Multilayer classification of web pages using Random Forest and semi-supervised Latent Dirichlet Allocation

Karim Sayadi
University Pierre and Marie Curie
CHArt Laboratory EA 4004
Paris, France
Email: karim.sayadi@upmc.fr

Quang Vu Bui
Ecole Pratique des Hautes Etudes
CHArt Laboratory EA 4004
Paris, France
Email : quang-vu.bui@etu.ephe.fr

Marc Bui
Ecole Pratique des Hautes Etudes
CHArt Laboratory EA 4004
Paris, France
Email : marc.bui@ephe.sorbonne.fr

***Abstract*—The classification of web pages content is essential to many information retrieval tasks. In this paper, we propose a new methodology for a multilayer soft classification. Our approach is based on the connection between the semi-supervised Latent Dirichlet Allocation (LDA) and the Random Forest classifier. We compute with LDA the distribution of topics in each document and use the results to train the Random Forest classifier. The trained classifier is then able to categorize each web document in different layers of the categories hierarchy. We have applied our methodology on a collected data set from *dmoz* and have obtained satisfactory results.**

Index Terms: Semi-Supervised Latent Dirichlet Allocation (LDA), Topic modeling, Web Classification, Random Forest.

I. Introduction

The task of text classification (also known as text categorization) is a standard problem addressed in machine learning and statistical Natural Language Processing (NLP) [1]. In this task, a text is assigned to one or more predefined class (i.e category) labels through a specific process in which a classifier is built, trained and then applied to label future incoming texts. Several Machine Learning algorithms have been applied to text classification, to name a few : Rocchio's Algorithm, N-Nearest Neighbors, Naive Bayes, Decision tree, Support Vector Machine (SVM), and Neural Network. These algorithms have showed good results [1], [2].

With the increasing popularity of the web, text classification was soon applied to web pages motivated by many information retrieval tasks [3] related to the web content. The following applications as described in [4] show this motivation.

First, web pages classification is essential to the development, expansion, and maintenance of web directories, such as those provided by *Yahoo!* [1] or the Directory Mozilla dmoz ODP (*dmoz*) [2] which used to require a considerable human effort to manage. In the aim of automatically maintain those directory services, this work [5] applied the Naive Bayes for an automatic classification based on the content of the home pages of different web sites. The Naive Bayes approach is easy to implement and gaves good results [2].

Second, web search engines usually present the search results in a ranked list. Web pages classification gives us the possibility to have different ranked results lists with different categories. This may help the user to get more insights on what he is looking for when he does not have a well formulated query. Towards this effort, the authors of this work [6] used SVM to cluster the research results into different categories. Although, their approach showed good results, their clustering was flat and did not take into account the hierarchy of the categories.

Third, the task of data extraction from the web, also known as crawling, is a critical problem due to the highly heterogeneous data sources. Web pages classification can help with building a focused web crawler rather than performing a full crawl which is usually inefficient. This article [7] presented a good architecture based on XML to extract data from web sites. This data extraction requires a solid data validation and error recovery rules. For now those rules are being manually edited and the authors emphasized the importance of using classification techniques to generate the rules.

In order to propose a new approach of the issues cited above, this paper presents a methodology for a multilayer soft classification of web pages textual content. Soft classification refers to a set of probabilities distribution representing the features of each web page and multilayer refers to the different layers in the hierarchy of categories (see fig. 5.). In contrast to the related work (section II), our approach (section III) takes into account all the semantic structures (subsection A.) of the text in the web page and classify it accordingly (subsection B.). Our methodology allowed us to obtain good accuracy results on a Data collection from *dmoz* (section IV).

[1]http://www.yahoo.com
[2]http://www.dmoz.org

978-1-4673-7328-9/15/$31.00 ©2015 IEEE

II. Related work

Random Forests have been used in different area of research [8] including : image classification, network intrusion detection, fraud detection, biological activity categorization, etc. Nevertheless, few works have been dedicated to the categorization of web text content using random forests. We briefly review them in the following.

In [9] the authors employed the Random Forest to classify web documents into a hierarchy of directories. The keywords were extracted from the documents and were used as attributes to learn the random trees. The authors used a already implemented version of Random Forest in WEKA 3.5.6 software developed by the University of Waikato and showed that Random Forest performed better than other well known learning methods such as Naive Bayes or the multinomial regression model.

