Baltic J. Modern Computing, Vol. 8 (2020), No. 2, 259-274
https://doi.org/10.22364/bjmc.2020.8.2.04

Document Capturing Automation: Case Study

Juris RĀTS, Inguna PEDE

RIX Technologies
Blaumaņa 5a-3, Rīga, LV-1011, Latvia

juris.rats@rixtech.lv, inguna.pede@rixtech.lv

Abstract. Capturing of new documents into the enterprise document repository involves some kind of (usually labour intensive) task of metadata attributing. Attributes include document types, folders, cases and others. This is, in fact, a classification process and may be addressed by Machine Learning based classification methods.

The research presented in this paper aims to introduce a flexible framework for document capturing automation employing Machine Learning based document classification bots and providing an intuitive user interface featuring analysis of the document classification performance, creation of the domain specific rules, and configuration or the process parameters. The paper outlines important aspects of the framework – channels, rules, handling imbalanced data and continuous training. The prototype DOBO of the framework is presented and its performance evaluation results are discussed.

Keywords: Machine Learning, Document Classification, Enterprise Content Management, Python, Elasticsearch.

1. Introduction

Machine Learning (ML) techniques have become lately a kind of omnipotent solution in wide variety of domains. If you cannot teach a computer to do something you should certainly could teach it to learn to do the thing. At least that's what people believe. Computer vision, Cancer diagnosis, recommendation systems (e.g. used by Netflix), sentiment analysis and news classification are just few domains where ML has been applied with a promising success.

One of the important domains ML is used in actively is text and document classification (some of the research results of interest for us are referenced in Chapter 2). A particular research mainly focuses the attention though on a couple of publicly available datasets to evaluate and compare a number of machine learning methods. Two important issues frequently are out of the focus:

- how to deal with text classification in data sets where the new data are added constantly and where the classification criteria tends to change over the time (like enterprise document repositories);
- how to create the generic framework that could be easily tuned for the particular data set and classification rules.

The aim of our research is to create a framework on top of the available ML and classification methods that would provide users with:

- classification bots that would learn to classify the captured documents;
- easy to use User Interface (UI), including advanced visualization tools, to evaluate the results of the document classification;
- configuration UI allowing to tune easily the ML and classification process to improve the performance.

The texts we aim to classify are those of the documents of an Enterprise Content Management (ECM) system that usually contain a large number of voluminous documents. The framework we propose spans both the implementation of the document automation process as well as the production phase.

2. Related work

Text classification methods are researched by various authors for sentiment analysis (Avinash and Sivasankar, 2019; Pahwa et al., 2018; Fu et al., 2018; Mass et al., 2016; Saif et al., 2014; Tam Hoang, 2014; Pang and Lee, 2008), news classification (Kadriu et al., 2019), web page classification (Shawon et al., 2018) etc. Some commercial mainly rule-based solutions are created for similar tasks like Xtracta (WEB, e), Serimag (WEB, b) and ABBY FlexiCapture (WEB, a).

Document classification research frequently is based on machine learning methods, some of important methods being Naïve Bayes classifier, K-Nearest Neighbours, Support Vector Machine and Deep learning models (Kowsari et al., 2019).

Machine learning methods generally work with feature sets and labels. Feature set represents an object that must be learned and predicted while label is an entity the value must be predicted for. Feature set is extracted from the source. This is handled by first splitting the text into tokens (e.g. words, tokenization) and then by converting tokens to numbers (vectorizing). One of the core approaches to the text vectorization is the Bag of Words (BoW) model. BoW has been successfully applied in broad range of domains as the medical domain with automating medical image annotation (Lauren et al., 2018), biomedical concept extraction (Dinh and Tamine, 2012), and recommender systems for medical events (Bayyapu and Dolog, 2010). Feature extraction is arguably one of the most important factors in successful machine learning projects (WEB, c).

BoW ignores the semantic information therefore several extensions of the approach are developed. This includes Bag of meta-words (BoMW) (Fu et al., 2018) that uses meta-words instead of words as building blocks. Another extension is the Bag of Embedded Words (BoEW) proposed in (Passalis and Tefas, 2018) where the word semantics is learned in so called word embeddings – vectors, created using the data on words neighbouring the current word in the text.

Various researchers (Kadriu et al., 2019; Stein et al., 2018) have analysed usage of embedding methods (Word2Vec, GloVe, FastText) and found they can improve predicting accuracy in some cases (e.g. large data volumes) and make the learning curve steeper. Another research (Avinash and Sivasankar, 2019) compares tf-idf and Doc2Vec and shows Doc2Vec has better accuracy than tf-idf for most cases.

Dimensionality reduction methods like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and non-negative matrix factorization (NMF) allow for

more efficient use of machine learning algorithms because of the time and memory complexity reduction (Kadhim, 2019; Kowsari et al., 2019; Lauren et al., 2018). This is of particular importance for the text classification of the ECM documents (as the feature sets there tend to have large dimensionality).

Support Vector Machine (SVM) is one of the most efficient machine learning algorithms (Karamizadeh et al., 2014), applied in particular to text categorization. Several drawbacks exist though as the lack of transparency in results caused by a high number of dimensions.

Deep learning networks are applied for text classification, e.g. Recurrent neural networks, Convolutional neural network, Hierarchical Attention networks (Jacovi et al., 2019; Johnson and Zhang, 2014; LeCun et al., 2015; WEB, c), as well as combined RCNN - Convolutional neural with Recurrent neural network (Lin et al, 2019). Basic methods as Naïve Bayes show good results with smaller data sets, while Convolutional neural network shows superior performance with larger data sets (Wei et al., 2018).

