildings using labels that are given at the moment of the installation for brief description of the component's type, location, parent-relationship, etc. (for example, Boiler 1 Temp Sensor describes a sensor that measures the temperature of the water in boiler number 1). The BMS data used in this paper was provided by a private company and cannot be disclosed in full (i.e., specific location, buildings, etc.) due to permissions restrictions.

There are several problems related to this descriptive naming system:

- There exists no naming standard. This complicates tagging when mobilising several buildings to the same analytics platform. Most of these analytics use naming conventions to create general rules that apply to all buildings.
- Some tags are incomplete. The labels are created to give a short description. This means that some words would appear shortened. Such as Temp instead of Temperature or Grnd flr instead of Ground floor.
- Duplicates. If the building is big enough, with a lot of equipment and assets, there can be duplicates in labels. This is due, for example, to upgrades of the equipment that may occur when renewing older systems, or if new areas are built.

The aim of this part of the paper is to automatically create tags, defined by the Haystack specification, based on the information provided from BMS data. A tag is a name/value pair applied to an entity (sites, equipment, sensor points, etc.) following the Haystack standard (Haystack), which is an open source initiative that standardises semantic data models with the goal of making easier to extract value from the data. Their applications include automation, control, energy, lighting, HVAC and other environmental applications (Haystack Project, 2019). A tag defines a property or attribute of an entity. Some elements are already tagged by default, whereas others need to be tagged according to some data criteria.

There are three main groups of tags on which to classify every label: Point tags, Service type and Equip tags. These are the descriptions of every main category according to Haystack, from lowest to highest level of concision:

- **Service type:** Used to classify the labels into eight main categories, namely cooling, heating, lighting, ventilation, metering, monitoring, terminals and globals.
- **Equip tags:** Refers to equipment type. Equipment is often a physical asset such as an AHU, boiler or chiller. These tags can also refer a logical grouping such as a chiller plant. There are a total of 26 equip tags including category '0', when a point does not belong to any of them. Each label must belong to one of them at least. There are never more than two categories assigned to it.
- **Point tags:** It refers to a lower level abstraction of the labels. There are 36 main types of Point tags in the dataset and they represent the most complex classification part, as there can be many tags refering to one label and even its manual assignation is difficult.

To perform the classification in the above categories, we divide the process in two experiments: first we consider only the category of tags with the highest level of concision (Point Tags) with the five classification methodologies to obtain the top- performer technique. Subsequently, we use such technique for multi-stage classification using all groups of tags in series thus, using the output of the prior classification as an input for the next one, so we add an extra feature at every stage.

2.1. Tagging experiment 1 description: Point Tags classification

In the first part of the paper we focus on the multi-label classification problem concerning the tags with the highest level of concision: point tags. The reason why the first classification problem should focus on the highest level categories (point tags) is that this will reduce the complexity of the problem by dividing it into several classification stages. Subsequently, after the dataset has been classified within the point tags categories, a second experiment is carried out that includes the rest of the main categories with the most successful technique to provide a complete solution of the tagging problem, as well as an overall accuracy result.

The data has been extracted from different versions of the same BMS controller type (Trend). The raw data is used for the purpose of training/testing the algorithms. An example of the extracted data is shown in Table 2.

The tags that we aim to assign to each row of data are defined by the Haystack specification (Haystack), and they are separated into several categories for different purposes. Some of the examples of these categories are shown in Table 3.

All points are classified as sensors, commands, or setpoints, apart from other possible assigned categories, using one of the following three tags:

- **sensor:** input, analogical/digital input, sensor
- **cmd:** output, analogical/digital output, actuator, command
- **sp:** setpoint, internal control variable, schedule.

Table 2. Extraction of BMS raw data

Label	Type	Outstation	Module	Units	Interval
IL4-6 Damper	Boolean	15	D11(Sv)	None	300
AHU2 Low Temp Hold Off SP	Numeric	12	K1(V)	°C	3600
Extact Fan28 Override	Boolean	15	W3(S)	None	3600
AHU Heating Coil	Numeric	17	D2(Sv)	None	300
		...			

Table 3. Example of tagged data (other categories have been removed from the table for simplicity)

Label	Point tags
AHU1 Dampers	sensor, damper, recirc
AHU1 Frost Stat	sensor, valve, frost
AHU1 Max Supply Temp	sp, temp, air, discharge, oneA
UPS Rm Fire Sys Fault	sensor, alarm
...	...

2.2. Tagging experiment 2 description: Complete tagging problem evaluation

For this part, all three categories are used to define the entire problem: Service type, equip tags and point tags. The three compound three separated problems, whose output is connected to the next. Because of this, the problem is solved with an increased level of complexity.

An example of data classification can be seen in Table 4. This would constitute a complete formulation of the tagging problem as it is constructed in the BMS trend system. For this part, a different result is obtained for point tags as this is a concatenation of separated problems. This work uses BMS controller data, according to the real system used at the Mitie company (U.K.). The dataset available has data from 37 buildings, from which 36 (40472 points) have been using for training and the largest building (1875 points) has been used for testing.

3. Methodology

This section describes the classification algorithms we used and how the data has been pre-processed. First, we pre-process the data and concatenate sparse matrices before the two-step model classification. The high level overview of the process is shown in Figure 4.

