Baltic J. Modern Computing, Vol. 10 (2022), No. 3, 545-559
https://doi.org/10.22364/bjmc.2022.10.3.25

Using a Topic Based Model to Automate Indexing and Routing of Incoming Enterprise Documents

Juris RĀTS, Inguna PEDE

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

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

Abstract. A machine learning based model aimed at automation of the indexing and routing of the incoming documents of the enterprise is proposed in this article. An overall automation process as well as support for trainset annotation and a model for handling of streams of incoming documents is described. Experts are supported during the annotation process by grouping the stream of documents into clusters of similar documents. It is expected that this may improve both the process of topic selection and that of document annotation. A binary classification-based model for topic prediction is proposed and analysed. Classification bots are trained for each of the largest topics and executed on incoming documents afterward. A number of model parameters are described and analysed.

Keywords: Machine Learning, Text Clustering, Document Classification, More Like This query.

1. Introduction

Machine Learning (ML) techniques are actively used lately in a wide variety of domains. Computer vision, Cancer diagnosis, recommendation systems (e.g. used by Netflix), sentiment analysis and news classification are just a few examples to mention. ML is used in particular for classification of texts and documents. The research is mainly focused though on comparing performance of machine learning methods on a number of publicly available static datasets.

The objective of our research is to create a ML based model for handling of incoming documents of an enterprise. The main idea is to classify documents by topics and use further the document topic (in concert with other metadata like sender and addressee) to properly index and route the document.

Several challenges have to be addressed here:

- we are not dealing here with a static document set – it expands constantly;
- the document set is highly unbalanced;
- the penalty for false positives is probably higher than that for false negatives;
- supervised ML methods need appropriate annotated trainsets for topic prediction; experts need to be supported both to create a topic set and to annotate documents.

We outline here the overall architecture of the proposed model. Doing this we focus on two processes:

- supporting a user while annotating the samples for training of ML models;
- training the ML models and using the trained models for predicting the topics for new incoming documents.

Chapter 2 discusses the related work, Chapter 3 outlines the overall model architecture, Chapter 4 deals with the proposed solution for support of trainset annotation. Chapter 5 gives the overall description of process for training binary bots and using them for topic prediction. Chapter 6 describes in more detail the process of training a particular bot. Chapter 7 gives the conclusions.

2. Related Work

Supervised Machine Learning (ML) methods are used widely for text classification. Some domains of interest are sentiment analysis (Avinash and Sivasankar, 2019), news classification and web page classification (Shawon et al., 2018).

Supervised ML methods generally deal with features and labels. A feature set is a representation (numeric array or vector) of the object (e.g. a text or document) to be classified while a label is what has to be predicted by a classification bot. ML method is at first trained on a trainset of features and labels. The trained model is then used to predict labels for new objects.

Classification solutions can be represented as two pipelines – one for model training and another – for use of trained models to predict labels for new objects. Training pipeline comprises data cleaning, feature embedding, feature reduction, model training and validation. Prediction pipeline spans data cleaning, feature embedding, feature reduction and label prediction. It is necessary to prepare in advance the trainset for all this to work.

Data cleaning usually involves removing unnecessary for classification data – e.g. numbers, e-mail addresses, stop words. The cleaned text is then processed by some of the embedding methods that convert text to a numeric vector.

One of the basic embedding methods for text objects is the Bag of Words (BoW). BoW has been successfully applied in a 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). Examples of methods based on BoW model are CountingVectorizer and tfidf. Both methods create the vocabulary of all words of the documents corpus and base the text representation on this vocabulary.

Several extensions of the BoW model 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.

The main disadvantage of BoW models is they rely on unrealistic assumption that words occur in text independently one of other. A number of alternative models have been developed that use the context (surrounding words) of a word to create word embedding. This allows to create embeddings putting words with a similar meaning

close to one another in a representation space. The assumption here is that words that have similar context have to have similar meaning. Examples of such embedding methods are GloVe, word2vec, doc2vec, fasttext and others (Pennington et al., 2014). These are followed by so called contextual models BERT, ELMo, GPT, XLNet (Liu et al., 2020; Ular and Robnik-Šikorja, 2020), Sentence-BERT (Cygan, 20221) and others.

Advanced embedding methods like BERT, GPT and XLNet have demonstrated good results for a number of tasks still there is no single winner for all cases. A convenient method must be selected for a domain and task to solve.

Dimensionality reduction is a critical step of the text classification pipeline. The methods used include PCA (Principal Components Analysis) (Pennington et al., 2014), truncatedSVD (Hansen, 1987) and UMAP (Uniform Manifold Approximation and Projection) (Cygan, 2021). Those methods allow for more efficient use of machine learning algorithms because of the time and memory complexity reduction (Lauren et al., 2018). This is of particular importance for the text classification of the enterprise documents (as the feature sets there tend to have large dimensionality).