This work [10] introduced a news article classification framework based on Random Forests that are trained on multimodal features (i.e. textual and visual features). To extract the textual feature the authors used an N-gram statistics [3]. Although their multimodal approach only gave a slight improvement in the results compared to the random forests trained on the textual feature, this article proved the capacity of Random Forest to classify texts based on different types of attributes.

Towards a common framework for text categorization based on the Random Forest, the authors of this work [11] presented an improvement in the used methods for building the random forest. A new feature extraction method and a tree selection was developed and synergistically served for making random forest framework well suited for categorizing text. The authors compared their framework with classical machine learning methods used for text classification and showed that their algorithm effectively reduce the upper bound of the generalization error and improve classification performance.

The main issue with learning the random forest classifier is the task of features extraction from the web documents. The work of [10] used the N-grams method, and the work of [9] used a bag of word approach with a simple document frequency selection. In our work, we chose to use the Latent Dirichlet Allocation (LDA) because it showed better information rates than the both approaches [12], [13]. It also allows us to perform a soft classification with a set of probability distributions over the possible classes. But, LDA is a complete unsupervised method that does not allow any control on the computed features. Thus, we decided to use a semi-supervised version based on the works of [14], [15].

III. Methodology

In this section, we present our methodology for a soft multilayer classification of text content in web pages (i.e web documents). The classification is proceeded as following : we first compute the distribution of topics and the distribution of words for each topic in the web documents, this step will allow us to categorize the web documents in the lower level of the hierarchy (layer 3 in Fig. 5.). The second step consists of using the both distributions to learn a classifier and categorize the web documents in a higher level of the hierarchy (layer 1 and layer 2 in Fig. 5.).

We present in the first sub-section the semi-supervised version of the Latent Dirichlet Allocaiton [12], [14], [15] used in the first step and then we present in the second sub-section the Random Forest Classifier [16], [8] used in the second step.

A. Semi-supervised Latent Dirichlet Allocation

We first present the Latent Dirichlet Allocation and the topic latent variables. Then, we define the semi-supervised version with the new variables.

1) The Latent Dirichlet Allocation: Latent Dirichlet Allocation (LDA) by Blei et al. [12] is a generative probabilistic model for collections of grouped discrete data. Each group is described as a random mixture over a set of latent topics where each topic is a discrete distribution over the collections's vocabulary. LDA is applicable to any corpus of grouped discrete data. In this work, we will refer to the standard Natural Language Processing (NLP) use case where a corpus is a collection of documents, and the data are words. In more details, LDA represents documents as mixtures of topics that spit out words with certain probabilities. Unlike traditional clustering algorithms, e.g K-means and uni-gram models, LDA is a mixed-membership model which allow data to arise from a mixture of clusters rather than limiting from one single cluster.

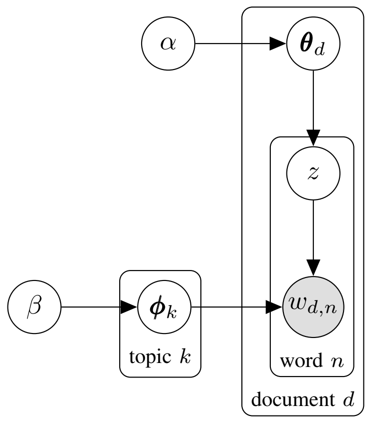

Fig. 1. Bayesian network (BN) of latent Dirichlet allocation. BN is a probabilistic graphical model where the observed variable is in gray, here for example the $w_{d,n}$ word n in the document d. The plates represent the occurrences of the variables or the pattern.

The generative model of LDA [12] is described with the probabilistic graphical model [17] in Fig. 1. In this LDA model, different documents d have different topic proportions θ_d. In each position in the document, a topic z is then selected from the topic proportion θ_d. Finally, a word is picked from all vocabularies based on their probabilities ϕ_k in that topic z. θ_d and ϕ_k are two Dirichlet distribution with α and β as hyperparameters. We assume symmetric Dirichlet priors with α and β having a single value.

