Ensemble Methods for Improving Classification of Data Produced by Latent Dirichlet Allocation

M. JANKOWSKI
maciej.jankowski@wat.edu.pl

Military University of Technology, Faculty of Cybernetics
W. Urbanowicza 2, 00-908 Warsaw, Poland

Topic models are very popular methods of text analysis. The most popular algorithm for topic modelling is LDA (Latent Dirichlet Allocation). Recently, many new methods were proposed, that enable the usage of this model in large scale processing. One of the problem is, that a data scientist has to choose the number of topics manually. This step requires some previous analysis. A few methods were proposed to automatize this step, but none of them works very well if LDA is used as a preprocessing for further classification. In this paper, we propose an ensemble approach which allows us to use more than one model at prediction phase, at the same time, reducing the need of finding a single best number of topics. We have also analyzed a few methods of estimating topic number.

Keywords: Dimensionality reduction, classification, machine learning, natural language processing, topic modelling.

DOI: 10.5604/01.3001.0013.1458

1. Notation

In this chapter, we introduce notation and definitions used in the paper. All symbols are summarized in table 1.

Tab. 1. Notation used in the paper

$\mathbb{N}$	Natural numbers
$\mathbb{Z}$	Integers
$K \in \mathbb{N}$	Number of topics
$V \in \mathbb{N}$	Number of terms in vocabulary
$M \in \mathbb{N}$	Number of documents
$C \in \mathbb{N}$	Number of classes
$N_d \in \mathbb{N}$	Number of words in document d
$N \in \mathbb{N}$	$N = \sum_{d=1}^{M} N_d$. Number of words in a corpus (in all documents)
$L \in \mathbb{N}$	Number of models in ensemble
$d \in \{1, \dots, M\}$	Index of a document
$k \in \{1, \dots, K\}$	Index of topic
$n \in \{1, \dots, N_d\}$	Index of word in a document
$i \in \{1, \dots, V\}$	Index of a word in a vocabulary
$c_d \in \mathbb{N}$	Class label for document d
$l \in \mathbb{N}$	Index of model or a confidence function of a model
$\boldsymbol{w}^{(d)}$	Single observed document, that is $\boldsymbol{w}^{(d)} = \left(w_1^{(d)}, \dots, w_{N_d}^{(d)}\right)$
$w_n^{(d)}$	n-th word in a d-th document
$T = \left\{\boldsymbol{w}^{(1)}, \dots, \boldsymbol{w}^{(M)}\right\}$	Corpus
$W = \{w_1, \dots, w_V\}$	Vocabulary
w_i	i-th word in a vocabulary. We index words in dictionary using i subscript and words in document using n subscript. For words in documents, we additionally add superscript with the index of a document
$e: \{1, \dots, M\} \times \{1, \dots, V\} \to \mathbb{N}.$	Function that for a given document index and word index, returns the number of times, the word appeared in the document
$\boldsymbol{\theta}^{(d)} \in (0,1)^K$	Parameters that control distribution of topics in document d. It is a categorical distribution, that is, a multinomial distribution, where number of experiments is equal to 1

$Z^{(d)} \sim \boldsymbol{\theta}^{(d)}$	Random variable that describes a topic assignment for words in document d. This random variable takes values according to distribution $\boldsymbol{\theta}^{(d)}$
$\theta_k^{(d)} \in (0,1)$	$\theta_k^{(d)} = P(Z^{(d)} = k)$ probability of topic k in document d
$\boldsymbol{\theta} \in (0,1)^{M\times K}$	Matrix of probabilities of topics. Each d-th row, defines distribution over topics for d-th document
$\boldsymbol{\beta}_k \in (0,1)^V$	Parameters that control distribution of terms in topic k
$\beta_{k,i} \in (0,1)$	$\beta_{k,i} = P(w_i\|Z = k)$ probability of i-th word from vocabulary, if topic index is k
$\boldsymbol{\beta} \in (0,1)^{K\times V}$	Matrix of topics. Each row defines distribution over vocabulary.
$S_l: T \to [0,1]^C$	Confidence function for model l
$R_l: W^+ \to \{1, 2, \dots, C\}$	l-th classifier in an ensemble
$E: W^+ \to \{1, 2, \dots, C\}$	Ensemble of L classifiers
$F_d^{MV} \in \{0,1\}^{L\times C}$	Matrix for majority voting classifier. It contains results of L classifiers for a single document
$\left(F_d^{MV}\right)_{lc} \in \{0,1\}$	This element is equal to 1, if l-th classifier chooses class c, and 0 otherwise
$F_d^{WMV} \in (0,1)^{L\times C}$	Matrix for weighted majority voting classifier. It contains results of L classifiers for a single document
$\left(F_d^{WMV}\right)_{lc} \in (0,1)$	Single element of matrix F_d^{WMV} contains confidence that l-th classifier gives to c-th class.

2. Introduction

In this paper, we experiment with a few ensemble algorithms that consists of many LDA models with different number of topics. We show, that those ensembles work better than any single model. In addition to improving accuracy, this approach let us choose a set of numbers of topics, and use them all for prediction. This is a simplification in comparison to previous methods, where a data scientist had to experiment with many models and finally choose one of them.

The rest of this paper is structured as follows: in the third chapter, we briefly introduce dimensionality reduction. In the fourth chapter, we introduce a definition of a language model. In chapter 5, we give a definition of classification, and shortly describe Random Forest classifier. In chapter 6, we focus on text processing, and the role of dimensionality reduction in text analysis. In chapters 7–9, we describe three methods that can be used to reduce dimension of texts: LSA, PLSA and LDA. In chapter 10, we introduce the problem of choosing the best number of topics, and review some of the contemporary methods to tackle this issue. In chapter 11, we introduce a concept of ensemble learning and provide description of three ensemble methods. In chapter 12, we describe results of numerical experiments for ensemble methods.

3. Dimensionality Reduction

According to [10], one of the problem of contemporary data analysis is the fact, that data are characterized by high dimensionality. This causes the following problems

- Noise accumulation
- Spurious correlation
- Incidental homogeneity
- Heavy computational cost.