Deep learning networks are applied for text classification, e.g. Recurrent neural networks, Convolutional neural network, Hierarchical Attention networks (Jacovi et al., 2018), 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).

Supervised learning is superior for text classification tasks. The better performance of supervised learning algorithms is based on knowledge accumulated in annotated trainsets. The disadvantage here is that those annotated trainsets need to be prepared that takes considerable resources. It is therefore of great interest to develop methods that could reduce the manpower necessary for trainset annotation.

Various approaches to text classification based on clustering (i.e. unsupervised learning) methods have been researched (Baker, 1998; Fernández et al., 2016; Kyriakopoulou, 2008; Kyriakopoulou and Kalamboukis, 2007). Researchers propose to add clustering data to the text representation or classify text by clusters (Brendel, 2020; Kyriakopoulou and Kalamboukis, 2007). Unfortunately, the clustering methods do not perform well for high dimensionality data (Assent and Seidl, 2009; Parsons et al., 2004; Steinbach et al., 2004). Text clusterization features both high dimensionality and large numbers of clusters (Karpov and Goroslavskiy, 2012). Still clustering could be used to support annotation of the training data of the supervised method.

The mainstream approach to text classification is the supervised machine learning. Still there are alternatives. One of them we came across is the contextual More Like This query (MLT) provided by Elasticsearch (Klinger, 2019). This approach has been applied for phrase classification (Yellai, 2016), recommendation engine and duplicate detection (Vola, 2017). We did not find any references to the research on application of MLT to the document classification.

Document classification solutions are mainly built upon the static model – a classification bot is trained and validated on a given labelled document set and then applied to new documents. It must be assumed that important features of the documents do not change over time for this model to work. We acknowledge that the documents form a stream rather than a fixed set and that there might be changes over time both in document dispersion by topics and even the document features inside a topic may

gradually change. We propose to use iterative process to address this challenge. Trainsets of the topic identification bots are constantly updated with the new documents and the bots are retrained on the new data over and over again.

3. Overall Process

Our approach to the automation of handling of enterprise incoming documents is based on the following assumptions (supported by observations in several organizations):

- documents belong to a large number of topics, still the document set is highly unbalanced in this regard – a handful of topics cover most of the document amount;
- document topics and distribution of documents between topics may change over time;
- documents may be lengthy and have various layouts with each part having a different role.

Unbalanced nature of the document set means, in particular, that most of the topics have a rather small number of documents belonging to them. This makes them inconvenient for automation because of a lack of training data. We propose thus to focus on automation of handling of the largest topics which should span a major part of all documents.

Sets of annotated training samples (samples with topics assigned) must be created in advance to use supervised ML methods for the classification. Experts face a number of challenges when creating the annotated trainset:

- a convenient set of topics must be created;
- a sufficient amount (to train classification method) of samples must be provided with correct labels

We assume that it is possible to create a set of topics such that a topic (possibly in combination with document sender and/or receiver data) unequivocally determines the indexing and routing of the document. Document sender/receiver may be identified by other means (out of scope of this research) therefore as long as the topic of the document is identified, the handling of the incoming document is determined.

We propose to create a set of topics and annotate some historical documents beforehand (setup process). This allows us to understand what are the main topics and to create initial trainsets for them. When ready with the setup process we switch to the main process where new incoming documents are handled by the model. Details of both processes follow below.

3.1. The Setup Process

Setup process involves experts to annotate a set of historical documents of the organization. The main goal here is to create topics that:

- unequivocally determine the handling of its documents;
- are separated well enough one from another.

The first depends fully on expert's domain knowledge. We aim to support the second by manipulating the order of documents presented to experts. Namely – we would like to increase the likelihood that the next document in the stream of documents presented to

expert has the same topic as the current one. At the highest extreme this would mean that when expert assigns a topic to a particular document he is presented with the next document of the same topic and this continues as long as there are documents of the topic available. The extreme currently is not achievable because we would need a method of automatic text labelling to do this. Still increasing the likelihood that the next document has the same topic we will be able to create longer sequences of the same topic documents and this will help both to create a topic set and to annotate the documents.

Two approaches for determining the next document (that would increase the likelihood it will have the same topic) are explored and compared in this research:

- clustering the documents in advance and selecting the next document from the same cluster as the current one;
- using MLT to find the next document.

The setup process should result in creation of an initial set of the largest topics (we call them major topics) and sets of annotated trainsets for them. The set of major topics as well as the annotated trainsets are later maintained (expanded, updated) by the main process.

