Improving Multi-label Document Classification of Czech News Articles

Jan Lehečka(✉) and Jan Švec

Department of Cybernetics, Faculty of Applied Sciences,
University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic
{jlehecka,honzas}@kky.zcu.cz
http://www.kky.zcu.cz

Abstract. In this paper, we present our improvement of a multi-label document classifier for text filtering in a corpus containing Czech news articles, where relevant topics of an arbitrary document are to be assigned automatically. Different vector space models, different classifiers and different thresholding strategies were investigated and the performance was measured in terms of sample-wise average F_1 score. Results of this paper show that we can improve the performance of our baseline naive Bayes classifier by 25% relatively when using linear SVC classifier with sublinear *tf-idf* vector space model, and another 6.1% relatively when using regressor-based sample-wise thresholding strategy.

Keywords: Multi-label classification · Topic identification · Threshold selection · Thresholding strategy

1 Introduction

With growing volume of electronic text documents available online, text filtering systems are increasingly required, especially in large text corpora. For purpose of text filtering, each document in the corpus can be associated with one or more label describing the topic of the document (e.g. "football", "politics" etc.). The natural requirement for modern text filtering systems is to assign these labels for an arbitrary document automatically. When the process of revealing topics from a text is based on a supervised learning (e.g. a classifier trained from manually labeled data), it is called multi-label document classification task.

Almost any binary classifier can be used for multi-label document classification using *one-vs-the-rest* strategy, which trains one binary classifier per label. Such multi-label classifier outputs a vector of scores for each document (*soft prediction*), which has to be processed by a thresholding strategy to obtain a binary vector (*hard prediction*), that is "true" or "false" for each label.

As for the thresholding strategy, there is a large volume of published studies reporting various attempts. Older attempts are using a fixed threshold to produce hard predictions, like *RCut*, *PCut*, *SCut* [1,2], also dynamic threshold *MCut*, which sets the threshold for each document in the highest difference between successive scores and thus doesn't need any training, has been presented [3]. In recent years, many other various thresholding strategies have been published in [4–6] and in many other papers.

2 Current System and Baseline Classifier

Our current system is a language modelling corpus [7] containing large, constantly growing, volume of text documents from various sources, mainly web-mined news articles. With increasing volume of documents in the corpus, the need of automatic document classification for purpose of data filtering became essential [8].

Currently in this system, a multi-label naive Bayes classifier is used and trained from documents incoming from one selected news server, which we believe to be labeled thoroughly. For each new document, the classifier produces a probability distribution over all topics and N most probable labels are assigned to the document. The number of assigned labels is currently fixed on the average number of labels in the training data, which is $N = 3$, although a threshold selection method has been recently reported to perform better [5].

3 Training and Testing Data

For the purpose of comparing different classifiers and thresholding strategies, we exported from the corpus all labeled data from a selected news server and split them into years of publishing. Then, from each year, we added documents from a 2-months epoch (different for each year) into the testing data set while keeping documents from the rest of the year as the training data set. Using this division technique, all documents were split roughly in ratio 5:1 into training and testing data sets while avoiding the absence of older topics in the testing data set.

Our training data set consists of $205k$ documents ($70M$ words total) with vocabulary size $700k$ and $21k$ different labels. Because of the lack of training data assigned to low-frequency labels, we decided to use only labels assigned to at least 30 documents, which decreased the number of labels to 1843. Our testing data consists of $44k$ documents.

4 Evaluation Metric

For binary classification tasks, where the retrieved output for each tested document is either 1 or 0 (i.e. the document has the label or not), the standard and widely used evaluation metric of a test outcome is the F_1 score

$$F_1 = \frac{2PR}{P+R}, \tag{1}$$

which is the harmonic mean of precision P and recall R, where

$$P = \frac{tp}{tp+fp} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}, \tag{2}$$

$$R = \frac{tp}{tp + fn} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}, \quad (3)$$

where tp are true positives, fp false positives and fn false negatives.

However, to evaluate outcome of multi-label classification test, where the retrieved output for each tested document is binary vector (1 or 0 for each label), some average value of F_1 is needed. There are several options:

- *micro-average*: total true positives, false negatives and false positives are counted across all the documents and used to compute P and R for (1);
- *label-wise average*, also known as *macro-average*: F_1 is computed for each single label and average value is obtained;
- *sample-wise average*: F_1 is computed for each single document and average value is obtained.

We decided to use *sample-wise average* F_1, because it gives us good image, of how well an unknown document would be labeled in average, although it produces slightly lower scores then the other metrics. In the following sections, we use short notation F_1 for *sample-wise average* F_1.