Therefore, a very important aspect of data analysis is the process of dimensionality reduction. There are two approaches to this task: feature extraction and feature transformation. Feature extraction relates to the method of finding a subset of features, that convey most of the information about the data. A very good introduction to this technique can be found in [11]. Feature transformation relates to the process of embedding the data in a lower dimensional space while retaining as much information as possible. Many methods have been proposed so far, including linear models like PCA, MDS, and nonlinear like Kernel PCA and Isomap. A good introduction to nonlinear methods can be found in [13]. In this work, we are interested in dimensionality reduction of texts in the context of classification.

4. Language model

Let $W = \{w_1, \dots, w_V\}$ be the vocabulary containing V words. A document in the language is a sequence of words $w_1 w_2 \dots w_s$, where $s \geq 1$

and $w_i \in W$. We will define W^+ to be the set of all documents with the vocabulary W.

A *language model* is defined by a finite set W and a function $p(w_1, w_2, \dots, w_s)$ such that

1. $\forall_{\langle w_1,\dots,w_s\rangle \in W^+}\, p(w_1, \dots, w_s) \geq 0$
2. $\sum_{\langle w_1,\dots w_s\rangle \in W^+} p(w_1, \dots, w_s) = 1$

That is, $p(w_1, \dots, w_s)$ is a probability distribution over documents in W^+.

We define a corpus $T \subset W^+$, as a set of observed text documents that are used for training and testing a classifier.

5. Classification

Classification is a supervised method. Suppose we are given a set of IID data $(\boldsymbol{w}^{(1)}, c_1), \dots, (\boldsymbol{w}^{(M)}, c_M)$, where $\boldsymbol{w}^{(d)} \in T$ and c_d takes values in some finite set $\{1,2, \dots, C\}$. In this definition, C denotes a number of classes. A classifier, is a function

$$R: W^+ \to \{1, \dots, C\}$$

that for a given new document, returns an index of a class to which the document belongs.

The function R, usually depends on some free parameters. Learning a classifier can be seen as a process of finding those parameters based on learning examples $(\boldsymbol{w}^{(d)}, c_d), d = = 1, \dots, M$.

One of the most popular classification algorithm is Random Forest [4]. It is based on combining many decision trees. A single decision tree can learn training dataset quite well, but it is prone to overfitting. Therefore, its prediction on a test set is characterized by high variance. To alleviate the problem, Random Forest combines hundreds of decision trees, each with only a subset of features. The more trees are added to the model, the lower is the variance.

6. Application of dimensionality reduction in Information Retrieval

A very important application of dimensionality reduction can be found in Natural Language Processing tasks, e.g. information retrieval.

Let $M \in \mathbb{N}$, be the number of documents and $V \in \mathbb{N}$, be the vocabulary size. Suppose that we have a corpus, that is, a set of documents $T = \{\boldsymbol{w}^{(1)}, \dots, \boldsymbol{w}^{(M)}\}$, where each document $\boldsymbol{w}^{(d)} \in T, d = 1, \dots, M$, contains some number of words from a set $W = \{w_1, \dots, w_V\}$. Before documents can be processed they undergo a preprocessing and resulting elements are called terms. However, in this paper, we will use both word and term interchangeably.

A very common approach in document modelling is assuming, that words order does not matter. The resulting model is called *Unigram Language Model*. This simplification makes text processing much simpler and, in many cases, works sufficiently well. A different approach is to assume, that a particular word depends on n previous words. This is called *N-gram Language Model*. A special case of it is *Bigram Language Model*, in which, each word depends only on the previous word.

When working with Unigrams, the first step in text processing is to convert each document in a corpus to a vector of numbers, each of which represents counts of terms appearing in a document. This is called *Term Frequency* matrix (TF). We define it as a function

$$e: \{1, \dots, M\} \times \{1, \dots, V\} \to \mathbb{N}. \qquad (1)$$

Each element $e(d, i), d = 1, \dots, M, i = 1, \dots, V$ describes a number of times word w_i appeared in document $\boldsymbol{w}^{(d)}$. Usually, we do not work directly with TF matrix, because some words are more important than others, and we want to include this knowledge in our model. One of the solution, is to use a scaling factor, known as *Inversed Document Frequency* (IDF). The idea is, that words that appear in many documents, have lower discriminating power, than words that appear in fewer documents.

The simplest classification procedure based on TF-IDF matrix, could be implemented as follows:

Learning phase:

1. Create TF-IDF matrix from labeled data;
2. Aggregate rows by class. That is, all rows pertaining to a single class, are summed and divided by number of documents in the class.

The resulting matrix has a number of rows, equal to a number of classes.

Prediction phase:

1. for each new example, calculate cosine similarity between the example and each row in aggregated TF-IDF data;
2. Choose the row with the highest cosine similarity. This row uniquely indicates the class.

Time complexity of this solution at prediction phase is $C \times M$, where C is the number of classes and M is the number of documents at prediction phase. However, it can be improved by using *K-means* algorithm and *Inverted Index* approach, that is, calculate cosine similarity only between documents that have at least one common term. The big advantage of this method is, that it scales for large number of classes. The disadvantage is, that there are other classification models, like *Naïve Bayes*, that achieve similar performance at a lower time complexity.

The problem with TF/TF-IDF representation is, that the resulting matrix is very wide. It is not uncommon to have hundreds of thousands of columns relating to terms extracted from corpus. To overcome the issue, a number of methods have been devised, to transform this matrix to a matrix with much lower number of columns. One of the first method is known as *Latent Semantic Analysis* (LSA), proposed in [12]. An extension of this method was proposed in [1] and is called *Probabilistic Latent Semantic Analysis* (pLSA). Further improvement was presented in [14] and is called *Latent Dirichlet Allocation* (LDA). We will describe those methods briefly in the subsequent chapters.