3. Learning Model

We use supervised-learning based classification algorithms in our research as ECM normally has large sets of labelled documents - ones already captured in the system and labelled by the users (our experience shows though that there are cases when the ECM system does not store all the label related data in its repository, this should require incorporation of non-supervised learning methods or upgrading of the source system to store the missing label data).

Main concepts of the supervised machine learning based classification process are:

- *sample* – a structure consisting of a feature set and a corresponding label class;
- *feature set* - a matrix of numeric values representing the sample;
- *label* - a category having two or more values alias classes; the label class is what has to be predicted;
- *bot* – a ML based classification algorithm that is trained to predict a label class for a given sample (feature set);
- *training sample* - used by bot for learning; bot is learning what label class corresponds to what feature set;
- *test sample* - used to evaluate the performance of the learning;
- *prediction sample* - early not seen sample that has only a feature set but no label class attached.

ML based document classification systems generally comprise four phases: feature and label extraction, dimension reduction, classifier (bot) selection (and applying), evaluation and prediction (adapted from (Kowsari et al., 2019)).

Feature extraction deals with converting the sample text to a matrix of numeric features usable for machine learning algorithms.

Dimension reduction is an optional phase that aims to reduce the dimensionality of the feature set used for training.

Evaluation. The performance of the trained algorithm is evaluated on the test data before to use it for prediction. Evaluation results may be used to select the best performing algorithms (or to decide if the algorithm has been trained sufficiently).

Prediction is the phase the rest is meant for. The trained bots are confronted with earlier unseen samples (feature sets) to predict the corresponding label classes.

Sections below describe some elaborations of this general learning model.

3.1. Channels

Incoming documents of the ECM may be captured through a set of *channels* (e.g. e-mail address defined to receive document to, or interface to external application) that are meant to group documents. That may be a good help for the document classification – in case if the channels are used by external users consistently (e.g. if the channel dedicated to the Human Resources department is not used to send in the HR unrelated documents like orders, supply agreements etc.). This should be analysed during the implementation of the framework.

If the channel data is reliable enough it may be used to determine some classification related data (e.g. documents captured through a particular e-mail address might be saved in a specific folder which means the folder is known and must not be predicted by the bot). Analysing the available channels, the list of reliable channels for the learning process should be determined. The classification bot should be implemented for every reliable channel plus a bot for all other incoming documents.

3.2. Label Extraction

ECM document classification usually involves several metadata fields to be determined (e.g. document type, folder, assignee). At least two approaches may be considered to handle this:

- Document may be classified separately for each of the fields;
- The label to predict may be related to a combination of all metadata fields (e.g. a separate label would be created for combination of document type, folder and assignee); the document is classified against the combined label.

Our experiments show the second approach is preferable as it takes less processing time (processing of each separate label takes as much time as the processing of the combined label) and the predicting accuracy for both cases is comparable.

The combined label case shows our model must handle conversion from metadata fields to label classes and vice versa. Metadata values of the document must be converted to label classes when preparing the training sample set. Label classes predicted by the classification bot have to be converted back to metadata field values.

3.3. Rules

Document classification rules of the may change. E.g. the documents previously routed to employee A are switched to employee B. These types of problems have to be addressed outside the main learning process. A set of substitution rules may be introduced. For a sample above the rules may be introduced that relate employees to responsibility domains. Responsibility domains are learned and predicted while a person a document should be routed to is determined by the appropriately specified rules. Rule example shown below says starting what date what responsibility domains must be routed to what users. The example specifies that archiving services related documents

must be routed to user1 and user2 when classifying documents created from 2016-01-01 to 2019-08-31. When classifying documents after 2019-09-01 the archiving services related documents must be routed to the user3.

2016-01-01: {“archiving services”: {“user”: [“user1”, “user2”]}}
2017-06-01: {“auditing”: {“user”: [“user2”]}}
2019-09-01: {“archiving services”: {“user”: [“user3”]}}

Another rule sample is a rule that specifies change of the attribution of the metadata to documents. The rule says that starting 2020-02-01 documents earlier attributed to folder “folder1” and case “case1” must be related to folder “folder2” and case “case2” :

2020-02-01: {“folder”: [“folder1”, “folder2”], “case”: [“case1”: “case2”]}

3.4. Handling Imbalanced Data

The analysis of the data we use for the current research shows that the top five most popular label classes (combined label document type + document folder + case) account for 94.76% of all samples while the rest (43) - for only 5.24%. The similar situation is identified in number of other ECM data sets we explored. The said means we are dealing with highly imbalanced data sets.

Imbalanced data sets can cause a number of problems for classification algorithms (Wong et al., 2011) for domains where rare cases tend to be more significant than the frequent ones (e.g. disease diagnosis and fraud detection). This does not apply for our case luckily.

The problem we have though is that rare cases have less training data (in a data sets we analysed there are classes that are used for less than 10 documents in a 5-year period). We propose the method of Major Classes (MC) to deal with this. MC specifies that only the most popular classes are used for label class prediction while the rest of classes are merged in one Others (O) class. The situation when the classification bot predicts O class for the sample is interpreted as - no prediction is made.

MC comprises four consecutive steps.