We should also define, how to deal with *out-of-training-data* labels. The standard approach is to use the same set of labels for training and evaluating phase, and thus ignore all labels in the testing data set, which were not seen during the training phase. This approach can be tricky, because one can artificially increase scores from measured metrics by simply selecting fewer labels for training phase and thus ignore more labels in evaluation phase and thus ignore more errors. However, we believe that the better approach is to consider all relevant labels, that were not retrieved by classifier, as a mistake. Of course, the later approach produces lower F_1 scores, because when evaluating documents labeled with *out-of-training-data* labels, it is impossible to obtain high recall, so the upper bound of F_1 score is not 1, but substantially lower. In the case of our data, where we used only labels assigned to 30+ documents, the upper bound of (*sample-wise average*) recall was $R^{ceil} = 0.906$ and $F_1^{ceil} = 0.943$, which is the highest value we could possibly reach.

We believe the later approach better reflects the true performance of a document classifier, therefore we decided to use it as an evaluation metric for our experiments.

5 Vector Space Models

When dealing with a text, direct use of text representation (i.e. a sequence of words) in classification tasks is impractical, because of a variable length of each document. Therefore, it is desirable to convert raw text data to a vector space model, where each instance of data (i.e. each document in a corpus) is represented as a vector of predefined features.

Choosing the right vector space model and the right set of features suitable for particular task, is very important and can have strong impact on document classification performance. In this section, we present most popular vector space models and show their performance in our problem of multi-label document classification of Czech news articles.

Very popular vector space models based on bag-of-words model (i.e. a document-term matrix), which are widely used in document classification tasks, are *tf* (term frequency), *idf* (inverse document frequency) and it's powerful combination *tf-idf*. The *tf* reflects within document frequency of words (terms), whereas *idf* holds information about term frequencies across all documents.

Pure *tf* uses only raw frequencies, $tf_{td} = N(t, d)$, where $N(t, d)$ denotes how many times term t occurs in document d. However in practice, some normalization is usually applied on *tf* model:

– *cosine normalization* is a length-normalization used to scale all values to a $[0, 1]$ while penalizing documents with high individual term frequencies or with many different terms:

$$tfc_{td} = \frac{tf_{td}}{\sqrt{tf^2_{t_1 d} + tf^2_{t_2 d} + ... + tf^2_{t_V d}}} \tag{4}$$

– *sublinear tf* claims that if a term occurs twenty times in a document, it doesn't carry twenty times the significance of a single occurrence, but the significance has rather logarithmic scale:

$$tfs_{td} = \begin{cases} 1 + \log tf_{td} & \text{if } tf_{td} > 0 \\ 0 & \text{else} \end{cases} \tag{5}$$

On the other hand, *idf* reflects how common or rare the term is across all documents. It is the logarithmically scaled inverse proportion of documents containing the term:

$$idf_t = \log \frac{N}{N(t)}, \tag{6}$$

where N is total number of documents and $N(t)$ denotes number of documents containing the term t. If a term occurs in almost every document, it's *idf* will be very low (almost zero), and rare terms that occur few times in whole corpus, will gain high *idf*.

Combination of *tf* and *idf* leads to very powerful vector space models for text classification task. The simple product $tf_{td} \cdot idf_t$ is not commonly used, because it completely ignores terms with $idf = 0$, i.e. terms which occur in every document. Instead, formula

$$tf\text{-}idf_{td} = tf_{td}[idf_t + 1] = tf_{td} + tf_{td} \cdot idf_t \tag{7}$$

is usually used. In this paper, different normalizations of *tf* vector space model were used as both stand-alone model and in combination with *idf* using (7).

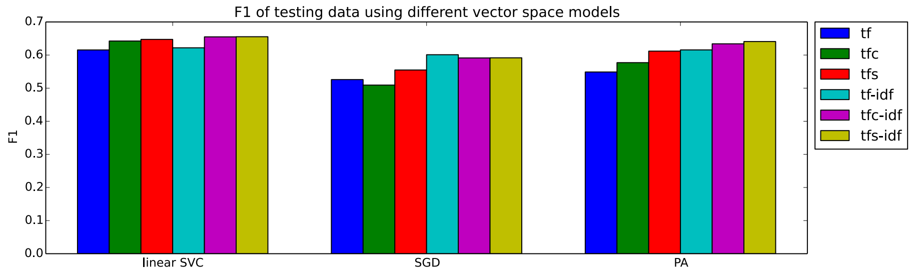

Fig. 1. F_1 score using different vector space models and different multi-label classifiers measured on testing data

5.1 Performance on Our Data

To see which vector space model is suitable for our data, we run following experiment. We converted raw text data to several different vector space models described in this section. For each vector space model, we trained three widely used linear classifiers using all the training data:

- *linear SVC* (linear Support Vector Classifier): SVC with linear kernel function, implemented in terms of *liblinear* [9];
- *SGD* (Stochastic Gradient Descend classifier): linear classifier with stochastic gradient descend learning;
- *PA* (Passive-Aggressive classifier) [10].