7. Latent Semantic Analysis

The first method, that can be used to reduce dimensionality of TF/TF-IDF matrix, is called Latent Semantic Analysis (LSA) and was first described in [12]. The idea is, that there are hidden concepts in documents that can be used, instead of the actual terms. The number of those concepts is potentially lower than the number of terms. To find the concepts, the original matrix X, is factorized into three matrices U, Σ and V, using linear algebra method known as *Singular Value Decomposition* (SVD)

$$X = U\Sigma V^T \tag{2}$$

If input matrix X, contains *terms* × *descriptions*, then the interpretation of the three matrices is that: matrix U connects terms to concepts, matrix V relates descriptions to concepts, and matrix Σ gives the strength of each of the concepts. Matrix Σ is always diagonal.

To reduce the dimensionality of data, we set the smallest of the singular values of matrix Σ to 0. Then, we can also eliminate corresponding rows of U and V. The graphical representation of the reduced matrices is presented in Figure 1.

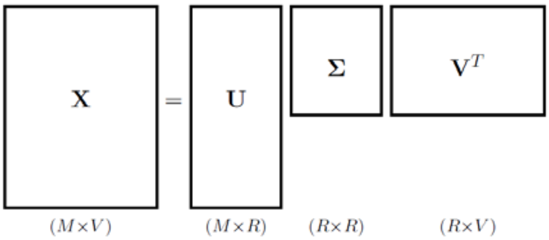

Fig. 1. LSA factorization. Matrix X is factorized into three matrices U, Σ, V as in equation (2)

Then, we train a classifier on a train set $U\Sigma$. If X' contains a new unlabeled data, that we wish to classify, we have to transform it into a lower dimensional space. To do that, we multiply it by V. Data $X'V$ constitute our test set on which, we can perform prediction.

8. Probabilistic Latent Semantic Analysis

The LSA method is based on SVD, which is a linear algebra method, and does not have an easy interpretation in terms of a language model. A big improvement in this area is a method proposed by Hoffman [1] called Probabilistic Latent Semantic Analysis (pLSA). It defines a generative model of data.

In this model, we have $K \geq 2$, distributions over the vocabulary. Each word in a document comes from one of those distributions. The distributions are called topics. We are interested in answering the question: how many times words in document d come from first distribution, second distribution and so on. Ultimately, we want to calculate the probability of each topic given the document. For example, if we have three topics, the following notation $(0.1, 0.3, 0.6)$ means, that ten percent of words come from the first topic, thirty percent come from the second topic and sixty percent come from the third topic. In other words, each document $\boldsymbol{w}^{(d)}$, can be approximated by a vector of real numbers in K dimensional space: $\boldsymbol{\theta}^{(d)} = \left(\theta_1^{(d)}, \dots, \theta_K^{(d)}\right)$, where each $\theta_k^{(d)}$, is a probability of topic k, appearing in document d. Each topic itself is a categorical distribution, controlled by parameters $\boldsymbol{\beta}_k$ for k-th topic.

The analogy to LSA model can be seen through the problem of matrix factorization [8]. Figure 2, shows PLSA problem as a matrix factorization. The equality in the figure does not hold and is only demonstrative.

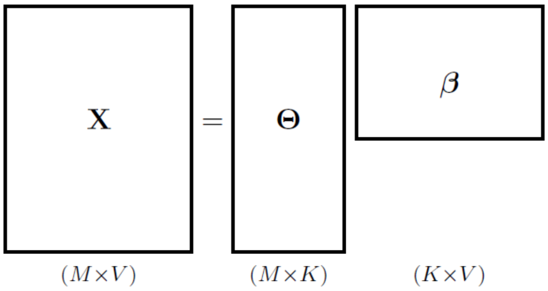

Fig. 2. PLSA factorization

9. Latent Dirichlet Allocation

The above algorithm is not perfect in that, it provides probabilities $\mathbf{\theta}^{(d)} = \left(\theta_1^{(d)}, \dots, \theta_K^{(d)}\right)$ only for observed documents. There is no natural way to extend it to unseen documents.

The first fully Bayesian generative model was proposed in [14]. In this approach, both parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are given Dirichlet prior controlled by hyperparameters α and η. This gives a greater flexibility because, instead of having point estimates of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, we allow them to vary according to some distributions.

The choice of Dirichlet distribution is dictated by the fact, that parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ follow categorical distribution and Dirichlet, is conjugate prior to this distribution. Very good description of this argument can be found in [15]. The generative process, is as follows:

For each topic k:

1. Choose $\boldsymbol{\beta}_k \sim Dir(\eta)$

For each document d in the corpus:

2. Choose $\boldsymbol{\theta}^{(d)} \sim Dir(\alpha)$
3. For each index $n \in (1, \dots, N_d)$
   a. Choose a topic $z_n \sim Mult(\mathbf{\theta}^{(d)})$
   b. Choose a word $w_n^{(d)}$ from $p(w_n^{(d)} | z_n, \boldsymbol{\beta}_{z_n})$, a multinomial probability conditioned on the topic z_n.

Compared to PLSA, we have two additional set of parameters α and η, where: $\boldsymbol{\theta}^{(d)} \sim Dir(\alpha), d = 1, \dots, M$, and
$\boldsymbol{\beta}_k \sim Dir(\eta), k = 1, \dots, K$.
In the original paper [14], authors add prior only for $\boldsymbol{\theta}$ parameter. In [2] authors consider symmetric prior for both $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. In [17] author work with asymmetric prior, that is, all hyperparameters of Dirichlet distribution can have different values. In such a case, it is knowledgeable to optimize for hyperparameters as well. Widely used method for finding α's, known as Minka's fixed point iteration, was given in [22].

Unlike in PLSA, finding $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ in LDA model cannot be done using EM algorithm, because there is no analytical closed form solution for posterior distribution of hidden variables $p(\boldsymbol{Z}|T)$ [14], [2]. We can rewrite the posterior as

$$p(\boldsymbol{Z}|T) = \frac{p(T, \boldsymbol{Z})}{\sum_{\boldsymbol{z}} p(T, \boldsymbol{Z})}$$

For each word $w_n^{(d)}, d = 1, \dots, M, n = 1, \dots, N_d$, we can assign K topics. In the end, the number of topics assignment equals K^N, where N is the number of words in the corpus. Moreover, the sum in the denominator does not factorize.