One of the biggest challenges in BMS data pre-processing is that the default names of the sensors (text labels) are introduced manually, therefore we can find many typos, acronyms, groups of words written together separated by upper case, etc. The text label is first separated by upper case letters, taking into consideration those which contain acronyms. Then these are converted to lower case. Next, we apply stemming, which is the process of reducing words to their stem. This refers to the roots of the words known as lemma. With this, the classification algorithm is more likely to capture similarities. Then, we categorised the result with bi-grams (groups of two words) bag of words.

Lan and outstation fields can be found together, so we separate them before categorisation. Same with controller reference, such fields contain sensor type information (I/O such as switches, temperature sensors, etc.) and data type (static, variable, etc.).

After that, all the fields with their respective vector representations, are concatenated in a matrix (a matrix in which most elements are zeros), whose rows serve as inputs for the first step. The first step aims to predict only sp, cmd or sensor, as all labels always belong to one of these three categories. Second step aims to predict the rest of the labels, by using the first step prediction, so an extra bit of information (plus the input data) is added for better accuracy.

3.1. Classification methodologies

For classification, we are dealing with multi-class and multi-label problems. There are several methods that we are comparing for the purpose of this work: logistic regression, decision trees, random forests, multinomial Naive Bayes and SVM. Considering that the text we aim to classify has the peculiarity of being very short (i.e., maximum 11 words per input), we study which methodology works better in this case.

Table 4. Example of tagging problem with all categories

Label	Service type	Equip tags	Point tags
CHW Pump 1 Enable East VT Valve HWS TEMP SETPOINT Space cooling setpoint . . .	Cooling Heating Heating Terminals . . .	cooling pump vtHeating boiler fcu . . .	sensor, run cmd, heat sp, leaving, temp, water sp, air, cool, temp, zone . . .

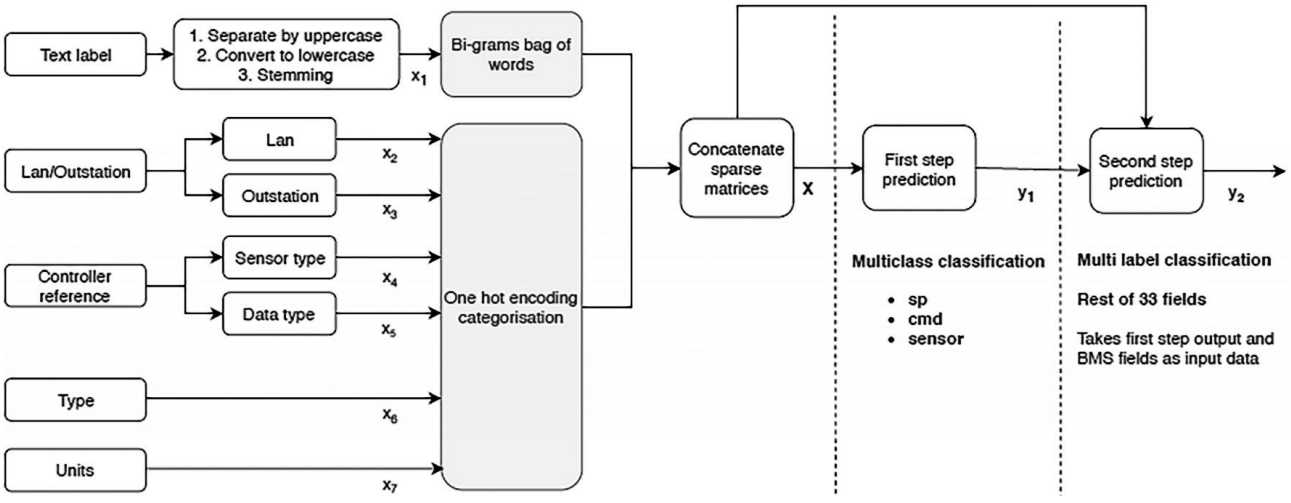

Figure 4. BMS label classification process for pre-processing data and predicting categories

From the different methodologies explored in the literature review for the text classification part, the ones explored in this Thesis are the ones whose applications are similar to the problem defined here. The text explored in this work for BMS application has a maximum length of 11 words, therefore past applications to short text classification with a high accuracy have been explored.

Other applications for short-text classification, such as the diverse tweets classifiers, utilised random forests, xgboost, logistic regression, Naive Bayes and SVM among others. These classifiers proved to deliver good accuracy for these particular problems. Therefore the methodologies explained in this section are the ones that are used to obtain the results, due to the length of the text used here.

All methods have been implemented in Python v3.7, and the library used to reproduce the algorithms has been sci-kit learn (Sci-Kit).

