Latent Dirichlet Allocation for Text Clustering

Yiorgos Kalantzis, Molly Rea, Cristiana Tisca

January 2019

1 Introduction

Latent Dirichlet Allocation (LDA) is one of the most popular topic modelling techniques, used in the analysis of large collections of texts. Topic modelling is a statistical method used as a text mining tool for the discovery of hidden semantic structures in a document. Topic models are based on the idea that documents are mixtures of topics. For example, a science paper can have two topics and be 10% about statistics and 90% about genetics.

LDA belongs to the mixed membership subclass of topic modelling techniques. It is a generative probabilistic model which belongs to the class of unsupervised learning techniques. It was introduced in 2003 by Blei et al. [3], and represents a three-level hierarchical Bayesian model, in which each text of a set of texts is modeled as a finite mixture over an underlying set of topics. Hierarchical Bayesian models are written in multiple levels that estimate the parameters of the posterior distribution using Bayesian inference. The sub-models are combined to form the hierarchical model, and Bayes' theorem is used to combine them with the observed data and thus account for the uncertainty in the system. Each resulted topic represents a distribution over terms in the vocabulary, and different topics have different words with different probabilities.

To summarise, following the notation by [4], this model assumes, given $i = 1, ..., N_d$, the document index, $v = 1, ..., N_w$, the word index, and $k = 1, ..., K$, the topic index, that

$$\begin{aligned} \pi_i &\sim \texttt{Dir}(\pi_i|\alpha) && (1)\\ z_{iv} &\sim \texttt{Cat}(z_{iv}|\pi_i) && (2)\\ \mathbf{b}_k &\sim \texttt{Dir}(\mathbf{b}_k|\gamma) && (3)\\ y_{iv} &\sim \texttt{Cat}(y_{iv}|z_{iv} = k, \mathbf{B}) && (4) \end{aligned}$$

where `Dir` and `Cat` are Dirichlet and categorical (multinomial) distributions, respectively, π_i is the topic distribution for the i-th document, z_{iv} is the topic for each word v in document i, and $\mathbf{b}_k$ is the distribution over all of the words in the dictionary for a particular topic k. To generate a particular word, we sample the word y_{iv} given the k-th topic distribution. Finally, α and γ are the parameters for the Dirichlet priors, which tell us how narrow or spread the document topic and topic word distributions are.

LDA is described by its generative process, the imaginary random process by which the model assumes the documents arose. The documents are modeled as being created by the following method: First, a random distribution over topics is picked for a particular document. Then, for each word, a topic is picked according to $p(z)$ (the probability of topic z occurring) and a word is picked according to $p(w|z)$. These probabilities belong to two different Dirichlet distributions, whose priors are characterised by two parameters: α and β (Fig. 1). The Dirichlet distribution is ideal for setting priors and modelling topic proportions in a document. α controls the per-document topic distribution, with a low α value meaning that the document is represented by a small subset of the topics, and a high value making documents to appear more similar to each other.

Consequently, β controls the distribution of the total number of words across each individual topic, with a high beta making topics appear more similar to each other.

In inference, this process is run in reverse: one wishes to obtain the proportional composition of topics for a document, and also learn the distribution of words given a topic for the collection of documents. A distinguishing characteristic of LDA is that all the documents of the collection share the same set of topics, but each document exhibits those topics in different proportions. Thus, when analysing collections of documents, further to implementing LDA, one can cluster documents according to the resulted topic proportions for each individual text.

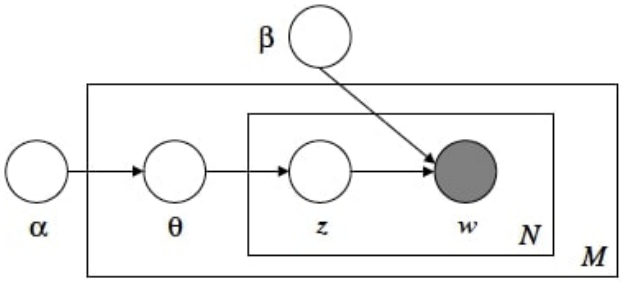

Figure 1: **Graphical model representation of LDA**[3].The boxes are *plates* representing replicates. The outer plate represents M documents, while the N represents all words in the M documents. θ represents the topic distribution for an individual text, Z represents the topic for a particular word in a document, and w represents the observed word in the given document. The grey/filled in circle represents the only known variable, the rest are hidden.