The main challenge is therefore, to approximate the distribution in an efficient way. In the literature, there are at least four methods that can be used [21], the main being *Variational Inference*, first used for LDA in [14], and *Gibbs Sampling*. The former simplifies the model by making some independence assumption. Inference in the resulting model is done by optimization. The latter is based on MCMC sampling procedures. Although it can be proved, that given infinite number of resources, the MCMC method arrives at the exact solution, in practice, some marginal error is allowed.

10. Estimate topic number

One of the problem with LDA method is, that we need to know the number of topics, before we start to train the model. This is a standard example of model selection. The simplest approach is to run estimation for different values of K, and test each model on a validation set. Finally choosing the number that gives the best performance.

To be able to compare quality of models, we need good evaluation methods. Of course, our interest is in classification, therefore, the better the classification accuracy, the better the model. But, training a classification model for each value of K, may be very time consuming. For this reason, we often want to evaluate models in an unsupervised way.

In this work, we have compared four methods of finding the best value for K. First method was proposed in [2]. The idea is based on *maximum likelihood* approach, which says, that good models give high probabilities to real values. Let T be a corpus, and K be a number of topics. We have run LDA for $K = 2{,}5{,}10{,}20{,}50{,}100{,}200{,}300{,}400{,}500{,}1000$, and then, for each model, we have calculated $\log P(T|K)$. The results are shown in Figure 3. As we can see, the best topic number is somewhere between 100 and 200 topics.

There is a technical difficulty with this approach. We have used implementation of LDA that uses Gibbs Sampling to obtain model parameters. This method does not give a direct estimation of log-likelihood. The presented chart is based on estimator of $\log P(T|K)$ as a harmonic mean of $\log P(T|Z,K)$, where Z is a random vector of topic assignments to words. In the literature we can find at least four other methods of evaluating $\log P(T|K)$. For good review see [24].

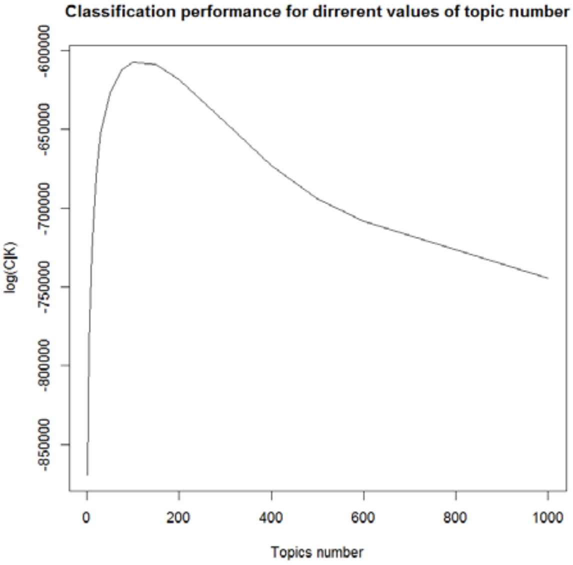

Fig. 3. Log-likelihood for different values of K (topic number)

Another method was proposed in [7]. In this work, authors use average cosine similarity between topics (distributions over words), as a measure of instability of topic structure. They argue, that more stable structures, can better describe data. Let $\boldsymbol{\beta} \in (0,1)^{K \times V}$, be a matrix of topic densities, that is, k-th row is a distribution over words in k-th topic. Then, average cosine similarity is given by

$$avg\_sim(\beta) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{K}\cos(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j)}{\mathrm{K} \times (\mathrm{K}-1)/2}$$

where

$$\cos(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j) = \frac{\boldsymbol{\beta}_i \cdot \boldsymbol{\beta}_j}{\|\boldsymbol{\beta}_i\|_2 \|\boldsymbol{\beta}_j\|_2}$$

Similarly, to the previous experiment, we have trained LDA model for different values of K, and for each model, we have calculated structure stability. The results are presented in Figure 4.

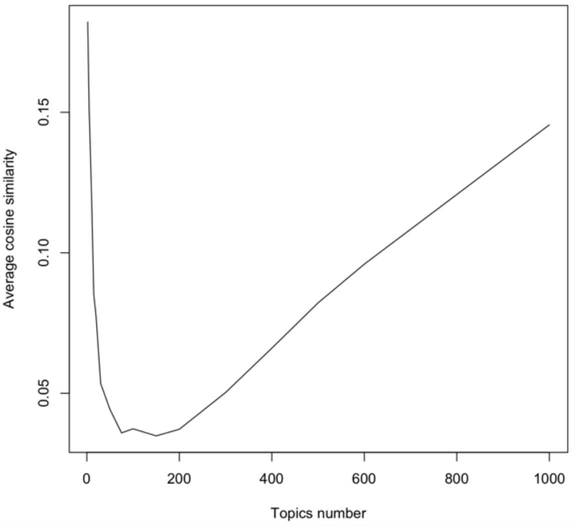

Fig. 4. Structure stability for different values of K (topic number)

It is interesting to note, that the method does not use validation dataset. Instead, the measure is calculated based only on the model parameters.

In [6], authors propose a measure for finding the best K, that is based on viewing LDA as non-negative matrix factorization. Let $\boldsymbol{X} \in \mathbb{R}^{M \times V}$, be a Document-Word frequency matrix. LDA factorizes $\boldsymbol{X}$, into matrices $\boldsymbol{\theta} \in (0,1)^{M \times K}$ and $\boldsymbol{\beta} \in (0,1)^{K \times V}$. However, there is no equality between $\boldsymbol{X}$ and $\boldsymbol{\theta\beta}$, because both $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are stochastic matrices, therefore row normalized. Additionally, there is a vector $B = (N_1, \dots, N_M) \in \mathbb{R}^M$, which contains lengths of documents.

The following measure introduced in [6] can be used to estimate the right value for K

$$A(\boldsymbol{\theta}, \boldsymbol{\beta}) = KL(C_{\boldsymbol{\theta}}\|C_{\boldsymbol{\beta}}) + KL(C_{\boldsymbol{\beta}}||C_{\boldsymbol{\theta}}),$$

where:

$C_{\boldsymbol{\beta}}$ is the distribution of singular values of Topic-Word matrix $\boldsymbol{\beta}$,

$C_{\boldsymbol{\theta}}$ is the distribution obtained by normalizing the vector $\boldsymbol{B\theta}$.