3.1.1. Logistic regression classification

Logistic regression, is known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. Despite its name, this linear model works as a classifier more than as a regressor. The logistic function is a monotonic function defined between 0 and 1:

$$f(x)=\frac{L}{1+e^{-k(x-x_0)}}, f(x)\in[0,1], \quad (1)$$

where x_0 is the value of the sigmoid's midpoint, L is the saturation point of the curve and k the logistic growth rate or steepness of the curve. The objective function of the logistic regression maximizes the likelihood function. The Maximum Likelihood Estimation (MLE) can be written as follows:

$$\arg\max_{\beta}: \log\left\{\prod_{i=1}^{n} P(y_i \mid x_i)^{y_i}(1-P(y_i \mid x_i))^{(1-y_i)}\right\}, \quad (2)$$

where y_i is the output between 0 and 1, $P(y_i \mid x_i)$ the posterior probability which is equal to $1/\left(1+e^{-f}\right)$, and β is the vector of weights/coefficients.

Predictor 1	Predictor 2	...	Predictor *n*	Class
a	TRUE		C	0
b	FALSE		B	1
b	TRUE	...	C	0
c	TRUE		B	0
		...		...

3.1.2. Decision trees

Decision trees build classification (or regression) models in the form of a tree structure. They break the dataset down into increasingly smaller subsets while the decision tree is incrementally developed. The resulting decision tree has decision nodes and leaf nodes (Figure 5).

The core algorithm for building decision trees is the Iterative Dichotomiser 3 (ID3), developed by Quinlan 1986). The algorithm begins with the original set, iterates on every unused attribute of the set S and calculates the entropy $H(S)$, defined as $H(S)=\sum_{x\in X} -p(x)\log_2 p(x)$, where S is the current dataset, X the set of classes in S and $p(x)$ the proportion of the number of elements in class x with respect to the number of elements in set S.

3.1.3. Random forests

Random forests or random decision forests, from Barandiaran (1998), are an ensemble learning method for classification, among others, that constructs a finite number of decision trees at training time, increasing the number of results for a better output. This ensemble method should, by definition, provide better, although sometimes very similar, results.

After training, predictions for unobserved samples x' can be made by averaging the predictions from all the individual regression trees on x', $\hat{f}=\sum_{b=1}^{B} f_b(x')$. In the case of classification trees, the alternative option is performed by taking the majority vote, also known as voting algorithm.

3.1.4. XGBoost

XGBoost is developed by Chen and Guestrin (2016). The methodology creates a scalable end-to-end tree boosting system and introduce a sparsity-aware algorithm for parallel tree learning. It uses a gradient boosting framework.

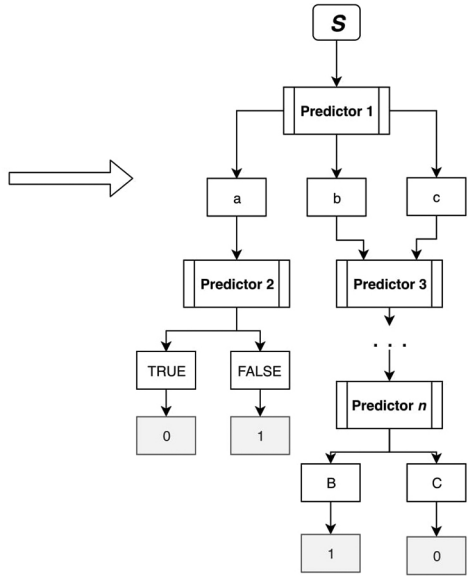

Figure 5. Generic example of decision table and decision tree

Generally, XGBoost is fast when compared to other implementations of gradient boosting. The summary of the main features is listed below:

- Regularisation: It penalises more complex models through LASSO and Ridge regularisation to prevent overfitting.
- Sparsity awareness: It automatically captures missing values depending on training loss and handles different types of sparsity patterns more efficiently.
- Cross-validation: The algorithm comes with built-in cross-validation method at each iteration, thus excluding the need to hard-code this search and to specify the number of iterations required.
- Parallelisation: It uses parallelised implementation. This is possible due to the interchangeable nature of the different loops, building many different trees in parallel. This feature allows many users to run state-of-the-art algorithm without requiring a very powerful computer.

3.1.5. Multinomial Naive Bayes

Multinomial Naive Bayes (Maron, 1961) is a specialized version of Naive Bayes that is widely used in text analysis. Whereas simple NB would model presence and absence of particular words, multinomial naive Bayes explicitly models the word counts and adjusts the underlying calculations, as explained in McCallum and Nigam (1998), combining probability distribution of P_r with fraction of documents belonging to each class for each class j and word i, at a word frequency of f_i:

$$Pr(j) \propto \pi_j \prod_{i=1}^{|V|} Pr(i \mid j)^{f_i}, \tag{3}$$

where $\pi_j = \dfrac{\text{class}_j}{\sum_{n=1}^{N} \text{class}_n}$ is the fraction of documents or labels on each class, and $|V|$ the feature space. We use the sum of logs and to smooth the probability going increasingly up when a word re-appears several times, we take the log frequency:

$$Pr(j) = \log(\pi_j) + \sum_{i=1}^{|V|} \log(1 + f_i) \log(Pr(i \mid j)). \tag{4}$$

3.1.6. Support vector machines classification

SVMs are discriminative classifiers originally defined by Vapnik and Lerner (1963), based on a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In a two-dimensional space, this hyperplane is a line dividing a plane into two parts, and each class lays on each side.

