A News Recommender System for Media Monitoring

Francesco Barile
Free University of Bozen-Bolzano
francesco.barile@unibz.it

Francesco Ricci
Free University of Bozen-Bolzano
fricci@unibz.it

Marko Tkalcic
Free University of Bozen-Bolzano
marko.tkalcic@unibz.it

Bernardo Magnini
Fondazione Bruno Kessler
magnini@fbk.eu

Roberto Zanoli
Fondazione Bruno Kessler
zanoli@fbk.eu

Alberto Lavelli
Fondazione Bruno Kessler
lavelli@fbk.eu

Manuela Speranza
Fondazione Bruno Kessler
manspera@fbk.eu

ABSTRACT

Media monitoring services allow their customers, mostly companies, to receive, on a daily basis, a list of documents from mass media that discuss topics relevant to the company. However, media monitoring services often generate these lists by using keyword-filtering techniques, which introduce many false positives. Hence, before the end users, i.e., the employees of the company, may consult these lists and find relevant documents, a human editor must inspect the keyword-filtered documents and remove the false positives. This is a time consuming job. In this paper we present a recommender system that aims at reducing the number of documents that the editor needs to inspect every day. The proposed solution classifies documents (represented with TF-IDF and embeddings features) using techniques trained on data containing the editors' past actions (i.e. the removals of false positives). The proposed technique is shown to be able to correctly predict the true positives, thus reducing the number of documents that the editor needs to inspect every day.

KEYWORDS

Media Monitoring, News Recommender Systems

ACM Reference Format:
Francesco Barile, Francesco Ricci, Marko Tkalcic, Bernardo Magnini, Roberto Zanoli, Alberto Lavelli, and Manuela Speranza. 2019. A News Recommender System for Media Monitoring. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI '19), October 14–17, 2019, Thessaloniki, Greece.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3350546.3352510

1 INTRODUCTION

Media monitoring (MM) services, also referred to as clipping services, provide their customers with a selection of daily media content that is of interest to them. Such content, here referred to as documents, can be obtained from any kind of media, such as newspapers and other print media, video and audio services, and web and social media [16]. These services are used by customer companies to track the mentions of their brand name, products or even advertisements, as well as those of their business partners and competitors. MM usually allows each customer to define a set of specific keywords (e.g. the name of their company or some topics of interest), which are then used to keyword-filter the documents of interest. Such daily inspection of documents allows the customer company to assess its reputation and visibility along with helping the company in strategic planning. Examples of MM services are *Meltwater* [2], *Cision Media Monitoring* [1], *Mention Reviews* [3], and *News Exposure* [4].

However, the required process of inspecting the keyword-filtered documents and removing false positives is time-consuming. The work-flow in the customer company typically involves a human editor who inspects, on a daily basis, each document provided by the MM service and decides whether it is really relevant or not. Keyword-filtering, used by the MM service, can yield false positives, especially if the name of the company is polysemous (e.g. Apple, Jaguar). In a successive step, the documents selected by the editor are typically forwarded to other employees in the customer company (end users) in the form of internal press releases.

The aim of the system presented here is to minimize the daily work done by the human editor. To this end, we have developed a recommender system that operates after the keyword-filtering step and before the final check of the human editor occurs. Our recommender system uses machine learning classification techniques to label the keyword-filtered documents as either relevant or non-relevant. After that, the editor receives a newly filtered set of documents with a high probability of being relevant, which do not need manual inspection, and a set of documents that the system scores with a lower probability of being relevant, which on the other hand do require manual inspection. Our experimental results show that the proposed system has (i) high precision, which means that the documents classified as relevant are truly relevant, and (ii) high recall, meaning that the classifier is able to capture most of the relevant documents.

The research presented here was applied to the *Infojuice* product, offered by the Italian company *Euregio*, which provides MM services together with a tool for the qualitative and quantitative analysis of