1. The set of N label classes L is ordered descending by appearance count in samples (popularity) into ordered set L^{ord}.
2. L^{ord} is split into two sets – major classes (or Majors - M_n, comprising first n classes from L^{ord}

$$M_n = (L_1^{ord}, L_2^{ord}, \dots, L_n^{ord}) \quad (1)$$

and minors

$$m_n = (L_{n+1}^{ord}, L_{n+2}^{ord}, \dots) \quad (2)$$

with total popularity

$$P(m_n) = SUM(L_i^{ord}) \ \text{i=n+1, ..., N} \quad (3)$$

not more than some a priori set *Minors Threshold* (Tm, $P(m_n) <= Tm$), such as $P(m_{n-1}) > Tm$.

3. Majors set is supplemented with Others class (O) that represents all minor classes

$$\mathsf{M}_n = (L_1^{ord}, L_2^{ord}, \dots, L_n^{ord}, O) \quad (4)$$

4. Training sample set is created for each Major class.

Said above means in our model the classification algorithm has 3 possible prediction outcomes – accurate prediction, false prediction and no prediction. It should be noted that for a number of domains no prediction may be a better result than a false prediction.

The proposed MC method allows to produce set of training data for highly imbalanced data sets as long as there are enough samples in a total data repository. For rare classes this may not be the case therefore introducing "no prediction" feature has a good ground as it can be used to configure the learning process to abandon predicting rare cases where there are no sufficient data anyway.

3.5. Continuous Training

Machine learning algorithms usually are evaluated on publicly available data sets. The evaluation consists of two steps:

- the algorithm is trained on training samples and tested on test samples;
- the algorithm is evaluated on separate sample set to make sure how it performs on new data.

If the trained models show acceptable performance they can be used later on for prediction of new data.

In an ECM repository new data is captured constantly. Moreover, even the classification rules may change with the time. The changes may be of two extremes – the changes introduced by a particular modification in the organization rules that may cause steep drop of the prediction accuracy at a time when the modification is introduced. Another kind of the change is subtler – when e.g. document vocabulary changes gradually because of new users or changes in user habits. We propose manual rules to deal with the abrupt changes. Continuous training is introduced with the subtle/gradual changes. The bots are retrained periodically (e.g. once per week) – the new data is added to the sample sets and the older data is removed.

The classification model has to provide means to control a number of parameters, like frequency to train the classificatory, volume of training samples to use, grace period (recent time period when the data from the repository is not used yet for training as prone to errors, e.g. incorrect or missing label values) etc.

Figure 1 below shows the accuracy at different time periods. Y axis shows here a percentage of true (green), false (red) and no (orange) prediction for several test runs (with test run ids indicated on x axis). The white hover text box shows percentage numbers for true, false and no prediction for a test run user moves mouse pointer over. Input fields below allow to select another test and/or another channel, buttons are for refreshing the page (if test or channel changed) and for leaving the page.

Each of the 7 tests is taken on a 1000 sample training set and the trained bots are used to predict new samples. Each next test takes data 3 months later than the previous one. The drop-in accuracy of the 4th test might mean the classification rule change

somewhere close to the end of the training period or to the start of the testing data period.

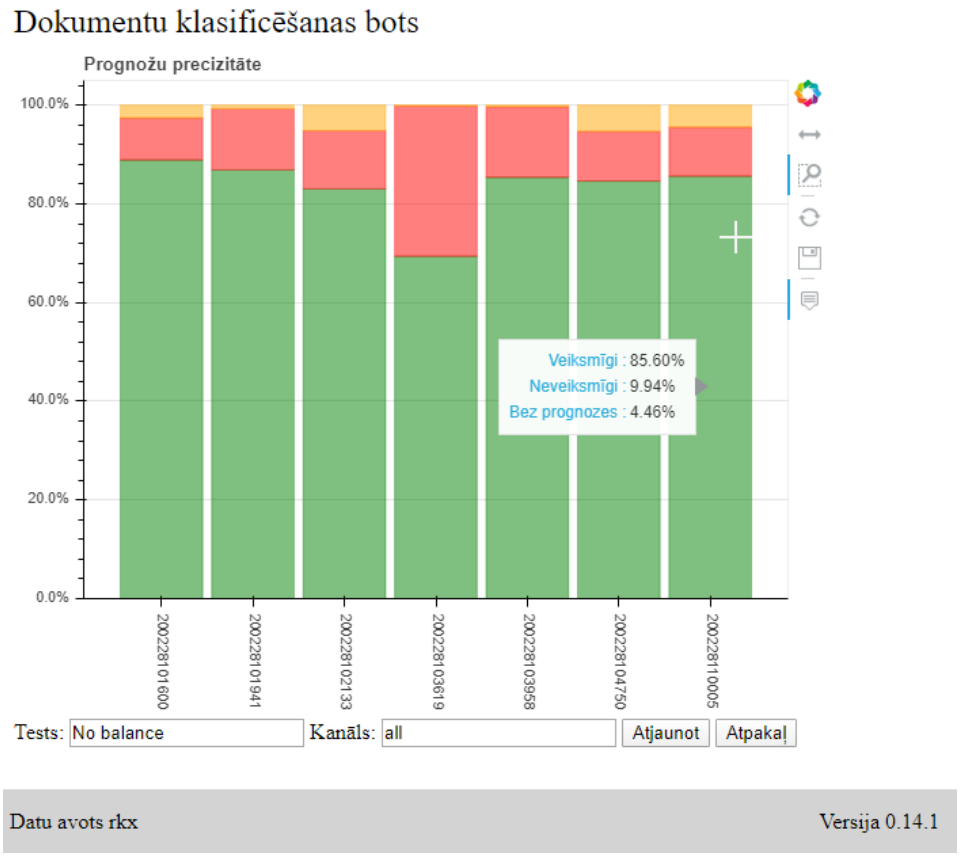

Figure 1. Prediction accuracy at different time periods.