Let us consider the case of two classes. Given a training dataset of n points of the form $(\vec{x}_1, y_1) \ldots (\vec{x}_n, y_n)$, where y_i has the value of either 1 or –1, indicating the class to which $\vec{x}_i$ belongs. The goal is to find the "maximum-margin hyperplane" that divides the group of points of both classes. Any hyperplane can be written as the set of points $\vec{x}$ that satisfies:

$$\vec{w} \cdot \vec{x} - b = 0, \tag{5}$$

where $\vec{w}$ is the normal vector to the hyperplane, and $\dfrac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector. For the hard-margin case, the minimisation problem to solve is:

$$\text{minimise } \|\vec{w}\| \text{ subject to: } y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 \text{ for } i = 1,\ldots,n. \tag{6}$$

In the soft-margin case, the function we wish to minimise is:

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0.1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right)\right] + \lambda \|\vec{w}\|^2, \tag{7}$$

where the parameter λ determines the trade-off between increasing the margin size and ensuring that each point belongs to the correct side of the margin.

The way of transforming SVM to create a non-linear classifier is by means of the kernel trick, a method of using a linear classifier to solve a non-linear problem. Boser et al. (1992) suggested a way to create non linear classifiers by applying the kernel trick to maximum margin hyperplanes. The resulting algorithm is similar, except that every dot-product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space.

Rane and Kumar (2018) compare some of these methodologies for sentiment analysis. The comparison of the different methodologies for this particular work can be seen in Table 5.

Table 5. Accuracy of classifier for tweets sentiment analysis by Rane and Kumar (2018)

Classifier	Precision	Recall	F- Measure
Decision Tree	63%	64.6%	64.5%
Random Forest	85.6%	86.5%	86.5%
SVM	81.2%	84.4%	84.8%
Gaussian Naïve Bayes	64.2%	64.7%	64.6%
AdaBoost	84.5%	83.5%	86.5%
Logistic Regression	81%	81.6%	81.9%
KNN	59%	59.2%	59.3%

4. Experiments

In this section we describe both experiments. First, we perform only Point Tags classification with five different methodologies to obtain the top-performing one. In the second experiment, such methodology is used for multi-stage text classification. This means that the output obtained in the previous classification stage is used as an input for the next, so an extra feature is added at every stage.

4.1. Tagging experiment 1: Point Tags classification

The results of the experiments have been presented in two tables. Table 6 presents the train and test accuracy for the first and second classification steps. The algorithm counts an element as correctly tagged when we obtain 100% true positives and 100% true negatives per label. Table 7 presents the individual accuracy per tag type for a sample of the eight first tags and per method used.

Results in Table 6 show that for the experiments performed with our data, XGBoost algorithm provides the best result for test accuracy, closely followed by logistic regression and linear SVC. The best accuracy achieved at the second step, however, are by logistic regression, followed by XGBoost and linear SVC.

Results per tag type in Table 7 show that the accuracy per tag type varies with every different method. In fact, we can observe that every method outperforms on at least one predicted tag. Table 8 shows the runtimes for each of the steps.

One can see that linear SVC is the fastest, followed by logistic regression and XGBoost. The choice of the classification algorithm stays the same, as the runtime of less than a minute is considered good if the runtime/accuracy tradeoff is acceptable.

Assessment of errors

We aim to consider the class probability for each prediction, which is the probability for each label of belonging to a certain class, to calculate the confidence of the prediction and to discard all the elements below a certain boundary. Gneiting and Raftery (2007) provide summary measures for the evaluation of probabilistic forecasts, by assigning a numerical score based on the predictive distribution and on the event or value that materializes. The probabilities of our system do not seem to follow a clearly-defined distribution, as shown in the histogram in Figure 6.

Table 6. Results of applying five techniques and their % of absolute and relative accuracy

	LogReg	RndForests	XGBoost	MultinomialNB	LinearSVC
First Step					
Train	99.80	99.99	99.78	98.75	99.78
Test	99.73	99.07	99.84	87.69	99.50
Second Step					
Train	83.75	96.34	81.43	55.84	85.21
Test	89.01	87.14	88.68	58.24	87.64

Table 7. Results of applying five techniques to the individual tags, % of accuracy per tag type. Bold font denotes the algorithm which produced the highest score for each tag type

Tag	LogReg	RndForests	XGBoost	MultinomialNB	LinearSVC
air	98.68	98.24	97.52	98.85	97.91
alarm	99.18	98.90	98.68	96.04	99.07
chilled	99.89	99.94	100.0	88.52	99.84
co2	99.56	99.50	99.89	90.66	99.84
cool	99.89	100.0	99.89	98.74	100.0
damper	99.57	99.56	99.50	90.44	99.56
discharge	98.52	98.90	98.79	94.40	98.57
enable	100.0	100.0	100.0	95.99	100.0

Table 8. Train/Test runtime of the used methods

	LogReg	RndForests	XGBoost	MultinomialNB	LinearSVC
First Step (milliseconds)					
Train	941.28	8670.94	925.26	15.96	170.76
Test	6.00	742.35	45.88	8.98	4.99
Second Step (seconds)					
Train	6.62	200.84	20.53	0.30	2.89
Test	0.12	10.98	0.80	0.18	0.12