WI '19, October 14–17, 2019, Thessaloniki, Greece

ACM ISBN 978-1-4503-6934-3/19/10…$15.00
https://doi.org/10.1145/3350546.3352510

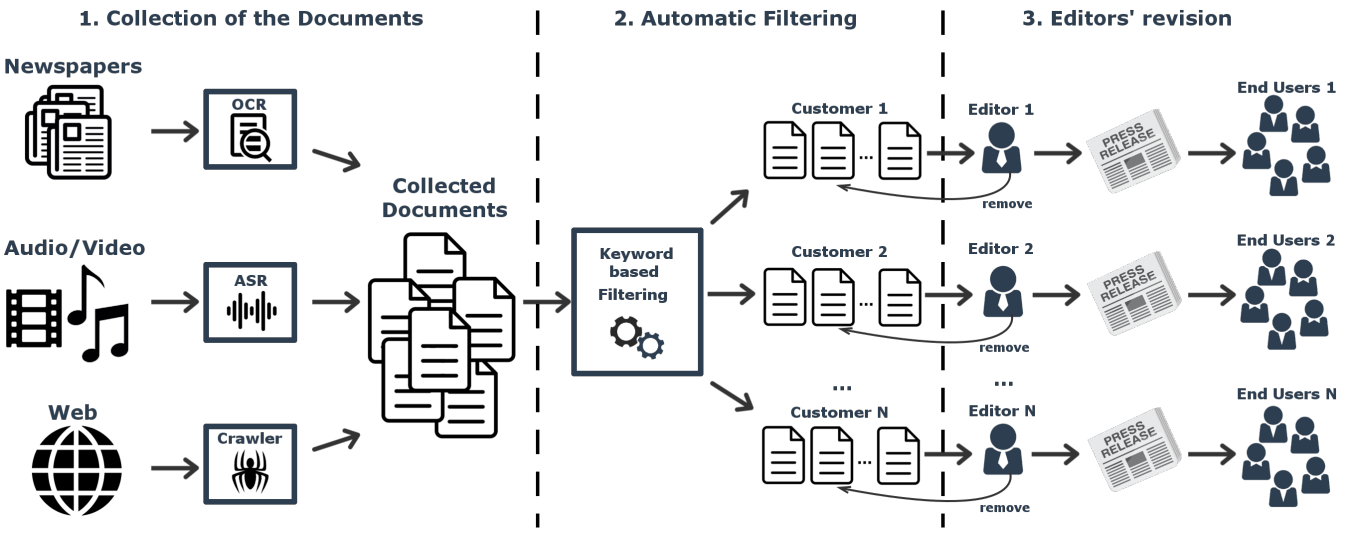

Figure 1: The publication workflow of the InfoJuice system.

their customers' level of representativity on the media. The product is able to manage documents in Italian, German and English, from a range of different sources. The proposed solution uses machine learning techniques to distinguish relevant from non-relevant documents for each customer. It is trained on data describing the past actions performed by the editors, i.e., the discarding of non-relevant documents. We have explored several machine learning classification strategies, using different combinations of document features extracted from the document title, body and source (e.g. the newspaper or the online source etc.). For extracting features from the text we have used term frequency–inverse document frequency (*Tfidf*) as well as *Fasttext* embeddings.

The system has been evaluated offline using data extracted over a time window of six months, taking into consideration the activity of five test customers of the Infojuice system. The results indicate that the number of documents that the editor needs to inspect manually (i.e. the ones that our system has not marked as relevant) is substantially lower than the number of documents returned by the keyword-filtering step, hence reducing the time that the editor needs to allocate for this task on a daily basis.

In the rest of the paper we present the application problem and, after a short summary of the relevant state of the art, we present our technical solution. Finally, we show the results of an offline experiment, which indicate the benefit of the technique.

2 APPLICATION PROBLEM

The Infojuice system provides customers with a service for the generation of personalized selections of daily media documents. The work-flow used for generating these document selections is organized in different steps, as shown in Figure 1:

(1) Collecting daily documents: in the first step documents are collected from different sources. The newspaper articles are scanned with OCR tools to automatically recognize the title, subtitle, header and content. After that, the scanned documents are inspected by human operators that fix potential mistakes. For the audio and video contents, an automatic speech recognition tool transcribes them as written texts. Furthermore, a web crawler collects data from several information sources on the Internet. After that operation, the documents are inserted into the system's database. The documents are characterized by a set of common attributes: (a) the *title* of the document (in the case of newspaper articles, the *header* and *subtitle* are also present); (b) the *text* of the document, i.e. the body of the article or the transcription of the audio/video stream; (c) the *date* in which the document has been created; (d) the *source* from which the document has been extracted.
(2) Automatic filtering: in the second step, a keyword-based filtering module creates a personalized selection of documents for each customer.
(3) Editor's revision: the final step consists of the manual inspection and selection done by an *editor user*, specific to each customer. Each editor checks the documents that have been selected automatically through keywords-filtering and performs two main actions: (a) **removes** the documents that are not relevant to the customer, i.e., that have been erroneously associated with the customer by the keyword-based filtering module; (b) selects a subset of the documents deemed as relevant, thus **composing a press release**, and sends it to a group of end users, typically the customer's employees.

During the inspection process, the editor usually reads the documents in order to verify their relevance. In fact, the whole inspection process can be very difficult and lengthy to perform. Furthermore, the relevance of the documents not only depends on the end users' interest in the documents but also on the company's policies. Thus, providing help during this activity could be very useful to speed up the work of the editor and to increase the quality of the final

result. To address this problem, we have designed and developed a software system that supports the editors by estimating a *Relevance Score* for the documents associated with each customer. The documents scoring higher than a threshold are deemed relevant and do not require the editor's attention, hence saving time. Our system computes that relevance score with machine learning algorithms that leverage the past actions performed by the editor and uses features extracted from the documents.