We used classifier implementations from *scikit-learn* [11] while leaving all parameters set on default values. To train a multi-label classifier, we used *one-vs-the-rest* strategy, which trains one binary classifier per label (document has the label or not). Then, all the testing data (converted to corresponding vector space model) have been labeled using the trained classifier, and the F_1 score has been measured. Results of this experiment are shown in Figure 1, where *tfc* denotes term frequency with cosine normalization (4) and *tfs* denotes sublinear term frequency (5). As a thresholding strategy, we used *top3* (see section 6.1).

We can see from the graph, that applying normalization on the term frequency, have mostly positive effect on the document classifier performance (in terms of F_1 score). The highest F_1 score has been obtained from *linear SVC* when using *tfs-idf* ($F_1 = 0.6554$), but the difference from *tfc-idf* ($F_1 = 0.6550$) or from the score obtained from *PA* when using *tfs-idf* ($F_1 = 0.6408$), is very small.

For experiments in the next chapter, we will use *linear SVC* classifier trained with data converted to *tfs-idf* vector space model.

6 Thresholding Strategy

A thresholding strategy describes the way how to select a set of relevant labels $\mathscr{L}_d^{rel}$ from a set of all possible labels $\mathscr{L} = \{l_k\}_{k=1}^{K}$ for an arbitrary document d

given it's multi-label classifier's output, i.e. one number (e.g. score, probability etc.) for each label. Most of thresholding strategies can be applied on any type of vectors, but in this paper, we assume the classifier's output for document d is a probability distribution over topics $P_d(k), k = 1, ..., K, \sum_{k=1}^{K} P_d(k) = 1$. In this section, we present some commonly used strategies as well as some our ideas how to select $\mathscr{L}_d^{rel}$.

6.1 Top*N*

Top$N$ thresholding strategy, also known as *RCut* [1] selects N most probable labels. It is very simple strategy with obvious drawback: the same number of labels is assigned to every document ignoring how probable the topics are. N can be set for example as an average number of labels in the training data set.

6.2 Threshold Selection

Next very simple thresholding strategy is to assign only labels of topics with the probability higher then some defined threshold t:

$$\mathscr{L}_d^{rel} = \{l_k \in \mathscr{L} : P_d(k) \geq t\}. \tag{8}$$

However, defining one fixed threshold for all documents can be impractical, because the higher number of relevant labels the document has, the lower corresponding probabilities are. There are many possibilities, how to set the threshold (or more thresholds) in more general way [1–5].

Our approaches are mainly based on learning thresholds from the probability distribution over topics $P_d^{train}(k), d \in \mathscr{D}^{train}, k = 1, ..., K$ of documents from training data set $\mathscr{D}^{train}$, which has been obtained by classifying the training data set after training the classifier. For $\mathscr{D}^{train}$, we also know document's true labels, which can be described by function

$$true(d, l_k) = \begin{cases} 1 & \text{if label } l_k \text{ is the true label of the document } d, \\ 0 & \text{else.} \end{cases} \tag{9}$$

Here, we describe thresholds used in this paper.

Label-wise thresholding: for each label $l_k \in \mathscr{L}$, one threshold t_k was set. First, two sets of probabilities were created for each l_k:

$$\mathscr{P}_k^{true} = \{P_d^{train}(k) : true(d, l_k) = 1, d \in \mathscr{D}^{train}\}, \tag{10}$$

$$\mathscr{P}_k^{others} = \{P_d^{train}(k) : true(d, l_k) = 0, d \in \mathscr{D}^{train}\}. \tag{11}$$

Then, performance of the following label-wise thresholds were investigated:

$$t_k^{true} = \min(\mathscr{P}_k^{true}), \tag{12}$$

$$t_k^{others} = \max(\mathscr{P}_k^{others}), \tag{13}$$

Table 1. Performance of different multi-label document classifiers

	strategy	P	R	F_1
naive Bayes classifier (*baseline*)	top3	0.486	0.607	0.524
SGD	top3	0.687	0.548	0.592
PA classifier	top3	0.742	0.594	0.641
linear SVC	top3	0.759	0.607	**0.655**
linear SVC	t_k^{true}	0.269	0.755	0.376
	t_k^{others}	0.784	0.468	0.554
	t_k^{mean1}	0.765	0.639	0.668
	t_k^{mean2}	0.758	0.672	**0.685**
linear SVC	$t_d^{\mathscr{R}}$	0.811	0.635	0.684
	$t_d^{\mathscr{R}sort}$	0.772	0.683	**0.695**

$$t_k^{mean1} = \frac{\min(\mathscr{P}_k^{true}) + \max(\mathscr{P}_k^{others})}{2}, \tag{14}$$

$$t_k^{mean2} = \frac{\text{mean}(\mathscr{P}_k^{true}) + \text{mean}(\mathscr{P}_k^{others})}{2}, \tag{15}$$

where mean(x) denotes the mean value of set x, min(x) minimal value and max(x) maximal value of x. Now, (8) can be modified for *label-wise thresholding*:

$$\mathscr{L}_d^{rel} = \{l_k \in \mathscr{L} : P_d(k) \geq t_k\}, \tag{16}$$

where t_k can be obtained by any of (12), (13), (14), (15).