3.2. The Main Process

The main process uses a set of rules that specifies actions for specific topics (or combination of topics and/or senders and/or addressees). An action specifies suggestion for the metadata to be assigned to the document (including the routing data). If the topic is assigned to the document by a bot the suggested metadata values are attached and the document is routed to the suggested assignee. The assignee then has a choice to approve or reject the assignment. In the latest case the document is routed back to the clerk responsible for the document indexing who provides the correct topic. In both cases the document is added to the trainset to be used in next training sessions of related classification bots.

The main process uses supervised machine learning methods to train a classification bot for each of the major topics. This is an iterative process – the bots are re-trained periodically (according to a configurable schedule) to include in the trainsets the newly handled documents. At a given iteration:

- topics are selected that have assembled enough (a configurable amount) documents; bots are trained and evaluated for each of selected topics; balanced trainset is used for training of each of the bots;
- trained models are saved for bots with performance (precision) above the configured threshold;
- each of the incoming documents is processed with all of the saved bots; a topic is predicted (or not predicted) according to the process described in Chapter 5;
- if a prediction is made the related rule is triggered which adds the indexing and routing metadata to the document, document is routed to the assignee;
- if a prediction is not made, document is routed to a clerk responsible for manual indexing.

4. Document Annotation

We assume that the document annotation process can be improved by organizing the stream of text samples in groups of similar texts. This would help both to create the appropriate topic set and to reduce the manpower necessary for sample annotation.

We research and compare here two technologies in this respect:

- clustering methods;
- Elasticsearch MLT query.

The result in either case will be a method that improves the sequence of samples. In the case of clustering the next sample will be fed to the expert from the same cluster (if there are any). In the case of MLT – the next sample will be the most similar found by the search. We analyse thus a number of clustering workflows as well as MLT configurations. The solution is considered to be better if the overall probability of feeding the sample of the same topic is higher.

Various clustering pipelines have been explored, using text embedding methods tfidf, doc2vec and lvbert, feature reduction methods umap, pca and truncatedSVD, and clustering methods kmeans, birch and hdbscan.

A number of methods exist for comparing of the clustering results. This includes a-priori metrics like Silhouette, Calinski-Harabasz and Davies-Bouldin (Petrovic 2006) that can be used to evaluate overall structural features of the clustering. Another group of metrics (like rand index) allow to assess how well the clusters match the grouping of documents by topics. Neither of those metrics allows to compare the performance of clustering methods against the MLT. We introduce therefore the Similarity Score metric.

Similarity score is the average probability that the next document in the stream has the same topic as the current. We calculate the Similarity Score empirically by running the process several times and averaging the results. In case of clustering we select randomly the cluster and then select randomly samples inside the cluster. In case of MLT we select randomly the first document and then use MLT search to find the most similar document out of all documents not handled yet. If MLT search returns nothing the next document is selected randomly.

While doing this we calculate for each topic the frequency

$$F_t = \frac{S_t^{poz}}{S_t} \tag{1}$$

Here F_t is the Similarity Score; S_t^{poz} is a number of times when the document of topic t is followed by the document of the same topic, S_t is a total number of documents of topic t.

Figure 1 shows similarity scores for the 4 largest topics (presented on x-axis with the topic name and count of topic's documents). The similarity score for MLT and clustering is shown as well as share of the topic in the total document set.