The minimal value of A indicates the best number of topics. We have shown the results for different values of K in Figure 5.

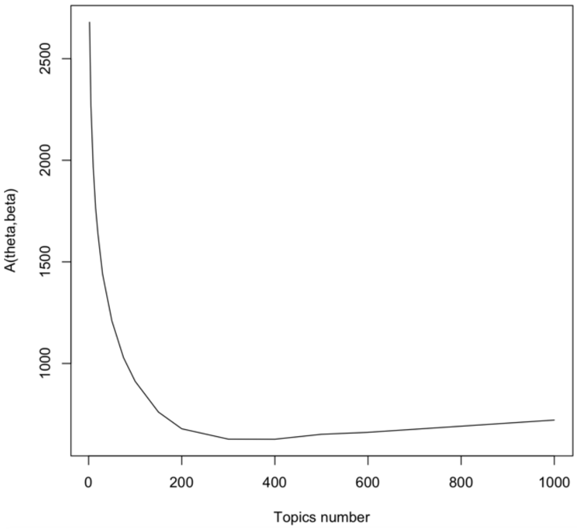

Fig. 5. A-measure for different values of K (topic number)

Yet another technique was proposed in [9]. Let choose some natural number n. For trained model with K topics, $\boldsymbol{\beta}$ is a matrix of its topics. For each topic $\boldsymbol{\beta}_k$, we define a set $W_k^{(K)}$, that contains n words that have highest probabilities in this topic. The best number of topics, according to [9], can be calculated using the following formula

$$\widehat{K} = \underset{K}{\operatorname{argmax}} \frac{1}{K(K-1)} \sum_{(\boldsymbol{\beta}_k, \boldsymbol{\beta}_{k\prime})} D(\boldsymbol{\beta}_k || \boldsymbol{\beta}_{k\prime})$$

where $D(\boldsymbol{\beta}_k || \boldsymbol{\beta}_{k\prime})$ is Jensen-Shannon divergence between two distributions $\boldsymbol{\beta}_k$ and $\boldsymbol{\beta}_{k'}$ defined as

$$\begin{aligned} D(\boldsymbol{\beta}_k || \boldsymbol{\beta}_{k\prime}) &= \frac{1}{2} \sum_{i \in W_k^{(K)} \cap W_{k\prime}^{(K)}} \beta_{k,i} \log \frac{\beta_{k,i}}{\beta_{k\prime,i}} \\ &+ \frac{1}{2} \sum_{i \in W_k^{(K)} \cap W_{k\prime}^{(K)}} \beta_{k\prime,i} \log \frac{\beta_{k\prime,i}}{\beta_{k,i}} \end{aligned}$$

Results are shown in Figure 6.

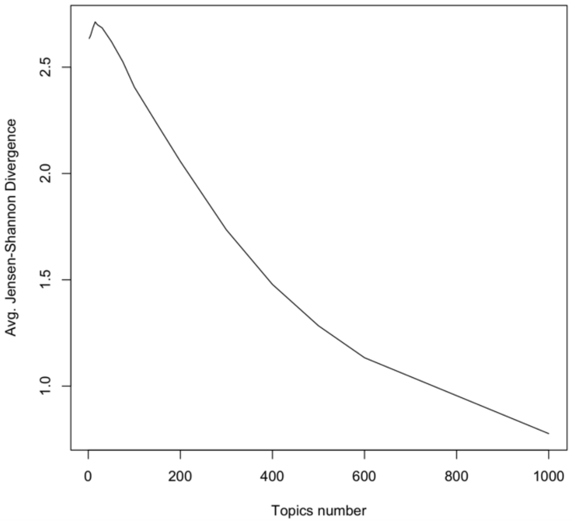

Fig. 6. A-measure for different values of K (topic number).

The first method [2] is based on maximum likelihood estimation. It is known, that this method tends to overfit. It is caused by the fact, that maximum likelihood, will always promote more complicated models over simpler ones, because models with more parameters, can usually better model training data. The problem is, that such complicated models, may not generalize well to unseen examples. This problem is not that severe in our example, because we are reducing dimension, which is some sort of smoothing.

10.1. Finding topic number using entropy of topics

As we can see, there is a great discrepancy between the results. We are interested to assess the quality of those methods in the context of classification. For each topic model, that we have used in previous experiments, we have run Random Forest algorithm, where instead of training on corpus T, we are training on $\boldsymbol{\theta}$. In this approach, we replace each document with respective row from matrix $\boldsymbol{\theta}$. For each classification model, we report its accuracy. Accuracies for different number of topics, are presented, together with results from previous section in Figure 7. In order to be able to compare results, we have normalized them. The interesting values are between 0 and 200, therefore, we do not include values for topic numbers above 200.

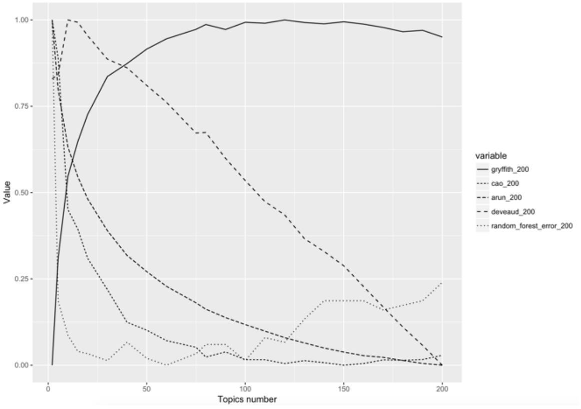

Fig. 7. Classification accuracy vs. topic number estimation

As we can see, none of the method of estimating topic number managed to predict, which value of K, will work best for Random Forest.