4. The Document Classification Model

As mentioned above there is no ground to differentiate the importance of label classes for ECM document classification. Considering the learning model introduced above (chapter 3) the predictions fall into one of three categories – true predictions, false prediction or no prediction. To evaluate the test run performance we use average values of the mentioned categories – average true prediction (tp), average false prediction (fp) and average no prediction (np). Obviously

$$tp + fp + np = 100\% \tag{5}$$

The performance of the classification model can thus be fully determined by two metrics – tp and fp. Because of no prediction component the maximum tp (best configuration against tp) may relate to other configuration as minimum fp (e.g. test 1 in Figure 1 has the maximum tp while tests 4 and 5 have minimum fp). The decision of which of the two metrics to select as a criterion to compare the performance of the classification methods depends on the particular case. There should be cases when larger np rates are more acceptable than larger fp rates. The difference is that the bots know which are the np cases and thus can communicate this to the user. This is not true about the fp samples. User has to look at the document to make sure it is predicted correctly or not.

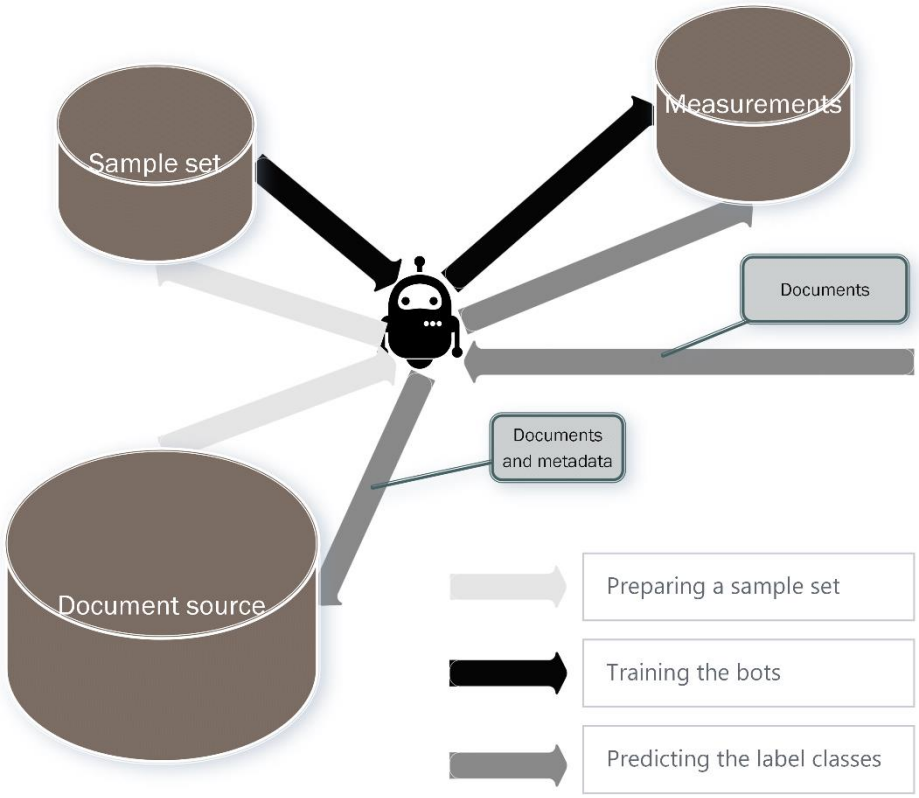

Figure 2. Document classification model

Other important features of the proposed document classification model are:

- the model handles continuous learning process as described in the previous chapter; the bots are trained periodically considering the new data and the user is empowered with advanced visualisation tools to analyse the samples

classified incorrectly by the bots and to eventually change the configuration for the next training cycles;
- the model supports multiple document classification bots, each for its own channel of document capturing;
- the model supports user defined rules for conversion between meta fields and labels and for time dependent classification criteria changes;
- the model supports a broad range of parameters, including frequency of bot retraining, grace period, sample volumes, feature extraction and classification methods (includes selection of methods and configuration of methods parameters), Minors Threshold etc.

The overall document classification model (Figure 2) comprises three main processes (preparing the samples, training the bots and predicting the label classes) and uses three storages – the Document source (the data from the source application), the Sample set and the Measurements set.

Sample preparation process is executed periodically to retrieve new samples from the main store (ECM application database) and save them into the *Samples store*. Samples are tagged with channels they relate to.

Training process retrieves (periodically) the samples for each bot, trains the bots and evaluates them. Bot performance evaluation statistics are stored in the *Measurements store*.

Trained bots listen to the respective channels and predict the metadata values when new data arrive. The prediction results are evaluated later:
- users evaluate the predictions during the grace period and correct the false predictions;
- the predictions are evaluated against the actual data.

Some of the hyperparameters supported by the model are:
- word embedding and text vectorization methods (hashing, tfidf, word2vec, fasttext);
- classification methods (Stochastic gradient descent, Multinomial naïve Bayes, Passive-aggressive classificator, Logistic regression, Support vector classifier, Linear support vector classifier, Convolutional neuron networks);
- train and test sample volume (number of samples for training and evaluation);
- Minors Threshold (see chapter 3.4);
- Stop words.