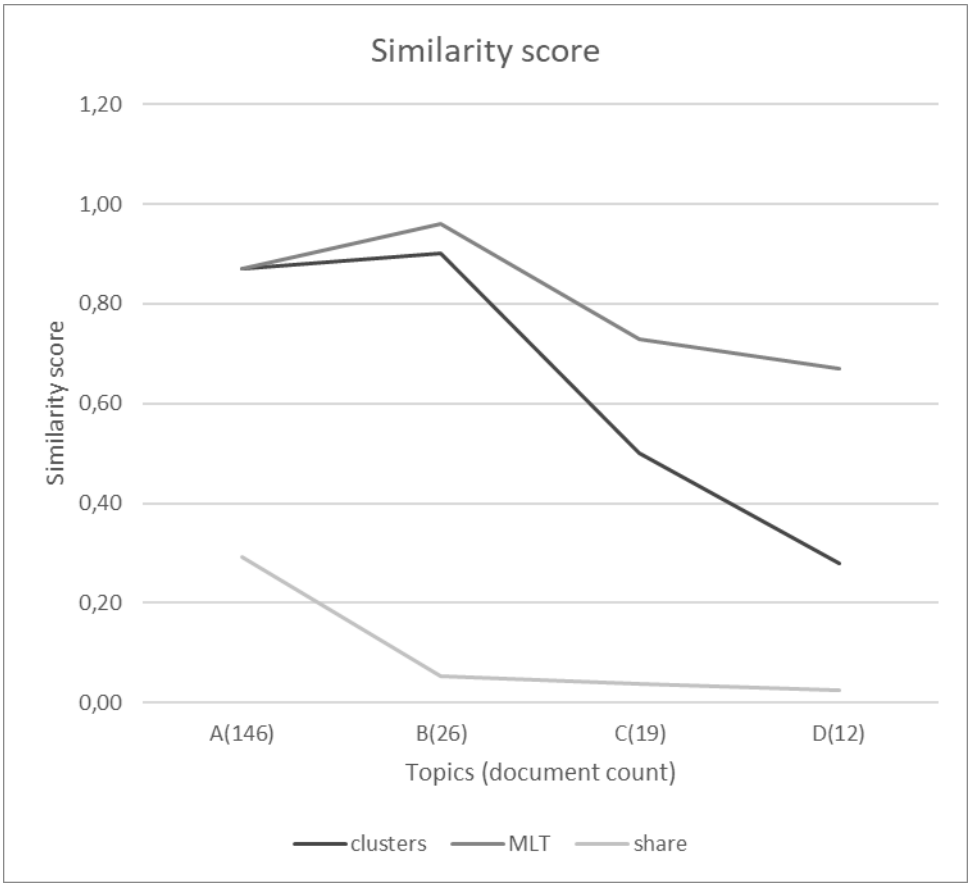

Figure 1. Similarity Score for clustering and MLT search.

We can see here that both MLT and clustering improve the probability that the next document comes from the same topic (as compared to probability for random selecting that equals to the topic share). MLT provides better results though as the explored clustering methods. The possible reason might be that clustering has no knowledge about what terms are important to distinguish topics of interest. MLT search determines a closest match for the current text on the fly and this gives a better chance to get the document of the same topic in contrary to clustering documents in advance and then selecting the next document from the same cluster.

Another observation here is that the similarity score strongly depends on the topic. The possible reason is that some topics are better separated. E.g. topic B is better separated from other topics than topic D. Further research is necessary here to understand if this information can be used to improve topics (i.e. merge, split or redefine some topics).

5. Topic Prediction

Figure 2 shows the document distribution by topics for one of the data sets (of 987 documents) explored in our research. We have observed a similar unbalanced distribution in other data sets of incoming documents as well (a total of 5 other data sets explored).

The four largest topics account for 58% of all documents and the 20 largest topics – for 82%. The rest belongs to documents currently not attributed to any topic (i.e. these are documents having no or very few similar documents to them).

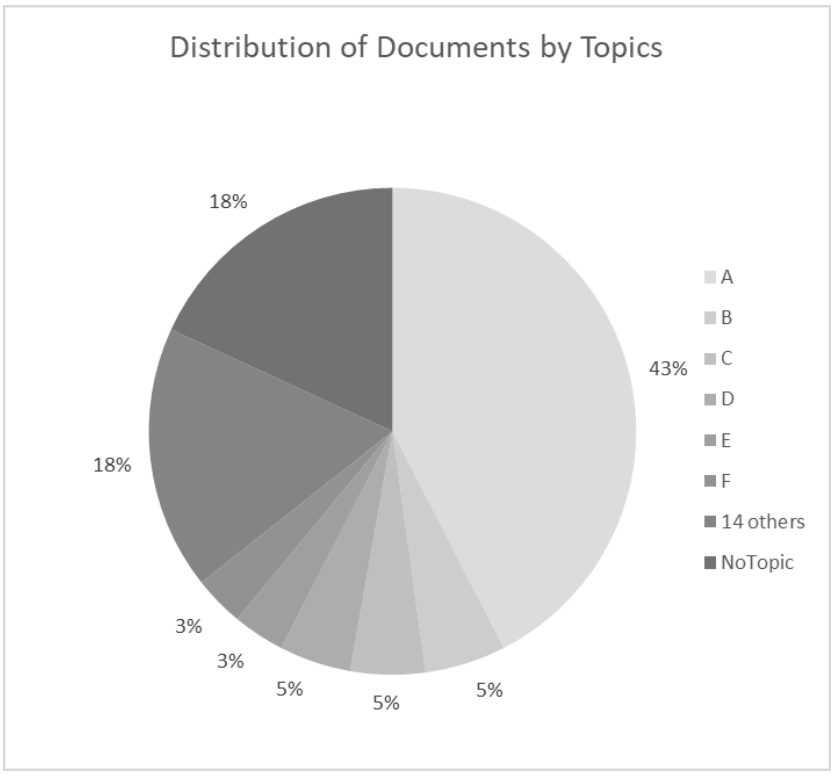

Figure 2. Distribution of documents by topics.