The hyperparameters specify the nature of the priors on θ_d and ϕ_k. The hyperparameter α can be interpreted as a prior observation count of the number of times a topic z is sampled in document d [18]. The hyper hyperparameter β can be interpreted as a prior observation count on the number of times words w are sampled from a topic z [18], [19].

The advantage of the LDA model is that examining at the topic level, instead of at the word level, allows us to gain more insights into the meaningful structure of documents, since noise can be suppressed by the process of clustering words into topics. Consequently, we can utilize the topic proportion in order to organize, search, and classify a collection of web documents more effectively.

The key problem in LDA is its posterior inference. This refers to reversing the defined generative process and learning the posterior distributions of the latent variables in the model given the observed data. In LDA, this amounts to solve the following equation:

$$p(\theta, \phi, z|w, \alpha, \beta) = \frac{p(\theta, \phi, z, w|\alpha, \beta)}{p(w|\alpha, \beta)} \tag{1}$$

Unfortunately, this distribution is intractable to compute [20]. The normalization factor in particular, $p(w|\alpha, \beta)$, is intractable. This problem is resolved with a number of state-of-the-art approximate inference techniques : Variational Inference and Gibbs Sampling are widely used in the literature [18]. Within our methodology we propose to use the Collapsed Gibbs Sampling version. This version proved its efficiency and simplicity [21].

The collapsed Gibbs sampler for LDA needs to compute the probability of a topic z being assigned to a word w_i, given all other topic assignments to all other words. Somewhat more formally, we are interested in computing the following posterior up to a constant:

$$p(z_i \mid z_{-i}, \alpha, \beta, w) \tag{2}$$

where z_{-i} means all topic allocations except for z_i.

$$P(z_i = j|z_{-i}, w) \propto \frac{n^{w_i}_{-i,j} + \beta}{n^{(\cdot)}_{-i,j} + V\beta} \frac{n^{d_i}_{-i,j} + \alpha}{n^{d_i}_{-i,\cdot} + K\alpha} \tag{3}$$

where $n^{w_i}_{-i,j}$ is the number of times word w_i was related to topic j. $n^{(\cdot)}_{-i,j}$ is the number of times all other words were related with topic j. $n^{d_i}_{-i,j}$ is the number of times topic j was related with document d_i. $n^{d_i}_{-i,\cdot}$ is the number of times all other topics were related with document d_i. V is the number of words in the vocabulary and K is the number of topics. Those notations were taken from the work of Thomas Griffiths and Mark Steyvers [21].

$$\hat{\phi}^{(w)}_j = \frac{n^{(w)}_j + \beta}{n^{(\cdot)}_j + V\beta} \tag{4}$$

$$\hat{\theta}^{(d)}_j = \frac{n^{(d)}_j + \alpha}{n^{(d)}_{\cdot} + K\alpha} \tag{5}$$

Equation (4) is the bayesian estimation of the distribution of the words in a topic. Equation (5) is the bayesian estimation of the distribution of topics in documents.

2) The semi-supervised version: LDA is widely used topic modeling method in the NLP area. However this methods is unsupervised and therefore does not allow to include knowledge to guide the learning process. In this subsection we present a semi-supervised version of LDA based on the works of [14], [15]. The supervision of the process is within two levels.

```
Require: : Labels for key words, Labels for document.
 1: loop for each iteration
 2:     loop for each document $d$
 3:         loop for each word $w$
 4:             if word labeled then
 5:                 apply equation (7)
 6:                 sample $z_{ij}$ form $T_{i,j}$ based on equation (6)
 7:             else if document labeled then
 8:                 apply equation (8)
 9:                 sample $z_{ij}$ form $T_i$ based on equation (6)
10:             else
11:                 sample $z_{ij}$ form $T$ based on equation (3)
12:             end if
13: Output : Distribution of the words in a topic $\hat{\phi}^{(w)}_j$ and the
    distribution of the topics in a document $\hat{\theta}^{(d)}_j$.
```

Fig. 2. The semi-supervised Latent Dirichlet Allocation algorithm

First, we assign labels at a word level: for each keyword kw_i in set of chosen keywords $SetKW = \{kw_1, kw_2, \ldots, kw_{n_{kv}}\}$, we assign kw_i with a new target topic that is restricted to belong to a set of labels $T_{i,j} = \{T_{i,j_1}, T_{i,j_2}, \ldots, T_{i,j_k}\}$. This step gives us the W_t dictionary where we have for each keywords an associated array of topics $W_t[keyword]$.