Therefore, we decided to create a different metric to assess and discard values based on the prediction confidence. It consists on scaling each probability from 0 to 1, so that a probability around 0.5 results in a confidence around 0. For each prediction probability, obtained as the algorithm's output, P_i, we define the confidence score, C_i, calculated as shown in Eqn (8):

$$C_i = 2 \cdot \left| P_i - 0.5 \right|. \quad (8)$$

For simplicity, we take the all tags average confidence per label. The aim of this is to filter the values by how strong the choice of the algorithm is, therefore the value of the confidence around the value 0.5 will be close to zero, but a probability close to either 0 or 1 will result in a confidence value close to 1. The resulting chart with all the confidence values of the test set can be seen in Figure 7, in which a boundary of 0.85 has been set. The reason for this boundary choice is by convenience: a higher boundary value would result in a poorer accuracy, but also in a lower percentage of false positives, whereas a lower boundary value would result in a better accuracy but more false positives.

With the chosen boundary of 0.85, the results can be found below in Table 9. Each dot of the cloud represents a particular confidence score for that particular label. The choice of the boundary is related with the strength of the decisions. A lower boundary value implies a lower value of true and false negatives, but a higher value of true and false positives.

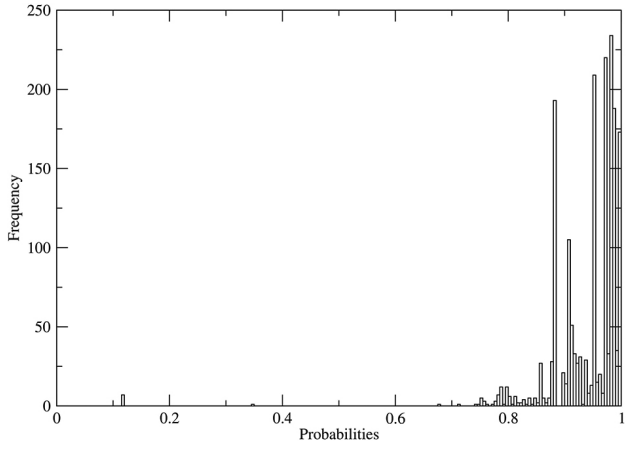

Figure 6. Prediction probabilities histogram of test set

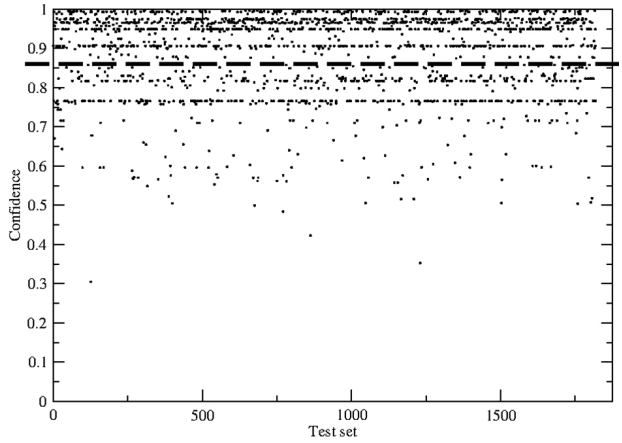

Figure 7. Prediction confidence of test set calculated according to Eqn (8). The dashed line represents the boundary, currently set to 0.85

Table 9. Percentages of true/false positives/negatives with respect to the total length of the test set, 1875 labels

	Positive	Negative
True	83.74%	3.95%
False	8.52%	3.79%

True positives are the labels that pass the boundary and whose classification is correct, true negatives are the ones that did not pass the filter because they are predicted with low confidence, but the classification is incorrect, false positives are the elements with an incorrect classification but that are not detected because they are predicted with high confidence and false negatives are the elements whose prediction is correct but with low confidence overall. As shown in Table 8, 83.74% of the test set is correctly put in the category of good predictions, whereas a 3.95% of the test set is correctly identified as misclassified elements.

4.2. Tagging experiment 2: Complete tagging problem evaluation

The complete text classification problem scheme has been illustrated in Figure 8. XGBoost methodology has been used in all prediction stages as conclusion from the previous experiment.

As can be seen here, the predictions start from the same pre-processing as the prior experiment. Then the first prediction considered has been for service type. As the results obtained for both train and test sets are above 95%, we sub-divide the output into the eight service type categories that serve individually as training inputs for the following step. The reason for doing this is to improve the chances of success for further category predictions. For example, a predicted equip tag of boiler would come from a heating service type for sure. Therefore, there is no need of training the equip tag problem according to other service categories. Then, every individual result is moved to both stages of point tag predictions as it was done in the previous experiment.

The results for every prediction stage can be seen below in Table 10.

As can be seen in Table 10, the percentage of true positives descends as the complexity of the forecast increases. Also, the number of false positives increases, as every stage inherits the errors from the previous one.

In absolute terms, the final total accuracy of the output is 90.29%. In comparison to the previous experiment, it can be seen that the accuracy results on this one are higher, this meaning that sub-dividing the problem into the eight predicted service type categories is favourable to the problem.