The unbalanced nature of data sets is a challenge for ML models. Other important challenges to address when handling the Enterprise incoming document are:

- the topic distribution and topic document content may gradually change over time;
- the penalty of false positives (i.e. topic assigned for the document that does not belong to the topic) should be bigger than that of false negatives (as false positive means incorrect indexing/routing that must be fixed

afterwards while false negative means the document is not indexed/routed automatically but handled by a clerk manually).

ML models need training data that should contain a balanced amount of positive and negative samples. The unbalanced nature of the dataset means that the major parts of topics would have small number of related documents – thus it would be not possible to train performant classification models for them anyway. We propose a dynamic topic model to address this. Main concepts of the model:

- topics are created on the fly as new documents appear; documents with topics assigned are added to the document pool;
- if a document in question does not belong to any frequent topic it is assigned a special *no_topic* label that signals that his document is not used for classification bot training; these *no_topic* documents may be revisited later at any time to assign topic if similar documents appear; *no_topic* documents are added to the pool as well;
- the document pool is examined periodically to create trainsets; the model attempts to create a balanced trainset for each of the topics; the trainset consists of equal number of positive samples (documents of the particular topic) and negative samples (documents of other topics);
- binary classification bots are created for each of the topics having enough (configurable number *min_samples*) samples;
- in case there are a lot of training samples available only the newest *item_count* samples selected (to adapt to possible changes in the topic's document content);
- binary classification bots are evaluated and bots with precision higher than a configurable threshold *min_precision* are selected for later use;
- all created bots are used for topic prediction of an incoming document;
- if some of the bots predict a topic with a score at least *min_proba* and the prediction score of the second-best prediction is at least *margin* less than the best prediction score, the topic of the best prediction is selected as the final prediction; otherwise no prediction is made.

This is a dynamic model because the topic set may be changed as the distribution of documents by topics changes. It should be noted as well that the overall model performance depends on how well the topics are separated one from another. In the best-case scenario precision (and recall) for all topic classification would be high and thus most of the incoming documents would be indexed and routed correctly. Even in this case though some false positives may happen. This should not be a show stopper because invalid indexing and routing may happen with manual handling as well. The precision may be improved in some degree controlling the *min_proba* parameter. Higher values of *min_proba* mean higher precision and lower recall. This means that there will be less false positives at the cost of less topics handled automatically.

We use OvR (One-vs-Rest) approach to get the final prediction from the bot predictions. This means we have to train a bot for each of the topics. An alternative approach would be to use the ensemble classification technique ECOC (Error-Correcting Output Code) (Kumar, 2021) that represents the multi-class label as a binary code. Depending of the dimensionality of the code ECOC may be used to train more bots than there are topics. This may improve the robustness of the prediction model.

Figure 3 shows how *min_proba* influences the prediction and recall. The best value for the data in question is about 90%.