3 STATE OF THE ART

Media monitoring services automatically monitor different sources of information, such as newspapers or the Internet [16]. This monitoring activity is usually used by companies to perform an analysis of particular topics of interest, in order to determine the impact on the market and the value of their brands, but also to monitor competitors and to protect their reputation and plan the company's policies [17, 19]. Several companies offer these kinds of tools, but there are very few papers in the literature presenting the technological solutions adopted. Meltwater [2] is a monitoring service tracking keywords and phrases in more then 300,000 online sources, offering a personalized dashboard that allows the customers to perform various analyses on the retrieved documents. Cision [1] and News Exposure [4] provide monitoring on different kinds of media: online monitoring on the Internet, broadcast monitoring on TV and radio broadcasts print monitoring on newspapers and social monitoring that analyzes social networks, also providing analytic tools. Mention [3] is a social media monitor, hence it focuses on web and social media content, providing tools to monitor in real-time customers' mentions and also allowing for the tracking of competitors. While these tools use data mining techniques to analyze documents and provide customers with several reports and statistics, the selection of the documents related to the customers is mostly based on keywords typically describing the customers' name, products and competitors.

The problem that we are addressing (i.e. selecting the relevant documents for customers from a huge set of documents collected and digitized each day from different sources) can also be tackled using news recommendation techniques. The goal of News Recommender Systems is to provide suggestions about the news to read [11]. Generally, such recommendations are provided to single users, and the system learns their preferences from their past activities, which are in turn used to generate the suggestions. In our case, however, the target customer is a company and the relevance of the documents must be inferred from the feedback of the editor that reviews the documents associated with a particular customer. Another difference is that, while in the general case the news recommendation problem is characterized by a rapid shift in individual interests, often influenced by contextual factors [11], in our case this phenomenon is fairly limited, since the customers' policies do not change at the same rate. As stated above, News Recommender Systems typically use classical approaches like Collaborative Filtering (CF), Content Based (CB), or hybrid strategies [11]. While in most cases a CB approach is preferred, since it allows one to easily address the item cold-start problem, CF strategies are also used in some cases. In [7], a CF approach is presented that does not consider the content of the news and focuses on the users clustering in order to have a more scalable and content-independent solution, while [13] relies on the analysis of the click logs to infer changes in users' news interests. In [12] a CB approach is presented, using a custom article representation in order to speed up the similarity estimation and to process new articles as fast as possible. In [10] a taxonomy of hierarchical semantic abstractions is used, trying to capture different semantic facets of the news using several linguistic tools. Semantic web technologies are used in [6], where news contents and user profiles are both represented using the concepts appearing in a domain ontology.

In our scenario the CF approach is not feasible, given the small number of customers and feedback that we can use. Therefore, we built a collection of customer specific models by means of machine learning classifiers that are trained using the feedback collected by the editor users for each customer.

4 THE PROPOSED SOLUTION

The proposed solution is a *Recommender System (RS)* that computes recommendations by using one classifier for each customer. The RS is used by the *Infojuice System (IJ)* to provide support to the editors during their daily activity.

Several factors have been taken into consideration in the design of the RS. In particular, since the number of customers is very small, collaborative filtering is not applicable. Furthermore, the available feedback of the editors is restricted to the action of removing documents from the list produced by the keyword-based filtering queries and we must rely only on negative feedback. Hence, we decided to implement a solution optimized for each customer by using a set of automatic classification techniques that leverage data generated by the editors' actions in order to distinguish relevant from non-relevant documents.

We denote with C the set of the customers of the IJ system, with D the set of documents in the system and with $D_{i,k} \subset D$ the documents that the IJ system associates with the customer $c_i \in C$ in the month m_k by means of keyword-based queries. We then define the set of relevant and not relevant documents for the customer c_i in the month m_k, indicated as $Rel_{i,k}$ and $NotRel_{i,k}$ respectively, as follows:

- $NotRel_{i,k}$ is the set of the documents in D_i that have been removed by one of the editors of the customer c_i in the month m_k (with $NR_{i,k} = |NotRel_{i,k}|$);
- $Rel_{i,k} = D_{i,k} \setminus NotRel_{i,k}$ is the set of the documents in $D_{i,k}$ that have not been removed by the editors (with $R_{i,k} = |Rel_{i,k}|$).

4.1 Classification Strategies

We decided to classify the documents with two standard approaches: Multinomial Naïve Bayes (MNB) and Support Vector Machines (SVM). MNB is a probabilistic classifier based on *Bayes' Theorem* and the *Naive Conditional Independence* assumption that learns the probability of a document d_i belonging to a $class_k$ [14]. It uses a smoothed maximum likelihood estimate to derive the model parameters, which are the prior probability of a document being in $class_k$ and the conditional probability of a feature x_i to appear in a

Table 1: Number of relevant ($R_{i,k}$) and not relevant ($NR_{i,k}$) documents for each month m_k of the dataset considering the customers used in the experiment.