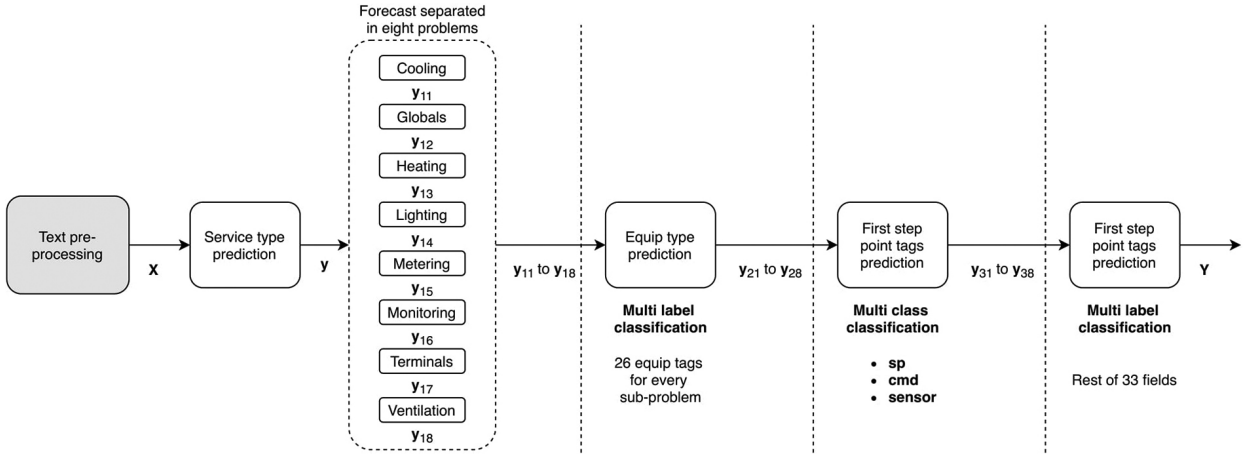

Figure 8. BMS label classification for the whole process

Table 10. Percentages of true/false positives/negatives with respect to the total length of the test set for all predictions, 1875 labels

	Service type		Equip tags		Point tags	
	Positive	Negative	Positive	Negative	Positive	Negative
True	93.85%	0.99%	92.37%	1.59%	90.29%	2.08%
False	1.09%	4.06%	4.96%	1.09%	5.45%	2.18%

5. Limitations and further work

The limitations of this work are related to feature extraction from the labels. The maximum length of each label is 11 words, with misspelt words in many occasions, therefore, to extract as much information as possible to perform classification into three categories has been the main challenge. This said, many incoherences are found in the training set, as the system relies on manually tagged data for training. This means that some engineers many use slightly different tags sometimes or simply that the information contained within the label itself is just incomplete and only compensated by personal experience, which limits the system results.

For this work, only proprietary data structure from Trend BMS controllers have been used. To further improve this model we could use other BMS types (Samsung, Tridium, etc.) as they use different fields and will probably output different results. This may be potentially a factor for a choice on which BMS to use in the future. Another potential future work could consider multi-task learning by using a multimodal deep learning architecture named DISTILLER that would allow solving the three considered problems (service, equip, tags) simultaneously (Gneiting & Raftery, 2007).

Conclusions

This paper presents a two-stage text classification study in the field of BMS. The results of the first stage show that XGBoost performs better than the other four, but the others make good candidates for this stage too, except maybe for multinomial Naive Bayes, which shows slightly worse results. The outperformer in the second stage classification, the multi label problem, is logistic regression. The top performers that follows are followed by XGBoost algorithm and, again, the Naive Bayes method performs the worst of the five. The accuracy per tag type shows that certain algorithms may be better in predicting certain tags than others. In the current paper, we have considered XGBoost and logistic regression to design the system, but the aim for further work will be a combination of methods for the second stage, using each method for doing only the classifications they are the best at, to improve the general accuracy of the whole implementation. Sub-dividing the problem into several problems improves its accuracy for the whole system as expected.

In terms of the model's deployment, the assessment of errors is very important. The main problem for this system's implementation is to locate false positive elements. The false positives are the incorrectly tagged elements that passed to the building analytics software. These elements may be difficult to detect, especially for buildings with a big number of points. Increasing the confidence boundary to a higher level may help to solve this problem and reduce false positives to a minimum. This also may reduce the number of true positives, increasing the amount of manual work.

The findings of this work open a new field of application for text classification methodologies, aiming to a scientific audience, which may explore the methodologies of this paper further to generalise this field of application for text processing and categorisation, or to industrial professionals who seek to implement this system to reduce operational tagging times from several days to a couple of minutes. Our research provides a novel multi-stage machine learning solution for the real-world BMS problem, which can be applied in several systems, or even re-trained with new standards that could appear in the future.

Acknowledgements

We would like to thank the Department for Business, Energy and Industrial Strategy of the United Kingdom and the College of Engineering, Design and Physical Sciences of Brunel University London for funding this research.

References

Akinyelu, A. A., & Adewumi, A. O. (2014). Classification of phishing email using random forest machine learning technique. *Journal of Applied Mathematics*, *2014*, 425731. https://doi.org/10.1155/2014/425731

Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random forests and decision trees. *International Journal of Computer Science Issues*, *9*(5), 272–277.

Alsaleem, S. (2011). Automated Arabic text categorization using SVM and NB. *International Arab Journal of e-Technology*, *2*(2), 124–128.