5. DOBO prototype

Web application DOBO (DOcument classification BOt) has been developed to prototype the proposed model. The prototype implements all three processes described above (chapter 4) and supports the following features:
- interactive configuring of hyperparameters;
- execution of classification tests;
- browsing of false predicted, no predicted and true predicted samples;

- grouping and filtering of samples by actual and/or predicted label classes;
- side-by-side comparing the document texts;
- side-by side comparing of two test runs;
- histogram of performance of several test runs (Figure 1);
- date histogram of performance metrics for the samples of the given test run against the time periods of the sample creation date/time (this aims to spot time periods of low prediction performance that might be caused by the changes in organisations document classification criteria).

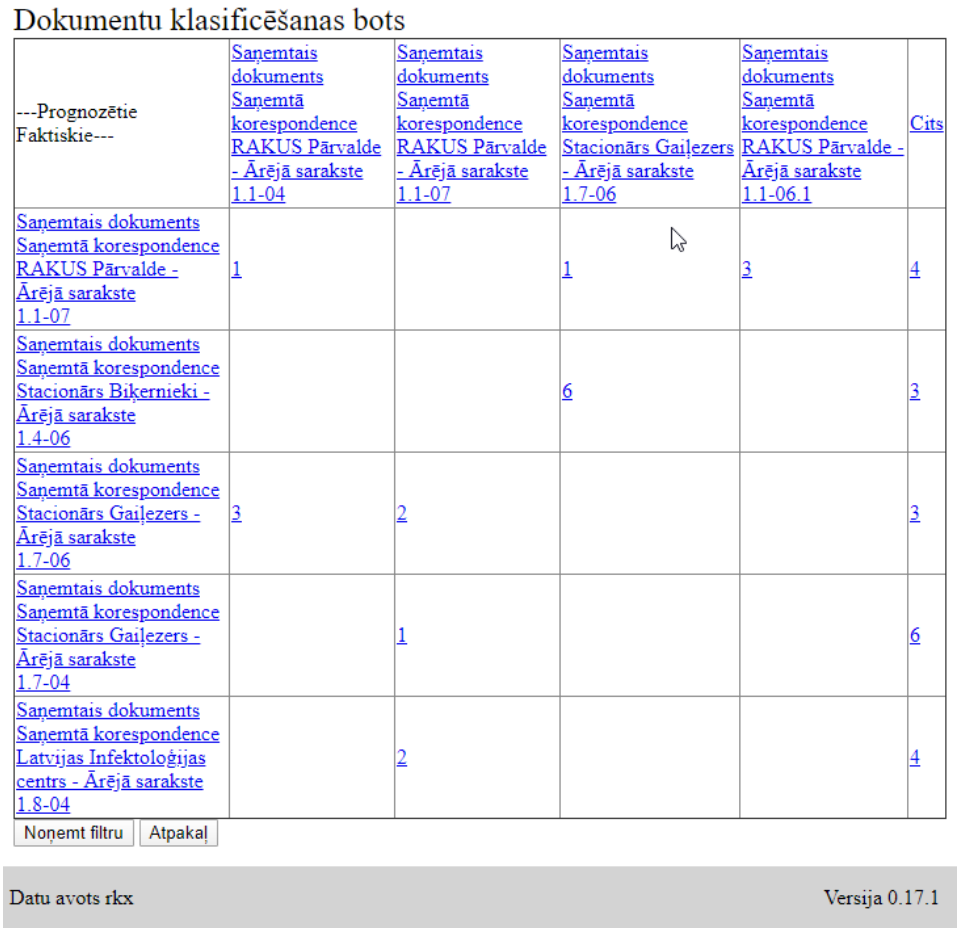

Figure 3. Samples grouping against actual and/or predicted label classes.

Figure 3 shows the DOBO page for sample grouping and filtering. Row headers show actual label classes and column headers – predicted label classes. Cells show the number of samples of the actual label class of the row that has been predicted the label class of the column. The page shows only false and no predictions (no predictions are on

the rightmost column). If user clicks on the cell the related samples are filtered and listed on the main page for further analysis. User may click on a row or column header as well to filter samples related to the respective actual or predicted label class. Two buttons below allow to remove a filter and to leave the page.

Figure 4 shows the date histogram for true prediction of a particular test run. The vertical axis shows average true prediction (tp) while circle size visualizes sample count for a given time period. X axis shows the time, buttons below allow to switch time interval of the date histogram to days and to leave the page.