In previous chapter, we have introduced a method proposed in [7], that uses dependency between topics as a measure of model power. It is interesting to see, whether topics itself, can be used to asses predicting capacity of a model. For that, we have calculated entropy of each topic, to see how focused each topic is. Lower values of entropy mean, that distribution is concentrated around small number of terms, while high values, indicate that the distribution is more evenly spread across all terms. We expect, that more specific topics, are better for prediction tasks. Results are shown in Figure 8.

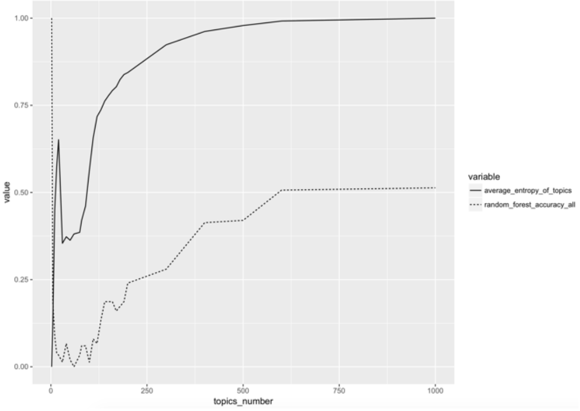

Fig. 8. Classification accuracy with average entropy

As we can see, for the chosen dataset, there is clear correlation between classification accuracy, and average entropy of topics.

10.2. Finding topic number using LSA

Although LSA is a different model than LDA, they share some similarities. Training LSA is much faster than training LDA. Therefore, we are interested, if the best number of topics for LSA is also good for LDA. In the experiment, LSA was executed for values 2, 7, 12, 17, 22,…, 997. As shown in Figure 8, initially the error rate was decreasing, finally achieving the level of 0.302 for 32 topics. For topic number bigger than 32, the error rate is only increasing, up to the value of 0.415 for 862 topics. The comparison of error rate for data produced by LSA and LDA is presented in Figure 9.

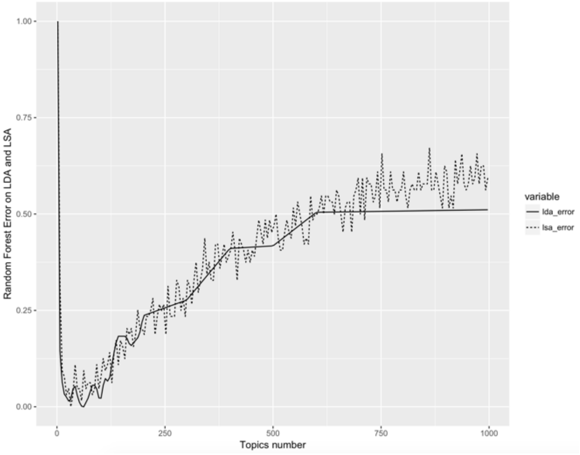

Fig. 9. Classification performance for different number of dimensions of LSA and LDA models

Of course, the results are not the same as for LDA, but we can observe that results are very similar.

In cases, where we have fast classification model, end training it many time is not a problem, but running LDA for many different number of topics is prohibitively expensive, we can actually train LSA. Then, we train a classifier on those LSA models. Based on classification performance, we choose the best number of topics.

11. Ensemble

In the context of classification, choosing the best number of topics, may not be possible using only unsupervised methods. To explain the argument, we will use similar argument that was presented in [25]. If, for example, our task is to classify movies as good or bad, we want to find topics that are somehow related to sentiments. However, if the dominant structure in reviews is genre, this is something that may be found. In [25], authors propose a method, that includes response variable into the model. This variable, influences choice of topics.

There may not be a single best number of topics for the entire corpus. The value of K, that works well for some documents, may not be good for others. This idea, was studied in [3], where authors built a model based on Dirichlet Process. In this model, each document, can have a different number of topics. Authors report improvements in the modelling ability of this new model. We propose another approach, based on ensemble of topic models.

One of the most promising area of research in machine learning to improve classification accuracy, is building ensemble of classifiers. The idea is very well known in ensemble of decision trees, e.g. Random Forest, Bagging and Boosing (see [20] for reference). In this paper, we have created an ensemble of LDA models, where each model uses different number of topics. The results show a slight improvement in classification accuracy, compared to the usage of a single model.

We have used three different strategies for merging results from models: majority voting, weighted averaging and perplexity based chooser, which chooses model with lowest perplexity on validation set.

For the purpose of this chapter, we will define a confidence function S, that for each document, returns a vector of confidences, that is, values between 0 and 1 that describe how confident the model is of each class

$$S_l\colon W^+ \to [0,1]^C.$$

Each confidence function S_l, defines a classifier R_l in the following way

$$R_l\left(w^{(d)}\right) = \mathrm{argmax}_{c\in\{1,2,\ldots,C\}}\, S_l\left(w^{(d)}\right)_c .$$

In the above description, $S_l\left(w^{(d)}\right)$ is a vector with C elements. Subscript c in the above equation means, that we are taking c-th element from this vector.

Given a set of confidence functions, $S_1,\ldots,S_L$, ensemble $E\colon W^+ \to \{1,2,\ldots,C\}$, of models is defined as a function f on those confidence functions. For a given document $\boldsymbol{w}^{(d)} \in W^+$,

$$E\left(\boldsymbol{w}^{(d)}\right) = f(S_1\left(\boldsymbol{w}^{(d)}\right),\ldots,S_L\left(\boldsymbol{w}^{(d)}\right)).$$

The exact behavior of the ensemble depends on the function f. We assume, that each model in the ensemble, consists of different number of topics. At prediction phase, each model returns for each document a confidence. Confidence describes, how confident the model is of each class.