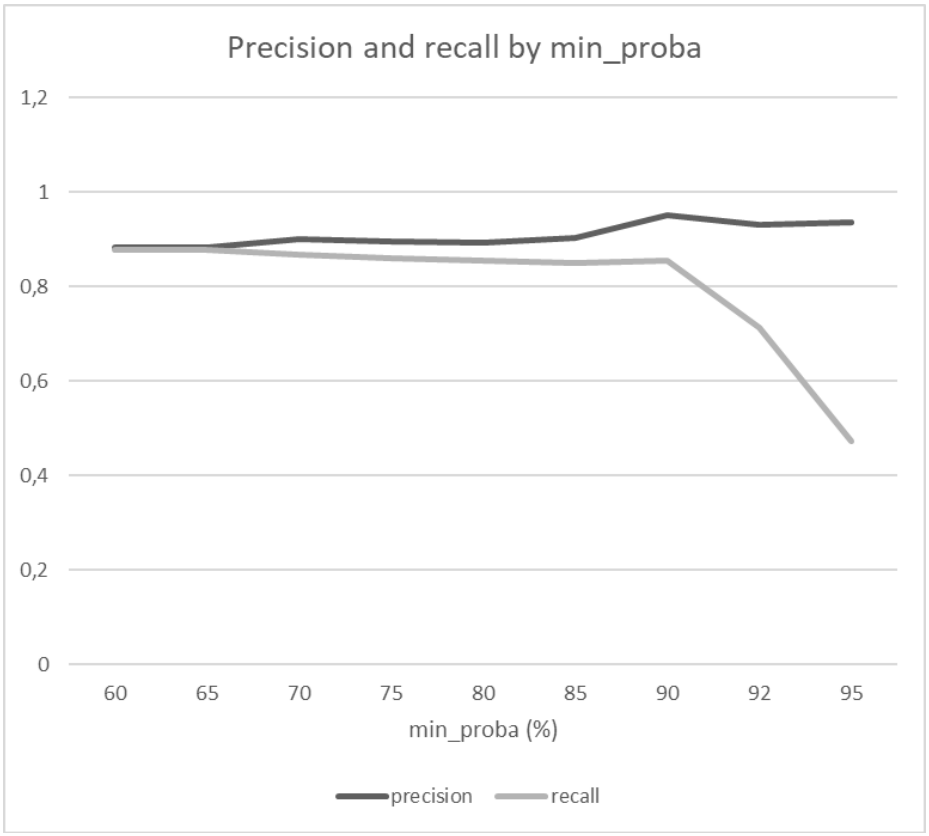

Figure 3. Precision and recall by *min_proba*.

In case of badly defined topic and/or noisy topic samples precision of the respective classification bot might be lower than *min_proba* and the handling of the topic would be switched to manual. Experts must be provided with tools to analyse samples and to remove bad samples or redefine topics (e.g. merge similar topics). MLT query or clustering could be used here to find the clusters of similar samples inside the topic. The analysis of these clusters might give an idea how to split the initial topic to get better classification performance. Another option would be to analyse topic samples with low prediction probability and to remove them from the topic.

6. Training the classification bots

The training of the classification bots is implemented as a pipeline of the following steps (**Table 1**). To use a bot for prediction the saved embedder, reducer and classifier models are loaded, the data is stripped and cleaned, then embedder, reducer and classifier executed to get the topic prediction score.

Table 1. The pipeline for bot training.

Step	Description
Select data	Balanced set of positive (documents of the given topic) and negative (documents of other topics) samples are selected from the pool.
Strip data	The document is split into blocks, leading and trailing blocks as well as short blocks are stripped.
Clean data	E-mail addresses, numbers, stop words removed
Split data into train and test sets	Train sets are used for the training of embedder, reducer and classifier. Test sets are used for validation. Kfold or stratkfold method used to create several splits for train and test data (kfold/stratkfold splits data in k folds, uses k-1 folds for training and one-fold for testing; this produces k different splits into train/test data).
Train feature embedder	Embedding model is trained on the train set. Tfidf, doc2vec and Latvian language trained bert model lvbert used.
Train feature reducer	Feature reduction methods umap and birch used to reduce embedding dimensionality.
Train classifier	Multi-layer perceptron model (mlp) used for classification. Several network layer configurations compared.
Validate models	Trained embedder, reducer and classifier executed on test data set and precision calculated
Save bot models	If the average precision for all tests is above *min_precision* threshold, train embedder, reducer and classifier on full data and save the trained models.

Embedding, feature reduction and classification methods can be switched in the model and hyperparameters tuned via editing the appropriate parameters in the configuration file. Adding new methods is relatively easy albeit this demands a little bit of *python* coding.

We used the model to measure performance for several combinations of methods and hyperparameters.

Figure 4 gives the comparison of topic prediction accuracy for embedding methods tfidf, doc2vec and bert (lvbert model) for largest 6 topics. Identic data cleaning, feature reduction and classification methods used in all three cases. As we can see tfidf shows the best results for our data and a task in hand (topic prediction). It should be noted that we used lvbert model as is – pretrained on general text corpus.