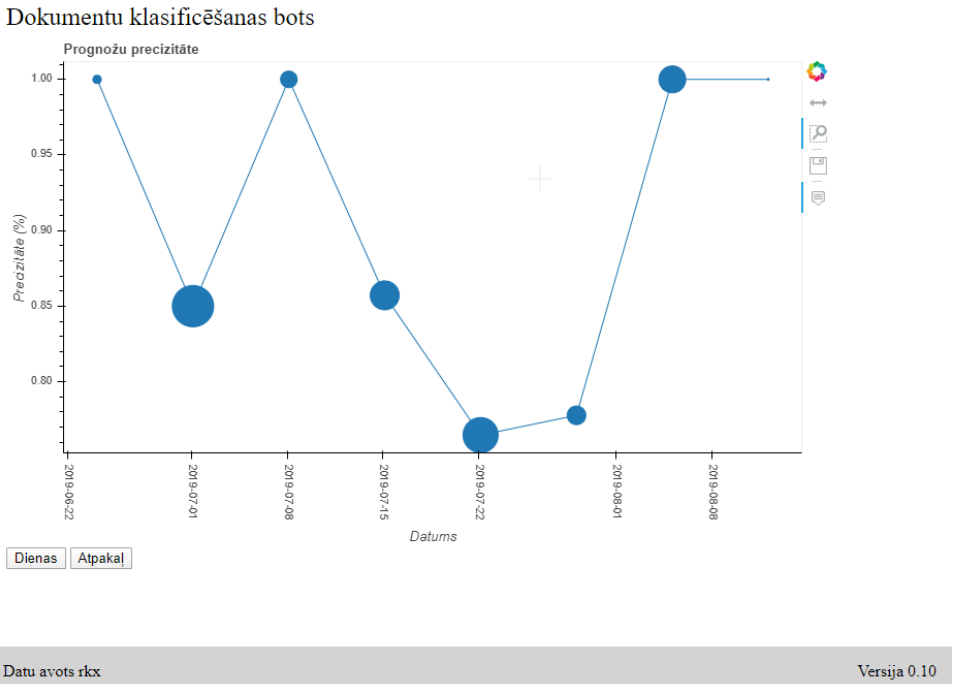

Figure 4. Performance date histogram for the test run.

Python (with the machine learning libraries scikit learn, keras and tensorflow, as well as visualization library bokeh), Elasticsearch and Kibana is a technology platform used for the implementation of the DOBO prototype.

6. Performance Evaluation

The performance of the learning model was evaluated on a data set containing more than 160 thousand of documents (half of them digitalised). Combined label (document type + document folder + case) is used for the classification.

Seven classification methods and four vectorisation methods were tested for a number of hyperparameter values. Other parameters explored are volume of the training set, number of features selected for text representation, analysis of bigrams and trigrams, learning in minibatches (partial fit) vs learning in one go, Minors Threshold etc.

Figure 5 demonstrates influence of Minors Threshold. Rightmost bar (value 0) represents the case when bot attempts to predict values of all label classes while the leftmost (value 0.2) – when bot ignores (does not predict) the minority classes accounting in total for 20% of all samples.

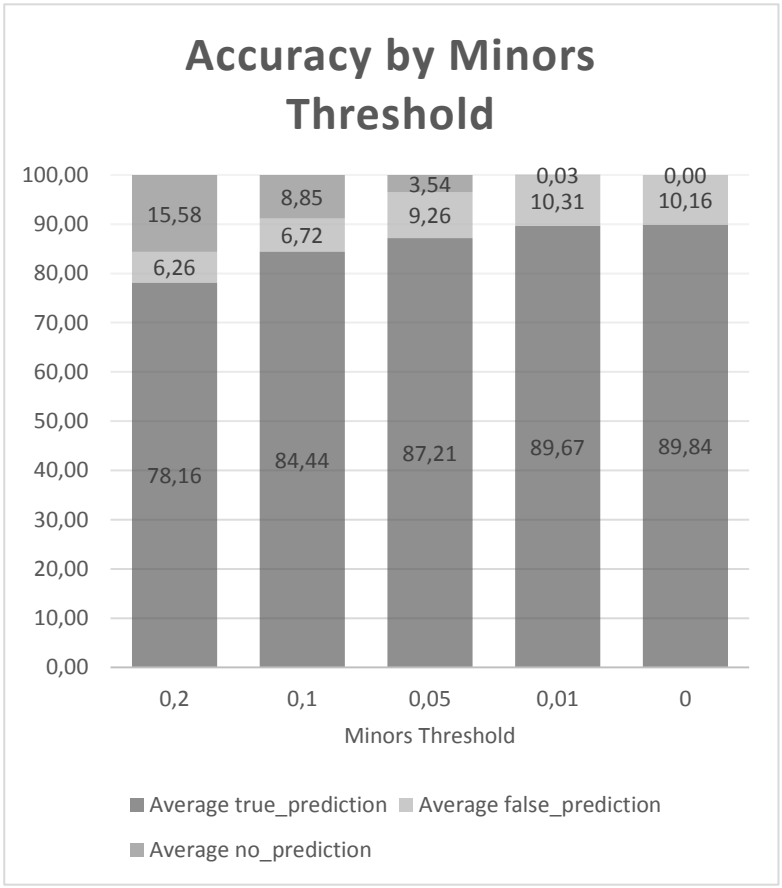

Figure 5. Prediction accuracy by Minors Threshold.

The 0.2 Minors Threshold has the worst true prediction value but it has as well the best (lowest) false prediction value. The no prediction part has an important advantage in comparison to the false predictions – it is known they have to be addressed by somebody. This means they can be routed to a particular workflow for handling.

One of the significant conclusions from the test measurements is that the accuracy of the document classification bots trained with different classification methods does not differ significantly (see Figure 6).