We have analyzed three methods of building an ensemble:

1. Majority voting;
2. Weighted majority voting;
3. Perplexity driven ensemble.

11.1. Majority voting

Majority voting for classification is a method, that chooses class, that was chosen by most of the models. Assume, that that confidence function S_l^{MV} returns one-hot vector, that is, it contains only single 1 and all other values are 0. Let $F_d^{MV} \in \{0,1\}^{L\times C}$, be a matrix that contains confidence vectors for each confidence function S_l^{MV} in an ensemble for a single document d. Row l contains confidence row of model l. A single element of this matrix $\left(F_d^{MV}\right)_{lc}$ is equal to 1 if model l chooses class c and 0 otherwise. Majority voting is defined as

$$E^{MV}\left(\boldsymbol{w}^{(d)}\right) = \underset{c\in\{1,2,\ldots,C\}}{\mathrm{argmax}} \sum_{l=1}^{L} \left(F_d^{MV}\right)_{lc}$$

11.2. Weighted majority voting

This ensemble, for each class, sums confidences of all models and chooses class, for which this aggregate confidence is largest. Formally, let S_l^{WMV} return vector of values between 0 and 1, and $F_d^{WMV} \in (0,1)^{L\times C}$ be a matrix of confidences for L confidence functions and a single document. Then, $\left(F_d^{WMV}\right)_{lc}$ describes how confident model l is of class c (this is a real value between 0 and 1). Weighted majority voting is defined as

$$E^{WMV}\left(\boldsymbol{w}^{(d)}\right) = \underset{c\in\{1,2,\ldots,C\}}{\mathrm{argmax}} \sum_{l=1}^{L} \left(F_d^{WMV}\right)_{lc}$$

The difference between majority voting and weighted majority voting is in the confidence function. For majority voting it is a sparse vector that contains only 0s and 1s, while for weighted majority voting it is a dense vector that contains values between 0 and 1.

11.3. Perplexity driven ensemble

A very popular quantitative method of evaluating language model is *perplexity* [14]. Let $p_l\left(\boldsymbol{w}^{(d)}\right)$ be a probability of obtaining a document $\boldsymbol{w}_d$, under the model M_l. Perplexity is calculated as

$$P_{M_l}\left(\boldsymbol{w}^{(d)}\right) = exp\left\{-\frac{\log p_l\left(\boldsymbol{w}^{(d)}\right)}{N_d}\right\}$$

We have used perplexity to improve performance of ensemble in the following way. For each test document, we have calculated perplexity under each model. The model that gave the lowest perplexity of the document was used for prediction.

$$E^{PERP}(\boldsymbol{w}_d) = M_{g(\boldsymbol{w}_d)}(\boldsymbol{w}_d),$$

where

$$g(\boldsymbol{w}_d) = \underset{l \in \{1,2,\ldots,L\}}{\operatorname{argmin}} P_{M_l}(\boldsymbol{w}^{(d)})$$

This way, we are using only one model for each example, but different models across the whole test set.

12. Experimental results

In this section, we present results of experiments that were carried out on a real-world dataset.

12.1. Mobile Apps dataset

The first dataset contains descriptions of mobile applications together with a label that specify application class. The example classes are: ANDROID_TOOL, APP_LIBRARY, GAME, WIDGET, USE_INTERNET, etc. Overall, there are 24 classes.

Mobile Apps dataset contains 1520 train and 380 test examples. In the first experiment, we have trained 23 LDA models on train dataset. Each model used a different number of topics: 2, 5, 10, 15, 20, 30, 40, 50, 60, 75, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200. Each model allows us to create lower dimensional representation of our data by estimating posterior probability of hidden variables Z. The dimension of this representation is equal to the number of topics for the model.

For each of the reduced dataset, we have trained random forest model. Then, we checked the accuracy of this model for test dataset. The results for the best 9 models are presented in Table 2.

Tab. 2. Comparison of classification errors of 9 best LDA-RF models for Mobile Apps dataset

Number of topics	Error
15	0.3447368
20	0.3500000
30	0.3526316
50	0.3368421
60	0.3394737
75	0.3500000
80	0.3473684
90	0.3657895
100	0.3421053

As we can see, the best result was achieved for 50 topics and is equal to 0.336. In the second experiment, we have created three ensembles as described in section 11. For this purpose, we have used 9 models that achieved the best accuracy. Results are reported in Table 3.

Tab. 3. Comparison of classification errors of single model and ensemble of models

Method	Error
Best single model (50 topics)	0.3368421
Majority voting	0.3368421
Weighted majority voting	0.3236842
Perplexity driven ensemble	0.2971429

As we can see, Majority voting is as good as single best model. Weighted majority voting slightly outperforms majority voting. Perplexity driven ensemble is significantly better than each single model, as well as the two other ensembles.

12.2. Poliblog dataset

The second dataset, is a collection of 773 (573 train, 200 test) political blogs available as part of the LDA R package [5]. Both datasets were split to train and test datasets. Train datasets was used for training models. Final accuracy was measured on test datasets. As previously, we have analyzed dimensionality reduction using LDA model. We have obtained error rates between 0.285 and 0.325. The eleven best models together with number of topics are summarized in Table 4.

Tab. 4. Comparison of classification errors of 8 best LDA-RF models for Sms Spam dataset

Number of topics	Error
20	0.325
30	0.325
40	0.325
50	0.285
60	0.300
75	0.305
90	0.315
160	0.315
170	0.325
190	0.300
200	0.325

The best result that can be achieved by those models is 0.285. As a next step, we have compared results for ensembles as described in section 11. Results are presented in Table 5.

Tab. 5. Comparison of classification errors of single model and ensemble of models

Method	**Error**
Best single model (50 topics)	0.285
Majority voting	0.270
Weighted majority voting	0.265
Perplexity driven ensemble	0.275

As we can see, ensemble methods achieve better results than any single model. It means, that even if we are not able to choose the best model, we can still train an ensemble of many models, and use them all at prediction phase. The result, should not be worse than that of the best model.

13. Summary

In this work, we have proposed a few methods of dealing with an unknown topic number in LDA model. In section 10 we have compared a few existing methods for estimating topic number. We have also proposed a new method based on entropy of topics. If the computational cost of training LDA models for many topics is too high, we have shown, that we can instead train many LSA models, and use them to roughly estimate best topic number. In section 11, we have proposed an ensemble approach, in which we train many models, each with a different number of topics. This let us choose a set of topic numbers and avoid manual experimentation with different models for a single number of topics. We have shown, that our approach not only helps to automatize the choice of topic number, but it also achieves much better results, than any single model.