Customer ID	m_1		m_2		m_3		m_4		m_5		m_6	
	$R_{i,1}$	$NR_{i,1}$	$R_{i,2}$	$NR_{i,2}$	$R_{i,3}$	$NR_{i,3}$	$R_{i,4}$	$NR_{i,4}$	$R_{i,5}$	$NR_{i,5}$	$R_{i,6}$	$NR_{i,6}$
62	426	96	588	152	450	145	589	123	588	128	663	163
332	239	995	307	981	282	855	316	1088	235	1216	403	850
502	145	757	207	54	208	47	230	63	245	78	353	117
191	769	124	819	122	689	129	1000	124	1166	120	1075	121
409	2519	85	2600	69	1593	23	3167	60	3056	99	2750	112

document of $class_k$ [14]. It uses a smoothing parameter α to estimate these probabilities and this is the unique hyper-parameter of the model.

In the SVM classifier the documents are modeled as points in the selected feature space. The training data are then mapped to a higher dimensional feature space and a hyper-plane (or a set of hyper-planes) that separates the documents belonging to different classes is searched [9]. The mapping is performed through a kernel function, in the simplest case it is a linear function, but also polynomial, exponential and sigmoid functions can be used. The hyper-parameters of the model depend on the kernel. We decided to use a linear kernel (LIN SVM), characterized by a hyper-parameter c, and an exponential one, called radial basis function kernel (RBF SVM), characterized by two hyper-parameters, c and γ [8]. We implemented all the classifiers using the python package *sklearn* [15].

4.2 Feature Models

Regarding the features extracted from the documents, we use information derived from the *Title*, the *Text* and the *Source* of the documents. The document's *Source* (for instance a magazine) is represented with a binary vector: one binary feature for each source; 12.025 sources are considered. In the following, we will refer to this feature as SourceBinary. For the textual components of the documents, i.e., the *Title* and the *Text*, we use two different representations:

- Bag of Words Model: we use *tf-idf* with two different models trained for the Title (TitleTfIdf) and for the Text of the documents (TextTfIdf);
- Word Embeddings Model: in particular, we use the *Fasttext* model on the Text of the documents [5] (TextFasttext);

All the features can be combined and used together. Hence, a classifier's configuration consists of (i) the classification strategy, which can be either the Multinomial Naïve Bayes (MNB), the Support Vector Machines with linear kernel (LIN SVM) or SVM with the rbf kernel (RBF SVM), and (ii) the features combination used.

4.3 Time Constraints

One important aspect considered in the design of our solution is related to the time constraints imposed by the application scenario. Our recommender system is used to help the editors in their daily activity, during the working days of each week (from Monday to Friday). Hence, the RS is trained once a week, during the weekends, in order to guarantee the system availability in the editor's working hours. Two main processes are executed:

- Training the RS; this is performed during the weekend and includes: (a) training of all the customer's classifiers (we refer to this step as CUST.CLS.), and the related features extraction operations (we refer to this step as FEAT.EX.CLS.); (b) Prediction of the document relevancy for all the customer and all the documents in the system (we refer to this step as REC.EV.), since it is possible that the IJ system will require recommendations for the old documents collected in the system. Again, to perform this task, a features extraction step is necessary (we refer to this step as FEAT.EX.REC.);
- Processing new documents; during the working days, the IJ system collects new documents that must be evaluated by the editors. These documents are sent to the RS that performs first the pre-processing step (we refer to this step as PRE.PROC.), then the FEAT.EX.REC step, and finally it calculates the predicted scores for each customer and for each new document (i.e., the REC.EV. step).

Hence, during the daily activity, the RS only processes the new documents produced during the day, and computes daily recommendations. During the weekend the system trains all the customer specific classifiers and computes the recommendations for all the documents in the system. To ensure the feasibility of our approach, we will evaluate the performances of each of the five steps that are mentioned above. The analysis is aimed at understanding the impact, on the system performance, of the alternative feature sets and the number of documents processed.

5 SYSTEM EVALUATION

We evaluated the proposed recommender system in a set of offline experiments using a dataset containing documents and editors' actions collected in a time window of six months. As the goal was to identify a set of documents that are likely to be relevant, we measured precision (because we wanted to minimize the number of false positives in the set of recommended documents), recall (because we did not want to leave relevant documents in the set of documents that the editor needs to inspect), and F-measure (in order to combine both precision and recall).

5.1 Dataset

Our data set contains documents and editors' actions timestamped from the 1st of December 2017 to the 31st of May 2018. It includes 365,430 Italian documents. We considered five test customers. For

them the editors performed removing actions in all six of the months considered. Table 1 shows the number of relevant ($R_{i,k}$) and not relevant ($NR_{i,k}$) documents for each customer c_i and for each month m_k in the dataset, following the definition introduced in Section 4. It is clear that the customers' data differs substantially.