Barandiaran, I. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844. https://doi.org/10.1109/34.709601

Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144–152). ACM. https://doi.org/10.1145/130385.130401

Brown, P., Desouza, P., Mercer, R., Della Pietra, V., & Lai, J. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, *18*(4), 467–479.

Chai, K., Chieu, H., & Ng, H. T. (2002). Bayesian online classifiers for text classification and filtering. In *SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 97–104). ACM. https://doi.org/10.1145/564376.564395

Chatterjee, S., George Jose, P., & Datta, D. (2019). Text classification using SVM enhanced by multithreading and CUDA. *International Journal of Modern Education & Computer Science*, *11*(1), 11–23. https://doi.org/10.5815/ijmecs.2019.01.02

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Dalal, M., & Zaveri, M. (2011). Automatic text classification: a technical review. *International Journal of Computer Applications*, *28*(2), 37–40. https://doi.org/10.5120/3358-4633

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding*. https://arxiv.org/abs/1810.04805

Elnagar, A., Al-Debsi, R., & Einea, O. (2020). Arabic text classification using deep learning models. *Information Processing & Management*, *57*(1), 102121. https://doi.org/10.1016/j.ipm.2019.102121

Gargiulo, F., Silvestri, S., Ciampi, M., & De Pietro, G. (2019). Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, *79*, 125–138. https://doi.org/10.1016/j.asoc.2019.03.041

Genkin, A., Lewis, D., & Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. *Technometrics*, *49*(3), 291–304. https://doi.org/10.1198/004017007000000245

Gneiting, T., & Raftery, A. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, *102*, 359–378. https://doi.org/10.1198/016214506000001437

Goodman, J. (2001). A bit of progress in language modeling. *Computer Speech & Language*, *15*(4), 403–434. https://doi.org/10.1006/csla.2001.0174

Gopi, A. P., Jyothi, R. N. S., Narayana, V. L, & Sandeep, K. S. (2020). Classification of tweets data based on polarity using improved RBF kernel of SVM. *International Journal of Information Technology*. https://doi.org/10.1007/s41870-019-00409-4

Hasanli, H., & Rustamov, S. (2019). Sentiment analysis of Azerbaijani twits using logistic regression, Naive Bayes and SVM. In *2019 IEEE 13th International Conference on Application of Information and Communication Technologies* (*AICT*). IEEE. https://doi.org/10.1109/AICT47866.2019.8981793

Haystack Project. (2019). https://project-haystack.org

Ifrim, G., Bakir, G., & Weikum, G. (2008). Fast logistic regression for text categorization with variable-length n-grams. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 354–362). ACM. https://doi.org/10.1145/1401890.1401936

Jaskie, K., Elkan, C., & Spanias, A. (2019). A modified logistic regression for positive and unlabeled learning. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers* (pp. 2007–2011). IEEE. https://doi.org/10.1109/IEEECONF44664.2019.9048765

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning* (pp. 137–142). Springer. https://doi.org/10.1007/BFb0026683

Joachims, T. (2001). A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 128–136). ACM. https://doi.org/10.1145/383952.383974

Kurnia, R., Tangkuman, Y., & Girsang, A. (2020). Classification of user comment using Word2Vec and SVM classifier. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*(1), 643–648. https://doi.org/10.30534/ijatcse/2020/90912020

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 2267–2273). AAAI.

Le, C., Prasad, P., Alsadoon, A., Pham, L., & Elchouemi, A. (2019). Text classification: Naive Bayes classifier with sentiment lexicon. *IAENG International Journal of Computer Science*, *46*(2), 141–148.

Liu, B., Lee, W., Yu, P., & Li, X. (2002). Partially supervised classification of text documents. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 387–394).

Liu, J., Chang, W., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 115–124). https://doi.org/10.1145/3077136.3080834

Liu, P., Zhao, H., Teng, J., Yang, Y., Liu, Y., & Zhu, Z. (2019). Parallel Naive Bayes algorithm for large-scale Chinese text classification based on spark. *Journal of Central South University*, *26*, 1–12. https://doi.org/10.1007/s11771-019-3978-x

Maron, M. (1961). Automatic indexing: an experimental inquiry. *Journal of the ACM*, *8*(3), 404–417. https://doi.org/10.1145/321075.321084

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization* (pp. 41–48).

Miaschi, A., & Della-Orletta, F. (2020). Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP* (pp. 110–119). https://doi.org/10.18653/v1/2020.repl4nlp-1.15

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). *Efficient estimation of word representations in vector space*. https://arxiv.org/abs/1301.3781v3

Mikolov, T., Le, Q., & Sutskever, I. (2013b). *Exploiting similarities among languages for machine translation*. https://arxiv.org/abs/1309.4168

Mikolov, T., Yih, W., & Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 746–751).