Then, at a document level, we label with one or several topics a set of chosen documents. In this step, the new target topic for all words in the labeled document d_i in the set of labeled documents $D_L = \{d_1, d_2, \ldots, d_{M_L}\}$ is restricted to belong to the set of labels $T_i = \{T_{i_1}, T_{i_2}, \ldots, T_{i_k}\}$. This step gives us the D_t dictionary where we have for each labeled document an associated array of topics $D_t[Labeleddoc]$.

For all words in any unlabeled document and unlabeled words, the topic is sampled within the whole topics domain $T = \{T_1, T_2, \ldots, T_K\}$.

Both labeling actions presents constraints for the new computed topics. In semi-supervised LDA when we process a labeled document or a labeled words, it applies the sampling only on the topics subset of the labeled entities. Thus, the sampling equation (3) is modified and replaced by equation (6).

$$P(z_i = j|z_{-i}, w) \propto \frac{n^{w_i}_{-i,j} + \beta}{n^{(\cdot)}_{-i,j} + v\beta} \frac{n^{d_i}_{-i,j} + \alpha}{n^{d_i}_{-i,\cdot} + k\alpha} \quad (6)$$

Where v and k are calculated by the equations (7) for the labeled word process and the equations (8) for the labeled document process :

$$k = |T_{i,j}| = len(W_t[w]); v = |SetKV| \quad (7)$$

$$k = |T_i| = len(D_t[d]); v = V \quad (8)$$

We note that the sampling with a Gibbs processing has the same behavior applied on complete sets or subsets [17]. Our version is similar to the work of [15] where they used a vector-valued for the hyperparameters α and β. In our algorithm we used a single value where $\beta = 0.1$ and $\alpha = 50/K$ as in [21] where they have conducted a series of experiments and have explained the chosen values. We implemented the semi-supervised version of LDA, as described in Fig. 2, in python on top of our own implementation of a standard LDA and Gibbs sampling inference. This implementation is a part of our Library *AMEUR* (Subsection D).

B. Random Forests

Random Forests belong to the machine learning *ensemble methods* [22]. The term 'random forests' originally comes from [23] where in 1995 Ho et al. proposed a model that aggregate a set of decision tree built upon a random subsets. However, this approach encounter an issue in prediction due to *overfitting* (know also as *overlearning*). This problem was approached by Kleinberg in 1996 [24]. In the same year, Breiman [25] proposed the Bagging technique which tends to resolve the problem of *overlearning*. Bagging improve the estimate or prediction itself by averaging this prediction over a collection of bootstrap (random training set) samples. Combing the ideas of decision trees and *ensemble methods*[22], Breiman in 2001 [16], gave use to decision forests, that is, sets of randomly trained decision trees. Random Forest improve bagging by reducing the correlation between the sampled trees by different sources of randomness.

In the random forest, there are two different source of randomness within the processing of building the trees. First, the bootstrap is a technique to build a random set that gives the best accuracy of a parameter estimate or prediction. We Draw a bootstrap (bagging : averaging estimators) sample $\mathbf{Z}^*$ of size N from the training data. Second, the random selection of the attributes to split each node where we draw uniformly at random with replacement N data. (Each tree will have a random different set). The process of building a random forest is described in Fig.3.

```
Require: : Data $D$ of $N$ points.
1: loop for each tree b $\in$ B trees
       Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
2:     repeat
3:         Select $m$ variables at random form the $p$ variables.
4:         Pick the best variable/split-point among the m.
5:         Split the node into two daughter nodes.
6:     until the minimum node size $n_{min}$ is reached.
7: Output : the set of trees $\{T_b\}_1^B$
```

Fig. 3. The random forest algorithm

C. Our Contribution

We propose a methodology of classifying web pages documents. The classification that we perform is a soft classification, where we take into account, within the process, the distribution of the words (topics) in each document and the distribution of the topics in each document. Thus, we do not classify according to one single word, or a set of keywords or a single topic.