5.2 Evaluation Procedure

We evaluated several different configurations specified by the classification strategy (MNB, LIN SVM or RBF SVM, as defined in Section 4.1), and by the used feature sets combination. The evaluation of a configuration requires two steps: (1) the validation/selection of the hyper-parameters for each classification strategy used, which determines the best hyper-parameter combination, and (2) the evaluation of the configurations using the best hyper-parameter combination for the classifier.

5.2.1 Balancing Strategy. Since for many customers the number of relevant and not relevant documents was substantially different in each test month (see Table 1), we used two different resampling strategies for balancing these numbers. If for a customer $c_i \in C$ and for a given month m_k the number $R_{i,k}$ of relevant documents was greater than the number $NR_{i,k}$ of not relevant documents, then an over-sampling method [18] was applied. We assumed that most of the documents that the keyword based filtering queries do not associate with a target customer must be not relevant documents. Hence, when considering the customer c_i and the month m_k, we randomly selected a set $D^{add}_{i,k}$ from the set $D_k \setminus D_{i,k}$ (where D_k is the set of documents collected by the IJ system in the month m_k) such that $|D^{add}_{i,k}| = R_{i,k} - NR_{i,k}$, and we added them to the not relevant set for customer c_i in the month m_k. On the other hand, if the number of relevant documents $R_{i,k}$ was lower then the number of not relevant documents $NR_{i,k}$, then we applied an under-sampling strategy [18], drawing a subset $D^{del}_{i,k} \subset NotRel_{i,k}$ of examples from the set of not relevant documents such that $|D^{del}_{i,k}| = NR_{i,k} - R_{i,k}$ and discarding them from the not relevant documents in the classifier training phase.

5.2.2 Hyper-parameters Setting. The performance of the classifiers introduced in section 4.1 is influenced by some hyper-parameters. We found their optimal values using grid search. Our data is time stamped, so, random cross-validation is not appropriate; validation data must be posterior to training data. Instead, we made two time correct train-validation splits and averaged the obtained scores over these two splits. In the first split, the first two months were used as training and the third month as validation; then the first three months were used as training and the fourth month as validation. In both cases, the training sets were balanced with the procedure defined in Section 5.2.1. In the following, we analyze how the classification performance changes depending on the criteria adopted in the optimization of the hyper-parameters (see section 5.3). Precision and F-measure were used as target optimization criteria.

5.2.3 Evaluation of Classifier Configurations. Once we had identified the best hyper-parameter combination for a given classification strategy, we evaluated the performance of the classifier using the fifth and the sixth month as test set, referred to as test month 5 and test month 6 respectively. In the first case, we trained the classifier using the first four months as a training set, while in the second case we used the first five months as a training set. Also in these cases, the training sets were balanced according to the strategy illustrated in Section 5.2.1.

Table 2: Precision obtained for each customer in the test month 6 using the best configuration for each type of classifiers. The *Precision* is used as validation score. The last column reports the precision of the current KBS.

Customer id	MNB	LIN SVM	RBF SVM	KBS
62	0.935	1.000	1.000	0.803
332	0.539	0.586	0.947	0.322
502	0.853	0.848	1.000	0.751
191	0.954	0.975	1.000	0.899
409	0.982	0.993	1.000	0.961

5.3 Results Analysis

We start the results analysis by comparing the proposed approach with the current Keyword-Based System (KBS) (see Table 2). It is important to highlight that our RS operates on the documents prefiltered by the KBS. Hence, we want to analyze to what extent the RS refinement of the results of KBS can increase the original performances; the ideal result is to replicate the operation performed by the editor, i.e., to discard exactly the documents that the editor considered as not relevant and therefore removed. The performance of the current KBS can only be evaluated for its *Precision*, since we only know the correctness of the documents that KBS selected as relevant. Results vary depending on the algorithm and features used, on the customers, and on the score used in the validation step. In Table 2 we show a comparison between the precision obtained for each customer in test month 6 using the best features configurations for each type of classifiers and using *Precision* to optimize the hyper-parameters of the classifiers. As we can see, all the proposed algorithms can outperform (improve) the precision of the current KBS system.

A second important aspect is the change in performance when different validation scores are used in the hyper-parameters optimization phase. Table 3 shows the performance of the best configuration for each customer and for each classifier obtained using *Precision* for hyper-parameters optimization. Table 4, instead, shows the results obtained using F_1Score to perform the hyper-parameters optimization. The first conclusion of this analysis is that we can obtain a system with very high *Precision* (equal or close to 1.000) if we use it as validation score. A second evidence from these results is that there is not a single winning classifier. SVM present the best results but, in the case of optimization for F_1Score (Table 4), we can notice that the linear kernel returns the best results for three out of five customers.