14. Bibliography

[1] Hofmann Th., “Probabilistic latent semantic indexing”, in: *SIGIR’99: Proceedings of the 22nd annual international SIGIR conference on research and development in information retrieval*, pp. 50–57, ATM, NY, USA, 1999.

[2] Griffiths T., Steyvers M., “Finding scientific topics”, in: *Proceedings of the National Academy of Sciences of the United States of America* 101, pp. 5228–5235, 2004.

[3] Teh Y.W., Jordan M.I., Beal M.J., Blei D.M., “Hierarchical Dirichlet Processes”, *Journal of the American Statistical Association*, Vol. 101, No. 476, 1566–1581 (2006).

[4] Breiman L., “Random forests”, *Machine learning*, 45.1, 5–32 (2001).

[5] Chang J., *lda: Collapsed Gibbs Sampling Methods for Topic Models*, Package v. 1.4.2, https://CRAN.R-project.org/package=lda (2015).

[6] Arun R., Suresh V., Veni Madhavan C.E., Narasimha Murthy M.N., “On finding the natural number of topics with Latent Dirichlet Allocation: some observations”, in: *PAKDD’10 Proceedings of the 14th Pacific-Asia conference on Advance in Knowledge Discovery and Data Mining*, pp. 391–402, Springer, Berlin 2010.

[7] Juan Cao, Tian Xia, Jintao Li, Yongdong Zhang, Sheng Tang, “A density-based method for adaptive LDA model selection”, *Neurocomputing*, Vol. 72, Issue 7–9, 1775–1781 (2008).

[8] Steyvers M., Griffiths T., “Probabilistic topic models”, in: T. Landauer, D. McNamara, S. Dennis, and W. Kintsch (Eds.), *Latent Semantic Analysis*: *A Road to Meaning*, Laurence Erlbaum, 2007.

[9] Deveaud R., Sanjuan E., Bellot P., “Accurate and effective latent concept modeling for ad hoc information retrieval”, *Document numérique*, Vol. 17(1), 61–84 (2014).

[10] Jianqing Fan, Fang Han, Han Liu, “Challenges of Big Data analysis”, *National Science Review*, No. 1, 293–314 (2014).

[11] Guyon I., Gunn S., Nikravesh M., Zadeh L.A. (Eds.), *Feature Extraction. Foundations and Applications*, Springer, 2006.

[12] Deerwester S., Dumais S.T., Harshman R., “Indexing by Latent Semantic Analysis”, *Journal of the American Society for Information Science*, Vol. 41(6), 391–407 (1990).

[13] Lee J.A., Verleysen M., *Nonlinear Dimensionality Reduction*, Springer, 2007.

[14] Blei D.M., Ng A.Y., Jordan M., “Latent dirichlet allocation”, *Journal of Machine Learning Research*, No. 3, 993–1022 (2003).

[15] Heinrich G., “Parameter estimation for text analysis”, *Technical report*, 2005.

[16] MacKay D.J.C., Peto L.C.B., “A hierarchical Dirichlet language model”, *Natural Language Engineering*, Vol. 1, Issue 3, 289–307 (1995).

[17] Wallach H.M., Topic Modeling: Beyond Bag-of-Words, in: *ICML’06 Proceedings of the 23rd international conference on*

Machine learning, pp. 977–984, ACM, NY, USA, 2006.

[18] Minka Th., Lafferty J., "Expectation--propagation for the generative aspect model", in: *UAI'02 Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 352–359, Morgan Kaufmann Publishers Inc., San Francisco, USA, 2002.

[19] Dempster A.P., Laird N.M., Rubin D.B., "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society. Series B*, Vol. 39, No. 1, 1–38 (1977).

[20] Hastie T., Tibshirani R., Friedman J., *The Elements of Statistical Learning*, Springer, 2008.

[21] Asuncion A., Welling M., Smyth P., Yee Whye Teh, "On smoothing and inference for topic models", in: *UAI'09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 27–34, AUAI Press Arlington, USA, 2009.

[22] Minka T.P., "Estimating a Dirichlet distribution", *Annals of Physics*, Vol. 2000, No. 8, 1–13 (2003).

[23] Jurafsky D., Martin J.H., *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Upper Saddle River, New Jersey 2006.

[24] Buntine W., "Estimating Likelihoods for Topic Models", in: *Advances in Machine Learning, ACML 2009, Lecture Notes in Computer Science*, Vol. 5828, pp. 51–64, Springer, Berlin 2009.

[25] Blei D.M., McAuliffe J.D., "Supervised topic models", in: *Advances in Neural Information Processing Systems 20* (*NIPS 2007*), J.C. Platt, D. Koller, Y. Singer, S.T. Roweis (Eds.), pp. 121–128, Cambridge, MA: MIT Press, 2008.

Metody bazujące na Ensemble do klasyfikacji danych wygenerowanych przez model Latent Dirichlet Allocation

M. JANKOWSKI

Modelowanie tematyczne jest popularną metodą analizy tekstów. Jednym z najbardziej popularnych algorytmów modelowania tematycznego jest LDA (Latent Dirichlet Allocation) [14]. Ostatnio zostało zaproponowanych wiele nowych rozszerzeń tego modelu, które pozwalają na przetwarzanie dużych ilości danych. Jednym z problemów podczas użycia algorytmu LDA jest to, że liczba tematów musi zostać wybrana przed uruchomieniem algorytmu. Ten krok wymaga wcześniejszej analizy i zaangażowania analityka danych. Powstało kilka metod, które pozwalają automatyzować ten krok, ale żadna z nich nie działa dobrze, gdy LDA jest użyte do redukcji wymiarów przed klasyfikacją danych. W tej pracy proponujemy podejście oparte na ensemble wielu modeli. Taki model unika problemu wybrania jednego, najlepszego modelu LDA. Pokażemy, że takie podejście pozwala uzyskać niższy błąd klasyfikacji. Zaproponujemy również dwie nowe metody wyboru liczby tematów, gdy chcemy użyć tylko pojedynczego modelu.

Słowa kluczowe: klasyfikacja, redukcja wymiarów, modelowanie tematyczne.