Our methodology takes into account the whole hierarchy of the categories that the web documents belong to, allowing to have a multi-layer classification (see Fig. 5.). First, we guide the learning process of LDA by a semi-supervised version to capture the low level categories of each web documents (Layer 3 in Fig.5.). This first step gives us a distribution of the different low level categories for each web document (e.g. Storage, Open Source, Educational, etc ..). Then, we build a spread sheet with the low level categories as features and the higher level categories as values that we want to predict. Finally, we run the Random Forest Classifier on a train data set (See the *WebDocClassif* procedure in Fig. 4.).

```
1: procedure MULTILAYERCLASSIF(WebDocs)
2:     DocsWordFile ← preProcess(WebDocs)
3:     $\theta_d$ ← ss-LDA(DocsWordFile)
4:     csvFile ← buildSheet($\theta_d$)
5:     clf ← RandomForestTrain(csvFile)
6:     predictionOfCategories ← predict(clf)
7: end procedure
```

Fig. 4. This algorithm describes our methodology for classifying web documents on a multiple layers begin from a low level categories and ascends to the high level categories. This classification takes into account the distribution of the topics in the web documents and the distribution of the words in each topic (i.e soft classification).

D. Implementation in python of the library AMEUR

In this part, we briefly present our research development named *AMEUR* where we have developed a library written in python.

The *topicmodeling* module implements generative models like the Latent Dirichlet Allocation, LDA Gibbs Sampling, semi-supervised LDA, that allows us to capture the relationships between discrete data. This module is used within the *AMEUR* library for querying purposes e.g to retrieve a set of documents that are relevant to a query document or to cluster a set of documents given a latent-topic query. The computation of these queries are insured by the connection between the *topicmodeling* module and the *sklearn* module of the *scikit-learn* library [3].

The *nlp* (natural language processing) module implements the necessary functions for getting unstructured text data of different sources from webpages or social medias and preparing them as proper inputs for the algorithms implemented in the rest of the modules of the library.

In the following we present our experiments and the accuracy of our results.

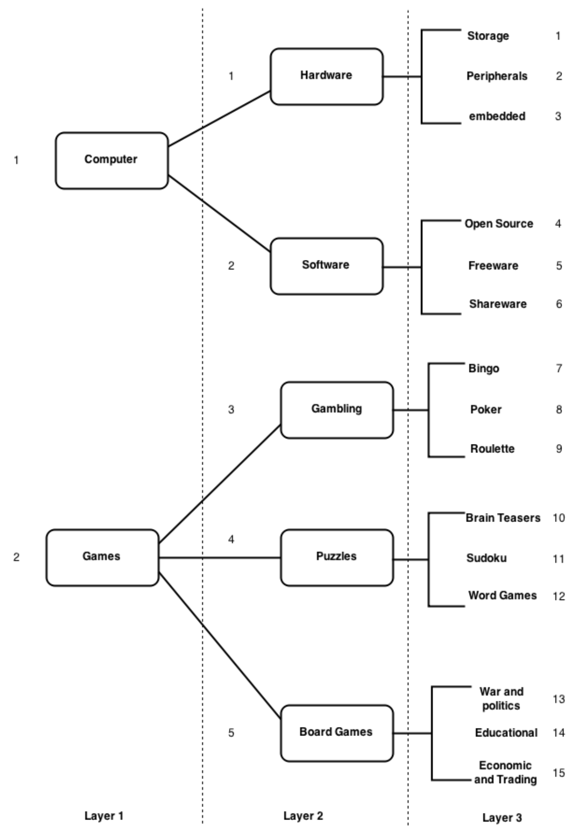

Fig. 5. We have 3 layers, in the first layer we have 2 main category Computer and Games, as we go deep in the layers we have more categories.