Sample-wise thresholding: the threshold t_d for each document d is obtained from a linear regressor $\mathscr{R}$ trained from $P_d^{train}(k)$. Target values for each document $d \in \mathscr{D}^{train}$ were set in the middle of mean probability of document's true labels and mean probability belonging to an irrelevant labels (i.e. in the spirit of (15) but in a sample-wise manner).

After the regressor $\mathscr{R}$ is trained, the probability distribution over topics $P_d(k)$, $k = 1, ..., K$ for document d can be used as an input of $\mathscr{R}$, then threshold $t_d^{\mathscr{R}}$ is returned and (8) can be modified for *sample-wise thresholding*:

$$\mathscr{L}_d^{rel} = \{l_k \in \mathscr{L} : P_d(k) \geq t_d^{\mathscr{R}}\}. \tag{17}$$

We also tried sorting $P_d^{train}(k)$ for each $d \in \mathscr{D}^{train}$ before training the regressor, i.e. we didn't care which label is relevant for the document d , but we rather trained the regressor from differences between successive probabilities. We denote these thresholds as $t_d^{\mathscr{R}sort}$.

7 Conclusion and Future Work

The results of this paper are summarized in Tab. 1. From label-wise thresholds best performed t_k^{mean2} (15), however, the best F_1 score was achieved when generating threshold for each document using regressor with sorted probabilities on input.

As we can see from the table, we can improve (in terms of F_1 score) the performance of our baseline multi-label document classifier by 25% relatively when using linear SVC classifier with *tfs-idf* vector space model, and another 6.1% relatively when using linear regressor to obtain thresholds, which together makes the improvement over baseline 32.6% relatively.

As it seems we are reaching the upper bound where we can go with classical linear classifiers, we'd like to try also neural networks, especially convolutional neural networks, which have recently became very popular in the field of image categorization. It would also be isteresting to compare our document classifier with winning classifiers of recent WISE 2014 challenge [6], where a lot of novel approaches have been introduced.

Acknowledgments. This research was supported by the Grant Agency of the Czech Republic, project No. GAČR GBP103/12/G084. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

1. Yang, Y.: A study of thresholding strategies for text categorization. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2001, pp. 137–145. ACM, New York (2001)
2. Montejo-Ráez, A., Ureña-López, L.A.: Selection strategies for multi-label text categorization. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) FinTAL 2006. LNCS (LNAI), vol. 4139, pp. 585–592. Springer, Heidelberg (2006)
3. Largeron, C., Moulin, C., Géry, M.: MCut: a thresholding strategy for multi-label classification. In: Hollmén, J., Klawonn, F., Tucker, A. (eds.) IDA 2012. LNCS, vol. 7619, pp. 172–183. Springer, Heidelberg (2012)
4. Fan, R.E., Lin, C.J.: A study on threshold selection for multi-label classification. National Taiwan University, Department of Computer Science, pp. 1–23 (2007)
5. Skorkovská, L.: Dynamic threshold selection method for multi-label newspaper topic identification. In: Habernal, I. (ed.) TSD 2013. LNCS, vol. 8082, pp. 209–216. Springer, Heidelberg (2013)
6. Tsoumakas, G., Papadopoulos, A., Qian, W., Vologiannidis, S., D'yakonov, A., Puurula, A., Read, J., Švec, J., Semenov, S.: WISE 2014 challenge: multi-label classification of print media articles to topics. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014, Part II. LNCS, vol. 8787, pp. 541–548. Springer, Heidelberg (2014)
7. Švec, J., Hoidekr, J., Soutner, D., Vavruška, J.: Web text data mining for building large scale language modelling corpus. In: Habernal, I., Matoušek, V. (eds.) TSD 2011. LNCS, vol. 6836, pp. 356–363. Springer, Heidelberg (2011)
8. Skorkovská, L., Ircing, P., Pražák, A., Lehečka, J.: Automatic topic identification for large scale language modeling data filtering. In: Habernal, I., Matoušek, V. (eds.) TSD 2011. LNCS, vol. 6836, pp. 64–71. Springer, Heidelberg (2011)
9. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)

10. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006)
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)