LDA uses the *bag-of-words* (BoW) representation, which is very common in the fields of natural language processing and text mining. Under this model, a document is represented as a a multiset, i.e. a *bag* of words, often omitting punctuation symbols and grammatical order but keeping multiplicity, the frequency of each term. Moreover, words are translated to integers (indices) for practical simplicity and a document can be represented by a set of tuples (j, d) where d is the number of times word j is mentioned in the text.

Example. Assume that one text consists of the following two sentences:

John likes to watch movies. Mary likes movies too.

A BoW representation is $[(1,1),(2,2),(3,1),(4,1),(5,2),(6,1),(7,1)]$ since, for instance, the fifth word (*movies*) has two occurrences. By adding the second element of each pair we get 9 which is the total number of words.

1.1 Clustering, Evaluation & TF - IDF Representation

Clustering is a widely studied data mining problem in the text domains [1]. In brief, a good clustering of a particular dataset is a partition in sets of subjects (for our project, documents) so that subjects of the same cluster are similar, in a way, and subjects from different clusters are dis-similar. As a result, clustering is based on the existence of a similarity function that enables the comparison of data instances. Instead of a similarity function that needs to give high values to similar elements, a distance function $d(a, b)$ is often used; if $d(a, b) < d(a, c)$ then a is assumed to be more similar with b rather than c.

In order to define such a function we need to "translate" the dataset (i.e. lists of texts) to numerical values (i.e. lists of numbers, ideally). As we mentioned previously, one preferred way is the bag-of-words representation. The next step is the construction of the *term frequency–inverse document frequency* (TF-IDF) matrix which is a widely used statistic for measuring the importance of words within a document. This

matrix has dimension $N_D \times N_V$ (#documents × #words) and each element is given by

$$\mathbf{TFIDF}_{dw} = \texttt{tf}(d, w) \cdot \texttt{idf}(w, D)$$

where $\texttt{tf}(d, w)$ stands for the frequency of word w in document d and $\texttt{idf}(w, D)$ is the logarithm of the inverse frequency of word w among a collection of documents D. Sometimes, $\texttt{tf}(d, w)$ is normalised by the length of each document or the highest frequency, but our project employed only the simpler formulation.

This matrix is always really sparse since each document uses only a subset of the available vocabulary. For instance, our dataset has a total of 62959 unique words in 10769 documents and the occurring TFIDF matrix is only 0.19% dense. This representation is very practical as it offers the usage of distances. For the rest of our work, we will be using Euclidean distances which corresponds to the norm-2 of vectors in the N_V-dimensional space.

Having defined distances among each subject we can now define a measure of distance for clusters of subjects. Again, there are various ways to achieve this (e.g. define *centroids* first) and we choose a direct one based on average distances. That is, if clusters A and B have sizes n_A, n_B respectively, then they differ by

$$d(A, B) = \frac{1}{n_A \cdot n_B} \sum_{x \in A} \sum_{y \in B} d(x, y). \tag{5}$$

With this notion, $d(A, A)$ can be seen as a measure of homogeneity within cluster A. As a result, an objective for clustering would be the minimization of the *within-cluster* deviance function $\mathcal{F}$, or the maximization of the *inter-cluster* deviance function $\mathcal{I}$, both defined by

$$\mathcal{F}(A_1, A_2, \dots) = \sum_i d(A_i, A_i) \quad \text{and} \quad \mathcal{G}(A_1, A_2, \dots) = \sum_{i \neq j} d(A_i, A_j). \tag{6}$$

These are two metrics upon which we will rely on for the evaluation of our method and the comparison of different approaches.

1.2 Summary

This report describes a project that focused on LDA implementation on a data set containing Wikipedia articles on various bird species. The project centered on the design of a research pipeline, which involved pre-processing the text files into a BoW output, and feeding this result to three LDA implementations: two that have been developed using the *scikit-learn* [5] and *gensim* [7] libraries (addressed in section 3.1, and the other being our own approach, based on Chapter 27 of [4]. Notably, while the *gensim* and *scikit-learn* libraries use online variational Bayes, our implementation was constructed using the Gibbs sampling algorithm, which will be described in section 3.1.1. Our implementation was run only on a subset of 100 uniformly-selected examples from the data set. This sample was performed every 107 texts, which were alphabetically ordered by the name of the bird species, to avoid sampling bias. The need for sub-sampling was due to the current state of our implemented algorithm having a long run-time. Additionally, we only run this implementation for 100 Gibbs iterations, for the same reason.