Since these results show that there is no classification strategy that outperforms the others in general, we have explored the possibility to automatically select the best strategy for each customer. Hence, the idea here is to use the first four months for hyper-parameters optimization, then to use the fifth month to compare the different classification strategies and select the best one. We

Table 3: Results obtained for each customer in test month 6 using the best features configurations for each type of classifier. The *Precision* is used for the hyper-parameter optimization. In bold the best classifier with respect to the *Precision* metric is highlighted.

Customer ID	MNB			LIN SVM			RBF SVM		
	Prec.	*Recall*	F_1Score	*Prec.*	*Recall*	F_1Score	*Prec.*	*Recall*	F_1Score
62	0.935	0.757	0.837	**1.000**	0.554	0.713	**1.000**	0.554	0.713
332	0.539	0.424	0.475	0.586	0.337	0.428	**0.947**	0.045	0.085
502	0.853	0.790	0.821	0.848	0.918	0.882	**1.000**	0.074	0.137
191	0.954	0.754	0.843	0.975	0.645	0.776	**1.000**	0.018	0.035
409	0.982	0.788	0.936	0.993	0.529	0.845	**1.000**	0.010	0.049

Table 4: Results obtained for each customer in test month 6 using the best features configurations for each type of classifier. The F_1Score is used for the hyper-parameter optimization. In bold the best classifier with respect to the F_1Score metric is highlighted.

Customer ID	MNB			LIN SVM			RBF SVM		
	Prec.	*Recall*	F_1Score	*Prec.*	*Recall*	F_1Score	*Prec.*	*Recall*	F_1Score
62	0.880	0.944	0.911	0.948	0.905	0.926	0.954	0.905	**0.929**
332	0.443	0.792	0.568	0.532	0.667	**0.592**	0.525	0.660	0.585
502	0.807	0.935	0.866	0.846	0.935	**0.888**	0.849	0.941	0.866
191	0.898	0.980	0.937	0.898	0.992	0.943	0.899	1.000	**0.947**
409	0.978	0.940	0.970	0.981	0.969	**0.979**	0.980	0.948	0.973

Table 5: Comparison between the results of the best classifier selected using the results from test month 5 (*Best@Test5*), evaluated on the documents contained in test month 6, with the results of the best classifier on test month 6 (*Best@Test6*).

Customer ID	*Best@Test5*			*Best@Test6*		
	Precision	*Recall*	F_1Score	*Precision*	*Recall*	F_1Score
62	0.955	0.894	0.924	0.954	0.905	0.929
332	0.558	0.563	0.560	0.532	0.667	0.592
502	0.789	0.997	0.881	0.846	0.935	0.888
191	0.899	1.000	0.947	0.899	1.000	0.947
409	0.962	0.999	0.969	0.981	0.969	0.979

call *Best@TestK* the classifier that obtains the best results w.r.t. the metric used for the hyper-parameters optimization in test month k. Hence, we evaluate the performance of the *Best@Test5* classifier (i.e. the best classifier for the given customer on the test month 5) on the documents contained in test month 6. We compare the performance of that classifier with the performance of the best classifier in this test month 6 (*Best@Test6*). Table 5 shows this comparison, using the F_1Score for the optimization. As we can see, there are just some small differences, and the loss in performance is very small. This means that a classifier very close to the optimum for a target month can be found by selecting the best performer from the previous month.

5.4 Execution Performance

To ensure the feasibility of our approach and to explore its execution performance scalability, we conducted a set of experiments. We measured the computational time required for running the main tasks of our approach (see Section 4.3):

- PRE-PROC.: Pre-processing operations on a set of documents;
- FEAT.EX.CLS.: Features extraction operations to prepare the data used in the training of the customer classifiers;
- CUST.CLS.: Training of the customer classifiers;
- FEAT.EX.REC.: Features extraction operations to prepare the data used for the computation of the recommendations;
- REC.EV.: Computation of the recommendations for each customer on a set of documents.

Several factors influence the computational time of these tasks. We decided to observe the dependency of the computational time on two variables: (a) the number of documents to process and (b) the set of features used. Hence, in order to isolate these two factors we fixed the other variables: we chose a single classifier, a single customer, and we ran all the evaluations on the same hardware.

Among the classifiers that were used in the recommendation process, we selected the one with the worst computational complexity according to the authors of the algorithms. For MNB the complexity is linear in the time necessary to scan the data [14].

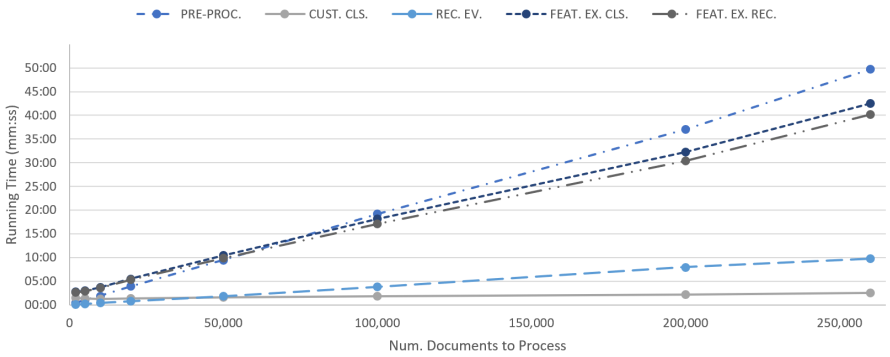

Figure 2: The running time for different number of documents to process, using all the set of features available.