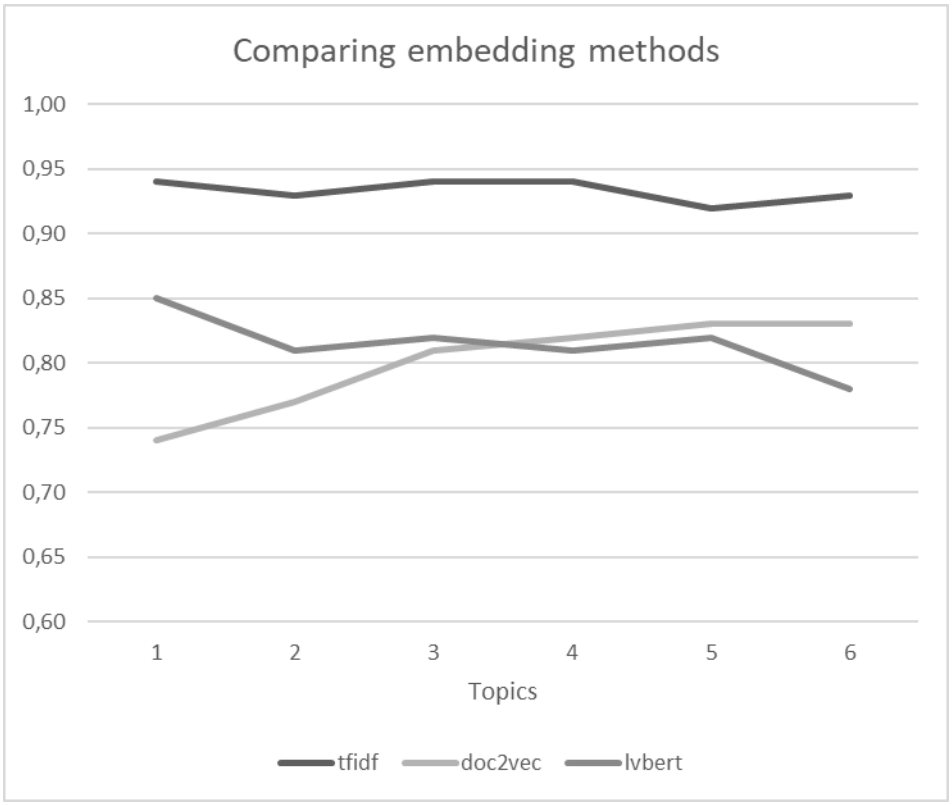

Figure 4. Comparing embedding methods.

7. Conclusions and future work

A model for automated handling of incoming enterprise documents is introduced in this article. The model comprises two processes – setup and main process. The goal of the setup process is a creation of annotated trainsets for classification bots of the main process. Applying clustering methods and Elasticsearch MLT query to support the annotation process is explored. We assume that grouping stream of documents presented to experts in clusters of similar documents should improve both the process of topic selection and that of document annotation. We assume as well that the configuration of the solution (both clustering and MLT) is better if higher the likelihood the next document in the stream is of the same topic as the current one.

As results of the experiments show both clustering MLT may be used to improve the annotation process of our model. It appears as well that MLT performs better than clustering here. This means that Elasticsearch MLT is a viable option for the support of trainset annotation for text classification of enterprise documents.

A parametric model for training of binary classification bots and for assembling the final prediction is proposed and analysed. The module has a number of configurable parameters:

- bots are involved in prediction if enough samples are available for training and if precision of the bot is better than a configurable threshold;
- prediction probability is used to decide if the sample belongs to the topic; by tuning the probability threshold above which the sample is considered to belong to the topic it is possible to increase the precision (at a cost of lowering the recall); the experiments show that it may be possible to increase precision by 5% at a cost of 3% loss in recall.

Currently the proof of concept is developed for the model and the first evaluations are performed on a couple of document sets. More analysis and evaluation are expected when the model will be implemented in a document management product and deployed to customers.

Further research is necessary to provide experts with tools for analysis of the created topics. It should be explored if it is possible to use data of the topic's documents (e.g. clustering data) to suggest topic improvements (e.g. merging, splitting or reorganizing topics).

List of abbreviations

BERT	Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing.
BoW	The Bag of Words model represents text as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.
doc2vec	An extension of word2vec to construct embeddings from entire documents (rather than the individual words).
ELMo	Embeddings from Language Model is a word embedding method for representing a sequence of words as a corresponding sequence of vectors.
fasttext	The library for learning of word embeddings and text classification created by Facebook's AI Research laboratory.
GloVe	The Global Vectors is a model for distributed word representation.
GPT	Generative Pre-Training – an unsupervised transformer language model.
MLT	More Like This query. The tfidf based means for searching similar documents provided by Elasticsearch software.
PCA	Principal Components Analysis – a method for feature reduction.
tfidf	Term Frequency – Inverse Document Frequency. The method for text embedding
ML	Machine Learning.
RCNN	Region-based Convolutional Neural Network.
UMAP	Uniform Manifold Approximation and Projection is a feature reduction method.
word2vec	The word2vec algorithm uses a neural network model to learn word

associations from a large corpus of text.

XLNet XLNet is an auto-regressive language model which outputs the joint probability of a sequence of tokens based on the transformer architecture with recurrence

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, research No. 1.17).

References

Avinash, M., Sivasankar, E. (2019). A Study of Feature Extraction techniques for Sentiment Analysis, Advances in Intelligent Systems and Computing, pp. 475-486.

Assent, I., Seidl, T. (2009). Evaluating Clustering in Subspace Projections of High Dimensional Data, available at http://dme.rwth-aachen.de/OpenSubspace/evaluation.

Baker, D.L. (1998). Distributional Clustering of Words for Text Classification, available at www.cs.cmu.edu/~ldbappwww.cs.cmu.edu/~mccallum.