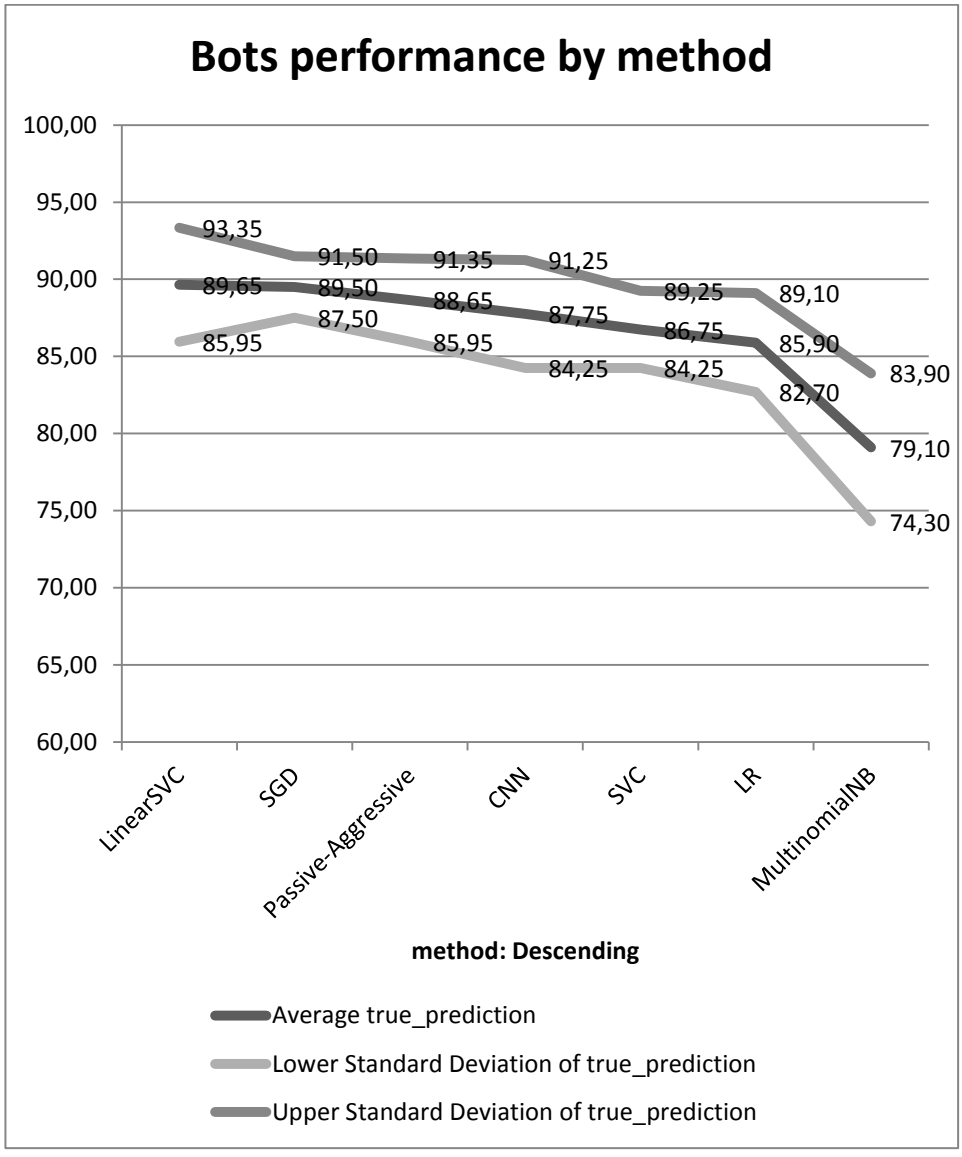

Figure 6. Prediction accuracy by method.

Multinomial Naïve Bayes (MultinomialNB) has lower accuracy here but this may change with new documents captured. Accuracy of other methods does not differ more than the standard deviation boundaries. Every next learning cycle may have its own best methods. This is why we aimed to create a flexible framework that allows for easy configuration of the learning process both when deploying and when maintaining the process in production.

It should be noted that the deep learning methods like multi-layered Convolution Neuron Network (CNN) did not show better results than basic methods like Stochastic Gradient Descent (SGD), Support Vector Classifier (SVC), Linear SVC, Passive-Aggressive Classifier and Logistic Regression (LR).

Two more data sets have been tested for comparison. The results were similar to the ones revealed above.

7. Conclusions and future work

The proposed document classification model is based on the following observations from the domain of research:

- Documents are captured by ECM system through channels, each channel is dedicated to handle its own specific set of documents;
- Label class distribution of a document set is highly imbalanced;
- The classification rules are inexact and subject to change.

Machine learning based document classification process has to be configured to work properly. The configuration includes a selection of classification and vectorisation methods, tuning of hyperparameters of said methods, the configuration of a number of parameters important for the learning process. The measurements we run did not identify any significant advantages of any particular classification method. We have a ground to believe, in the contrary, that methods and hyperparameters should be selected for the particular case (e.g. ECM document management process). Moreover – it is necessary to periodically analyse the performance of the classification model and to tune the configuration while in production to compensate for changes of the classification rules.

The document classification framework prototype DOBO has been developed and evaluated on a 160 thousand document strong ECM document repository. The findings are as follows:

- number of ML based document classification methods are available still they have to be tuned for the data set and the enterprises business process to provide acceptable classification performance;
- document classification rules change over time and this has to be handled by the classification bots; the analysis of the classification of the prediction performance over time may help to spot the changes in the classification criteria;
- the proposed model and tools allow to specify time dependent rules of classification rule changes over time; this can be used by the classification bots to improve the prediction accuracy.

The results of our research support the conclusion made by several authors (e.g. Faggella, 2019) that the feature extraction is the most important phase of the machine learning based document classification process. The means should be incorporated into our model to allow further configuration of the feature extraction. Ontology based feature extraction (Kolle et al., 2018) is one of possible direction to proceed. Another direction to proceed is incorporating of ensemble learning methods.

List of abbreviations

DOBO	DOcument classification BOt (Chapter 5).
ECM	Enterprise Content Management system (Chapter 1).
MC	Major Classes (section 3.4) are a set of most popular label classes that are predicted by the trained bots.
ML	Machine Learning (Chapter 1).
O	Other class (section 3.4) is a merged class of all minor classes that is not predicted by the trained bots.
UI	User interface (Chapter 1).