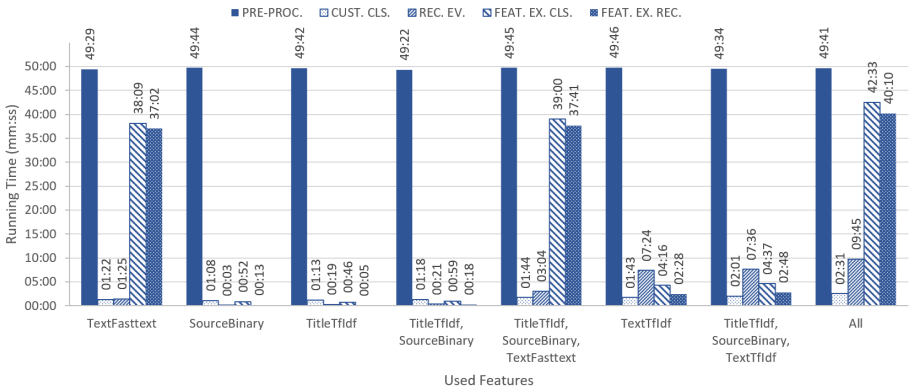

Figure 3: The running time for the different subset of features, using all the documents in the dataset.

For SVM we used the sklearn library implementation (SVC python class) for both linear and rbf kernels. The reported complexity is between $O(m^2 * n)$ and $O(m^3 * n)$ (with m the number of documents and n the number of features) [8]. Hence, we chose to report the data for SVM, which has a higher complexity than MNB. We chose the linear kernel (SVM LIN), since the difference compared to the RBF kernel about the required running time only depends on the number of hyper-parameter values that we decide to explore. Regarding the customers, we chose customer 62, since it represents an intermediate case with respect to the number of documents associated with the customer (see Table 1). All the experiments were executed on a server having eight 2.00 GHz processors Intel Xeon E7-2850, with 16GB of RAM running on the 64-bit Windows Server 2016 operating system.

The computational time for the five main tasks under evaluation depends on different aspects:

- PRE-PROC.: The computational time of this step should depend mostly on the number of documents, since it performs the same processing on all the documents independently of the features and classifiers used and the number of customers.
- FEAT.EX.CLS.: The computational time of this step should depend both on the number of documents and on the set of features used, while it is independent of the number of customers and the classifiers used.
- CUST.CLS.: This step contains the training of the classifier and depends on both independent variables, i.e., the number of features and the number of documents. With regards to the other parameters that we fixed, this step also depends on the number of customers. However, this dependency varies according to the classifier employed. But, as we said, we chose the worst-case classifier in terms of computational complexity for the evaluation of computational time. Hence, our evaluation should overestimate the computational time.
- FEAT.EX.REC.: Similarly to the previous FEAT.EX.CLS. step, this depends on both the number of documents and on the set of features used and is independent of the features and classifiers used.
- REC.EV.: In this step the prediction of the scores for each document and each customer is performed using the trained classifiers on the features extracted from the documents. Hence, this task should depend on both the number of documents and the features used. It also depends on the complexity of prediction with the chosen classifier.

In the optimization step of the CUST.CLS. task, we evaluated the hyper-parameter α of the classifier LIN SVM on 7 different

values $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$, using the $F_1 score$ for the optimization. During the training, the validation is performed as in the previous experiment (see Section 5.2.2).

In order to analyse the influence on the computational time of the two independent variables considered, we performed two sets of experiments: (i) we fixed the number of features to all features while changing the number of documents within the set of values {2000, 5000, 10000, 20000, 50000, 100000, 200000, 260000}; (ii) we fixed the number of documents at 260000 while changing the number of features by combining together various sets of features, as defined in Section 4.2: TitleTfIdf (24,000 features), TextTfIdf (70,000 features), TextFasttext (300 features), and SourceBinary (12,000 features).

Figures 2 and 3 show the results of the experiments with the computational times as a function of the two independent variables. We can not a linear relationship of the running time with the number of documents in the interval under evaluation. The pre-processing step PRE-PROC. appears to have the fastest growth, while the step CUST.CLS. appears to grow much slower with the number of documents. Now considering the impact of the chosen features set on the running times of the various tasks, we can see that the PRE-PROC. task takes the longest time, independently of the feature set used. Note that this task is generally performed on a daily basis on a small number of documents (around 2,000). Furthermore, we can notice that the configurations that use the Fastext features take a longer time to execute the FEAT.EX.CLS. and the FEAT.EX.REC. tasks.