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

Brendel, C. (2020). Cluster-Then-Predict for Classification Tasks | by Cole Brendel | Towards Data Science, available at https://towardsdatascience.com/cluster-then-predict-for-classification-tasks-142fdfdc87d6.

Cygan, N. (2021). Sentence-BERT for Interpretable Topic Modelling in Web Browsing Data Stanford CS224N Custom Project, available at https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/reports/final_reports/report017.pdf

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, pp. 41-52.

Fernández, J., Antón-Vargas, J.A., Villuendas-Rey, Y., Cabrera-Venegas, J.F., Chávez, Y., Argüelles-Cruz, A.J. (2016). Clustering Techniques for Document Classification. Research in Computing Science 118(1): 115–25.

Fu, M., Qu, H., Huang, L., Lu, L. (2018). Bag of meta-words: A novel method to represent document for the sentiment classification, Expert Systems with Applications 113, pp.33-43.

Hansen, P. C. (1987). The TruncatedSVD as a Method for Regularization. BIT Numerical Mathematics 1987 27:4 27(4): 534–53.

Jacovi, A., Sar Shalom, O., Goldberg, Y. (2018). Understanding Convolutional Neural Networks for Text Classification, Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 56-65.

Karpov, I., Goroslavskiy, A. (2012). Application of BIRCH to Text Clustering, available at https://www.researchgate.net/publication/286672732_Application_of_BIRCH_to_text_clustering.

Klinger, J. (2019). Big, Fast Human-in-the-Loop NLP with Elasticsearch, Towards Data Science, available at https://towardsdatascience.com/big-fast-nlp-with-elasticsearch-72ffd7ef8f2e.

Kumar, S. (2021). Stop Using One-vs-One or One-vs-Rest for Multi-Class Classification Tasks, Towards Data Science, available at https://towardsdatascience.com/stop-using-one-vs-one-or-one-vs-rest-for-multi-class-classification-tasks-31b3fd92cb5e.

Kyriakopoulou, A. (2008). Text Classification Aided by Clustering: A Literature Review. In Tools in Artificial Intelligence, InTech., available at www.intechopen.com.

Kyriakopoulou, A., Kalamboukis, T. (2007). Using Clustering to Enhance Text Classification, available at http://www.cs.cmu.edu/afs/cs.cmu.edu/.

Lauren, P., Guangzhi, Z.F., Lendasse, A. (2018). Discriminant Document Embeddings with an Extreme Learning Machine for Classifying Clinical Narratives, Neurocomputing 277, pp. 129-138.

Lin, R., Fu, C., Mao, C., Wei, J., Li, J. (2019). Academic news text classification model based on attention mechanism and RCNN, Communications in Computer and Information Science, pp. 507-516.

Liu, Q., Kusner, M.J., Blunsom, P. (2020). A Survey on Contextual Embeddings. ArXivID: 2003.07278v2, 13 Ap3. 2020.

Parsons, L., Haque, E., Liu, H. (2004). Subspace Clustering for High Dimensional Data: A Review. Sigkdd Explorations 6(1): 90–105.

Passalis, N., Tefas, A. (2018). Learning bag-of-embedded-words representations for textual information retrieval, Pattern Recognition 81, pp. 254-267.

Pennington, J., Socher, R., Manning, C. (2014). Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Stroudsburg, PA, USA: Association for Computational Linguistics, 1532–43.

Petrovic, S. (2006). A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters, available at https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.4114&rep=rep1&type=pdf.

Shawon, A., Zuhori, S.T., Mahmud, F., Rahman J. (2018), 21st International Conference of Computer and Information Technology (ICCIT), pp. 1-6.

Steinbach, M., Ertöz, L, Kumar, V. (2004). The Challenges of Clustering High Dimensional Data, New Directions in Statistical Physics, pp. 273-309.

Ular, M., Robnik-Šikonja, M. (2020). High Quality ELMo Embeddings for Seven Less-Resourced Languages. In LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings, pp. 4731–4738.

Vola, S. (2017). How to Use ElasticSearch for Natural Language Processing and Text Mining — Part 2 – Dataconomy, available in https://dataconomy.com/2017/05/use-elasticsearch-nlp-text-mining-part-2/.

Wei, F., Qin, H., Ye, S., Zhao, H. (2018). Empirical Study of Deep Learning for Text Classification in Legal Document Review, 2018 IEEE International Conference on Big Data (Big Data), pp. 3317-3320.

Yellai, M. (2016). GitHub - Pandastrike/Bayzee: Text Classification Using Naive Bayes and Elasticsearch, available at https://github.com/pandastrike/bayzee.

Received April 21, 2022, revised June 28, 2022, accepted August 18, 2022