Montieri, A., Ciuonzo, D., Bovenzi, G., Persico, V., & Pescape, A. (2019). A dive into the dark web: Hierarchical traffic classification of anonymity tools. *IEEE Transactions on Network Science and Engineering*, *7*(3), 1043–1054. https://doi.org/10.1109/TNSE.2019.2901994

Onan, A. (2017). Hybrid supervised clustering based ensemble scheme for text classification. *Kybernetes*, *46*(2), 330–348. https://doi.org/10.1108/K-10-2016-0300

Onan, A. (2018). An ensemble scheme based on language function analysis and feature engineering for text genre classification. *Journal of Information Science*, *44*(1), 28–47. https://doi.org/10.1177/0165551516677911

Onan, A. (2019). Topic-enriched word embeddings for sarcasm identification. In *Computer Science On-line Conference* (pp. 293–304). Springer. https://doi.org/10.1007/978-3-030-19807-7_29

Onan, A. (2020). Sentiment analysis on product reviews based on weighted word embeddings and deep neural networks. *Concurrency and Computation: Practice and Experience*, *33*(23), e5909. https://doi.org/10.1002/cpe.5909

Onan, A. (2021). Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach. *Computer Applications in Engineering Education*, *29*(3), 572–589. https://doi.org/10.1002/cae.22253

Onan, A., Korukolu, S., & Bulut, H. (2016). A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. *Expert Systems with Applications*, *62*, 1–16. https://doi.org/10.1016/j.eswa.2016.06.005

Onan, A., & Korukolu, S. (2017). A feature selection model based on genetic rank aggregation for text sentiment classification. *Journal of Information Science*, *43*(1), 25–38. https://doi.org/10.1177/0165551515613226

Onan, A., & Tocoglu, M. (2020). Satire identification in Turkish news articles based on ensemble of classifiers. *Turkish Journal of Electrical Engineering & Computer Sciences*, *28*(2), 1086–1106. https://doi.org/10.3906/elk-1907-11

Onan, A., & Tocoglu, M. (2021). A term weighted neural language model and stacked bidirectional LSTM based framework for sarcasm identification. *IEEE Access*, *9*, 7701–7722. https://doi.org/10.1109/ACCESS.2021.3049734

Prabhat, A., & Khullar, V. (2017). Sentiment classification on big data using Naïve Bayes and logistic regression. In *2017 International Conference on Computer Communication and Informatics* (*ICCCI*). IEEE. https://doi.org/10.1109/ICCCI.2017.8117734

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1007/BF00116251

Ramadhan, W., Novianty, S., & Setianingsih, S. (2017). Sentiment analysis using multinomial logistic regression. In *2017 International Conference on Control, Electronics, Renewable Energy and Communications* (*ICCREC*) (pp. 46–49). IEEE. https://doi.org/10.1109/ICCEREC.2017.8226700

Rane, A., & Kumar, A. (2018). Sentiment classification system of twitter data for us airline service analysis. In *2018 IEEE 42nd Annual Computer Software and Applications Conference* (*COMPSAC*) (Vol. 1, pp. 769–773). IEEE. https://doi.org/10.1109/COMPSAC.2018.00114

Singh, R., Kumar, B., Gaur, L., & Tyagi, A. (2019). Comparison between multinomial and Bernoulli Naïve Bayes for text classification. In *2019 International Conference on Automation, Computational and Technology Management* (*ICACTM*) (pp. 593–596). IEEE. https://doi.org/10.1109/ICACTM.2019.8776800

Sun, A., Lim, E., & Liu, Y. (2009). On strategies for imbalanced text classification using SVM: A comparative study. *Decision Support Systems*, *48*(1), 191–201. https://doi.org/10.1016/j.dss.2009.07.011

Tocoglu, M., & Onan, A. (2020). Sentiment analysis on students evaluation of higher educational institutions. In *International Conference on Intelligent and Fuzzy Systems* (pp. 1693–1700). Springer. https://doi.org/10.1007/978-3-030-51156-2_197

Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, *2*, 45–66.

Vapnik, V., & Lerner, A. (1963). Recognition of patterns with help of generalized portraits. *Avtomatika i Telemekhanika*, *24*(6), 774–780.

Venkatesh Ranjitha, K. V., & Venkatesh Prasad, B. S. (2020). Optimization scheme for text classification using machine learning Naive Bayes classifier. In A. Kumar, M. Paprzycki, & V. Gunjan (Eds.), *Lecture notes in electrical engineering: Vol. 601. ICDSMLA 2019* (pp. 576–586). Springer. https://doi.org/10.1007/978-981-15-1420-3_61

Wang, X., Sheng, Y., Deng, H., & Zhao, Z. (2019). CHARCNN-SVM for Chinese text datasets sentiment classification with data augmentation. *International Journal of Innovative Computing, Information and Control*, *15*(1), 227–246.

Xu, B., Guo, X., Ye, Y., & Cheng, J. (2012). An improved random forest classifier for text categorization. *Journal of Computing*, *7*(12), 2913–2920. https://doi.org/10.4304/jcp.7.12.2913-2920

Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370

Zhang, Y., Jin, R., & Zhou, Z. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, *1*(1–4), 43–52. https://doi.org/10.1007/s13042-010-0001-0

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28* (*NIPS 2015*) (pp. 649–657).

Zhang, M., Ai, X., & Hu, Y. (2019). Chinese text classification system on regulatory information based on SVM. In *IOP Conference Series: Earth and Environmental Science* (Vol. 252), 022133. IOP Publishing. https://doi.org/10.1088/1755-1315/252/2/022133