These results clearly show that one can perform the required daily and weekly activities respecting temporal constraints imposed by a realistic scenario. A typical daily activity requires processing around 1,000 documents, and here we can see that the required running time is below 5 minutes if there are fewer then 5,000 documents. Regarding weekly activity, the most time-consuming tasks are processed one time, and last less than 1 hour in the worst case. The training of the customer's classifiers and the prediction of the relevance of each document require less than 15 minutes for one customer.

6 DISCUSSION AND CONCLUSIONS

In this paper we presented a system that reduces the daily work of an editor for generating press releases from documents provided by a media monitoring system. Our system takes the documents identified by the Media Monitoring system as potentially relevant for a customer and classifies them into relevant (those that don't need further attention by the editor) and others, which require the additional inspection of the editor. Our classification approach uses past actions of the editor (i.e. the removal of documents) and features extracted from the textual content of the documents (using TF-IDF and embeddings). The results of offline experiments show that very high precision and recall can be achieved. This means that our system can identify a set of relevant documents with few false positives (i.e. the documents marked as relevant were mostly truly relevant) and with few false negatives (i.e. there were few relevant documents in the ones not marked as relevant).

As an example, customer 62 received on 30/May/2018 a total of 49 documents from the keyword-filtering module. After a thorough inspection of all the documents the editor marked 10 documents as non-relevant and kept the remaining 39 as relevant. Our system, optimized both for precision and recall, was able to mark 20 of the 39 truly relevant documents as relevant, hence reducing the workload of the editor by more than half.

We also measured the computational time needed to train the classifiers and to perform the recommendations. As a reference point, we took the load of customers and documents in the real IJ system. We showed that the various tasks necessary to implement the proposed approach can be executed with off-the-shelf hardware in reasonable time and that they do not require altering the current workflow. The results indicate that the computational complexity might be linear in the number of documents processed, hence the proposed approach is scalable.

In the future we plan to improve our system. One line of research will be to identify the documents that are clearly non-relevant, hence shrinking even more the grey-area of documents that the editor needs to inspect. In another line of research, we plan to evaluate the performance in a multilingual scenario, using documents of different languages (Italian, German, and English) and language-independent features. In addition, we plan to collect further feedback from the editors and also from the end users of the system, in order to design solutions to support the creation of press releases. Finally, an online evaluation with the editors will be performed to confirm the obtained results in a real scenario.

ACKNOWLEDGEMENT

This work was supported by the autonomous province of Bolzano-Bozen (Alto Adige-Südtirol) under the EUCLIP_RES project (EUregio CrossLinguistic REcommender System).

REFERENCES

[1] 2018. Cision Official Website. https://www.cision.com/us/ [Online; accessed 27-November-2018].
[2] 2018. Meltwater Official Website. https://www.meltwater.com/uk/ [Online; accessed 27-November-2018].
[3] 2018. Mention Official Website. https://mention.com/en/ [Online; accessed 27-November-2018].
[4] 2018. News Exposure Official Website. https://newsexposure.com/ [Online; accessed 27-November-2018].
[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.
[6] Iván Cantador, Alejandro Bellogín, and Pablo Castells. 2008. News@hand: A semantic web approach to recommending news. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. Springer, 279–283.
[7] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 271–280.
[8] Aurélien Géron. 2017. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.".
[9] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications* 13, 4 (1998), 18–28.
[10] Ilija Ilievski and Sujoy Roy. 2013. Personalized news recommendation based on implicit feedback. In *Proceedings of the 2013 international news recommender systems workshop and challenge*. ACM, 10–15.
[11] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News recommender systems–Survey and roads ahead. *Information Processing & Management* (2018).
[12] Michal Kompan and Mária Bieliková. 2010. Content-based news recommendation. In *International conference on electronic commerce and web technologies*. Springer, 61–72.

[13] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 31–40.
[14] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Text classification and Naïve Bayes. *Introduction to information retrieval* 1, 6 (2008).
[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.
[16] Stephen D Rappaport. 2010. Listening solutions: A marketer's guide to software and services. *Journal of Advertising Research* 50, 2 (2010), 197–213.
[17] Ioannis Stavrakantonakis, Andreea-Elena Gagiu, Harriet Kasper, Ioan Toma, and Andreas Thalhammer. 2012. An approach for evaluation of social media monitoring tools. *Common Value Management* 52, 1 (2012), 52–64.
[18] Aixin Sun, Ee-Peng Lim, and Ying Liu. 2009. On strategies for imbalanced text classification using SVM: A comparative study. *Decision Support Systems* 48, 1 (2009), 191–201.
[19] Boyang Zhang and Marita Vos. 2014. Social media monitoring: aims, methods, and challenges for international companies. *Corporate Communications: An International Journal* 19, 4 (2014), 371–383.