IV. Experiments and results

A. Data collection and preprocessing

We have collected text content from the home pages of 60 websites listed in the *dmoz* web directories. As described in Fig. 5, we have 15 low level categories (Layer 3), 5 mid-level categories (Layer 2) and 2 high level categories (Layer 1). For each low-level category we have 4 documents. Before running the *ss-LDA* and building the spread sheet for Random Forest, we first took the text from the home page of each web site and processed it to get good insights. The data processing comes in the following steps and it is described in Fig.6.

```
Require: $WebDocs$                    ▷ List of the Web Documents
 1: procedure PREPROCESS($WebDocs$)
 2:     for $user \in user_{list}$ do
 3:         $WebText \leftarrow$ Decode(UTF8)
 4:         $WebText \leftarrow$ HtmlParse()
 5:         $WebText \leftarrow$ UrlRemove()
 6:         $WebText \leftarrow$ AppoReplace()
 7:         $WebText \leftarrow$ StopwordRemove()
 8:         $WebText \leftarrow$ HExpRemove()
 9:         return CreateDocsWordsFile($WebText$)
10:     end for
11: end procedure
```

Fig. 6. Data collection and pre-processing for the ss-LDA input

First, the home pages usually contains a lot of html entities (e.g. &lt; &gt; &amp), the first step is to get rid of these entities by using the python library *HTMLParser* which converts these entities to standard html tags. Second, decoding the data in a standard encoding format e.g. UTF-8 encoding is widely accepted. To avoid any disambiguation in the text it is better to maintain a proper structure of the text with a free grammar context. Therefore, the third step consist of converting all the apostrophes into standard lexicons e.g {('s: is); ('re : are); ('ll : will); ...}.

The *ss-LDA* analyzes the data at a word level. We then need to remove the commonly occurring words by creating a list of the called stop-words. In the remaining steps we remove punctuations, common words for human expressions and replace the slang words by their actual meaning. We precise that we have not used any stemming techniques because we wanted to keep the different variations of a particular word that could be in different topics. As for the *ss-LDA* we have implemented our own NLP processing version with *Python* as described in Fig.6. In the following subsection, we present our obtained Results.

B. Results

We used the Random Forest implementation of the *Sklearn* package available within the *Scikit-learn* library [4]. The computed spread sheet with *ss-LDA* was divided into two files : 75% for training and 25% for testing. We have 15 attributes, corresponding to the 15 lower level categories (see Fig. 5. Layer 3) and 2 target classes (Layer 1 and 2 in Fig. 5.).

In our analysis, we compare two strategies of web pages classification. The first is comparable to a keyword classification [9] where we don't take into account the results of the classification of the layer above (i.e Layer 1 in Table 1 and Fig. 5.). We called this strategy

[3]http://scikit-learn.org/stable/

[4]http://scikit-learn.org/stable/

A. The second, is a multi-layer classification where the results of the classification of the above layer is taken as an input for the classification of the second layer's web pages. We called this strategy B. We also evaluate in our analysis the effects of the change in the number of trees and put the results in Table 1 where we averaged the results of the 10 times executions of the Random Forest classifier.

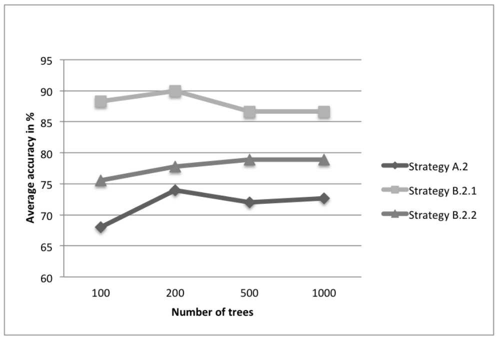

Fig. 7. This graph shows the average accuracy of the strategy A and B of our classification methodology

In the Layer 1 in Table 1 we obtained a classification rate of 93.93% with a null standard deviation. This classification of the web pages in the highest layer, is obtained from the results of the classification in layer 3, where we computed the topics with the *ss-LDA*. For the strategy A, Table Layer 2, we obtained a lowest classification rate of 66.67% and a highest classification rate of 73.33%. For the strategy B, where we took the results of classification of Layer 1 (i.e Layer 2.1 corresponds to classification of the computer category and Layer 2.2 to the Games category), we obtained a lowest classification rate of 77.78% and a highest classification rate of 100%. Thus, as mentioned in the conclusion of the work of [9], the more topics there are, the less accuracy can be obtained. Our methodology offer the use of the strategy B that leads to a better accuracy within the high level description categories (e.g Layer 1).