Acknowledgements

The research has received funding from the project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/18/A/003.

References

Avinash, M., Sivasankar, E. (2019). A Study of Feature Extraction Techniques for Sentiment Analysis. : 1–12.

Bayyapu, K.R., Dolog, P. (2010). Tag and Neighbour Based Recommender System for Medical Events. In: Proceedings of the First International Workshop on Web Science and Information Exchange in the Medical Web, MedEx 2010, APA, 14–24.

Dinh, D., Tamine, L. (2012). Towards a Context Sensitive Approach to Searching Information Based on Domain Specific Knowledge Sources. Journal of Web Semantics 12–13: 41–52.

Fu, M., Hong Qu, Li Huang, Li Lu. (2018). Bag of Meta-Words: A Novel Method to Represent Document for the Sentiment Classification. In: Expert Systems with Applications 113: 33–43.

Jacovi, A., Shalom, O.S., Goldberg, Y. (2019). Understanding Convolutional Neural Networks for Text Classification. : 56–65.

Johnson, R., Zhang, T. (2014). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks.

Kadhim, A.I. (2019). Survey on Supervised Machine Learning Techniques for Automatic Text Classification. In: Artificial Intelligence Review 52(1): 273–92.

Kadriu, A., Abazi, L., Abazi, H. (2019). Albanian Text Classification: Bag of Words Model and Word Analogies. In: Business Systems Research Journal 10(1): 74–87.

Karamizadeh, S., Abdullah, S. M., Halimi, M., Shayan, J., Rajabi, M. J. (2014). Advantage and Drawback of Support Vector Machine Functionality. In I4CT 2014 - 1st International Conference on Computer, Communications, and Control Technology, Proceedings, Institute of Electrical and Electronics Engineers Inc., 63–65.

Kolle, P., Bhagat, S., Zade, S., Dand, B., & Lifna, C. S. (2018). Ontology Based Domain Dictionary. In 2018 International Conference on Smart City and Emerging Technology (ICSCET), IEEE, 1–4.

Kowsari, K., Meimandi, K.J., Heidarysafa, M., Mendu, S., Barnes, L.E., Brown, D.E. (2019). Text Classification Algorithms: A Survey. Information (Switzerland) 10(4), 1-68.

Lauren, P, Guangzhi Qu, Feng Zhang, Lendasse, A. (2018). Discriminant Document Embeddings with an Extreme Learning Machine for Classifying Clinical Narratives. In: Neurocomputing 277: 129–38.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep Learning. In: Nature 521(7553): 436–44.

Lin, R., Fu, C., Mao, C., Wei, J., Li, J. (2019). Academic News Text Classification Model Based on Attention Mechanism and RCNN. In: Computer Supported Cooperative Work and Social Computing, 507–516.

Maas, A.L. Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C. (2016). Learning Word Vectors for Sentiment Analysis. In: European Review for Medical and Pharmacological Sciences (January 2011): 9.

Pahwa, B., Taruna, S., Kasliwal, N. (2018). Sentiment Analysis- Strategy for Text Pre-Processing. International Journal of Computer Applications 180(34): 15–18.

Pang, B., Lee, L. (2008). Opinion Mining and Sentiment Analysis: Foundations and Trends in Information Retrieval. In: Foundations and Trends in Information Retrieval Vol. 2, No 1-2 (2008) 1–135.

Passalis, N., Tefas, A. (2018). Learning Bag-of-Embedded-Words Representations for Textual Information Retrieval. Pattern Recognition 81.

Saif, H., Fernandez, M., He, Y., Alani, H. On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter. ResearchGate.

Shawon, A., Zuhori, S.T., Mahmud, F., Rahman, J. (2018). Website Classification Using Word Based Multiple N-Gram Models and Random Search Oriented Feature Parameters. In 2018 21st Int. Conf. of Computer and Information Technology (ICCIT), IEEE, 1–6.

Stein, R.A., Jaques, P.A., Valiati, J.F. (2018). An Analysis of Hierarchical Text Classification Using Word Embeddings. Cornell University.

Tam Hoang, D. (2014). Sentiment Analysis: Polarity Dataset. Charles University in Prague.

Wei, F., Qin, H., Ye, S., Zhao, H. (2018). Empirical Study of Deep Learning for Text Classification in Legal Document Review. In: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 3317–20.

Wong, A., Kamel, M.S., Sun, Y., Wong, A.K.C. (2011). Classification of Imbalanced Data: A Review Pattern-Directed Aligned Pattern Clustering View Project Pattern Discovery in Gene Expression Data View Project CLASSIFICATION OF IMBALANCED DATA: A REVIEW. In: International Journal of Pattern Recognition and Artificial Intelligence 23(4). www.worldscientific.com (July 17, 2019).

WEB (a). Intelligent Document Processing Platform - ABBYY FlexiCapture, https://www.abbyy.com/en-eu/flexicapture/

WEB (b). Serimag - Artificial Intelligence for Document Automation, https://serimag.com/en/

WEB (c). Faggella, D. What is Machine Learning? https://emerj.com/ai-glossaryterms/what-is-machine-learning/

WEB (d). Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E. Hierarchical Attention Networks for Document Classification, https://www.cs.cmu.edu/~diyiy/docs/naacl16.pdf

WEB (e). Xtracta: Automated Data Entry Software Powered by AI, https://xtracta.com/

Received March 16, 2020, accepted April 28, 2020