V. Conclusion and Future Work

We presented in this article a novel approach for classifying the web pages content. The Multi-Layer classification is a connection between the semi-supervised Latent Dirichlet Allocation and the Random Forest Classifier. We obtained a classification rate of 93,33% for the top layer and we improved the results of the classification rate for the lowest layers with a minimum variation. In future work we want to apply this method to a large data set and offer a parallel solution.

TABLE I. This table shows the classification rates of our methodology with different parameters values.

Layer 1				
	100 trees	**200 trees**	**500 trees**	**1000 trees**
Min	93.3%	93.3%	93.3%	93.3%
Max	100%	100%	100%	93.3%
Average	95.9%	95.9%	95.9%	93.3%
Standard deviation	3.4	3.2	2.8	0

Layer 2 (strategy A)				
	100 trees	**200 trees**	**500 trees**	**1000 trees**
Min	53.3%	66.6%	66.6%	66.6%
Max	80%	80%	73,3%	73,3%
Average	68%	74%	71,9%	72,6%
Standard deviation	6,8	5,8	2,8	2,1

Layer 2.1 (strategy B)				
	100 trees	**200 trees**	**500 trees**	**1000 trees**
Min	50%	83.3%	83.3%	83.3%
Max	100%	100%	100%	100%
Average	88.3%	89.9%	86.6%	86.6%
Standard deviation	15.8	8.6	7	7

Layer 2.2 (strategy B)				
	100 trees	**200 trees**	**500 trees**	**1000 trees**
Min	66.6%	66.6%	66.6%	77.7%
Max	88.8 %	100%	88.8%	88.8%
Average	75.5%	77,7%	78,8%	78,8%
Standard deviation	7	10.4	8.1	3.5

References

[1] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[2] Charu C. Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.

[3] Christopher D. Manning and Prabhakar Raghavan. An Introduction to Information Retrieval, 2009.

[4] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys*, 41(2):1–31, February 2009.

[5] Ajay S Patil and BV Pawar. Automated classification of web sites using naive bayesian algorithm. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2012.

[6] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 210–217, New York, NY, USA, 2004. ACM.

[7] Jussi Myllymaki. Effective web data extraction with standard xml technologies. *Computer Networks*, 39(5):635–644, 2002.

[8] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. *Foundations and Trends\r m in Computer Graphics and Vision: Vol. 7: No 2-3, pp 81-227*, 2012.

[9] Myungsook Klassen and Nikhila Paturi. Web document classification by keywords using random forests. In *Networked Digital Technologies*, pages 256–261. Springer, 2010.

[10] Dimitris Liparas, Yaakov HaCohen-Kerner, Anastasia Moumtzidou, Stefanos Vrochidis, and Ioannis Kompatsiaris. News Articles Classification Using Random Forests and Weighted Multimodal Features. In David Lamas and Paul Buitelaar, editors, *Multidisciplinary Information Retrieval*, volume 8849, pages 63–75. Springer International Publishing, Cham, 2014.

[11] Baoxun Xu, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. An Improved Random Forest Classifier for Text Categorization.

Journal of Computers, 7(12), December 2012.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[13] Hanna M. Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM, 2006.

[14] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[15] Youwei Lu, Shogo Okada, and Katsumi Nitta. Semi-supervised Latent Dirichlet Allocation for Multi-label Text Classification. In Moonis Ali, Tibor Bosse, Koen V. Hindriks, Mark Hoogendoorn, Catholijn M. Jonker, and Jan Treur, editors, *Recent Trends in Applied Artificial Intelligence*, number 7906 in Lecture Notes in Computer Science, pages 351–360. Springer Berlin Heidelberg, 2013.

[16] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[17] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[18] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[19] Thomas P. Minka. Estimating a Dirichlet distribution. Technical report, 2000.

[20] Gregor Heinrich. Parameter estimation for text analysis. Technical report, 2004.

[21] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[22] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, November 2009.

[23] Tin Kam Ho. Random decision forests. In *, Proceedings of the Third International Conference on Document Analysis and Recognition, 1995*, volume 1, pages 278–282 vol.1, August 1995.

[24] E. M. Kleinberg. An overtraining-resistant stochastic modeling method for pattern recognition. *The Annals of Statistics*, 24(6):2319–2349, December 1996.

[25] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.