2 Dataset & Pipeline

The data set used in this project was composed of 10,769 text files (.txt format), each containing the text of a Wikipedia article pertaining to a particular species of bird. Using an LDA topic modelling analysis of this data, the aim of this project was to cluster bird species based upon their significance in human culture, as found on Wikipedia. An example of such clustering by human significance could be species with positive perceptions in culture, such as robins, eagles and parrots, and negative cultural perceptions, such as vultures, crows and pigeons.

To do this we must implement an LDA topic modelling algorithm, the pipeline of which involves three key steps: pre-processing the data to ensure accurate results, training the model using the processed text data, and clustering the species of birds according to the output, i.e. the topic distribution across various texts. For the purposes of this project we worked with the PYTHON programming language.

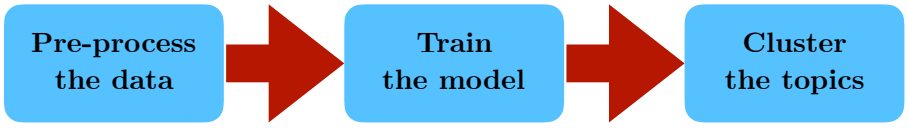

Figure 2: **Pipeline**. The pipeline of our LDA topic modelling implementation consists of 3 steps: pre-processing, training and clustering.

2.1 Pre-processing

The success of topic modelling using LDA is heavily dependent upon the quality and clarity of the data input into the algorithm. For better results, the text data from the Wikipedia articles must be pre-processed. The text cleaning processes implemented were as follows, using PYTHON's *Natural Language Tool Kit* (NLTK) [2]:

- Tokenisation
- Conversion to lower case
- Removal of common words
- Removal of numeric characters
- Removal of symbols and abbreviations
- Lemmatisation

The first step of text pre-processing was to separate the text data (initially in a single string as extracted from the Wikipedia articles) into individual words, or 'tokens', to enable analysis of each word for text cleaning. Once the words were isolated, they were converted to lower case to allow for text matching to the NLTK library corpora. This allowed for the removal of 'stop-words', which are the common words in the English language such as 'a','the','it', etc.. Such words provide little, if any, insight for topic modelling, and will affect our results since they appear in high frequencies across all documents, and hence were removed from the text data input for the LDA algorithm. There were also numbers, symbols and abbreviations (such as 'mm' or name initials) that appeared in some of our articles, which provided little insight and were inconsistent across all articles. These too were removed, by filtering out any numeric characters and any tokens with a length of two characters or less. Finally, a lemmatiser was applied to each word, which identified the different, inflected forms of words and converted them into the basic dictionary form (for example, 'walked', 'walking', and 'walks' would all be lemmatised to 'walk'). This would ensure that such inflected words in use across different documents, which a human would instantly recognize as having the same meaning, were perceived as such by the algorithm, and hence all contribute to the clustering of the same word into the topics output.

3 Methods

Here, we present the different approaches that we followed to produce a semantic clustering of the dataset previously described. We start by explaining how LDA was implemented and which other methods were followed to create the clusters. Results and comparisons will be reported in Section 4.

Before any implementation, a choice regarding the number of topics had to be reached. To do that, we evaluated the topic coherence score within the *gensim* implementation. The coherence is a number between 0 and 1 that is used to evaluate topic models. The number of chosen topics needs to maximise the coherence score. We explored various numbers on subsets from the data set until we found that 5-7 topics yielded the highest coherence values (between 0.4 and 0.6). We therefore chose to run our various LDA algorithms using 5 topics, a number that is also ideal for avoiding long running times.

3.1 LDA Implementations

3.1.1 LDA Implementation Using Gibbs Sampling

An implementation of LDA using Gibbs sampling was attempted. The equations used in the implementation can be found in [4] and were previously presented in Section 1. In this algorithm, one first initialises the word-topic assignment matrix, the document-topic distribution and the word-topic distribution matrices, by randomly sampling from the distributions described in Equations (1-4).

Gibbs sampling is a Markov Chain Monte Carlo (MCMC) algorithm used for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. Gibbs sampling is used as a means of Bayesian inference, to run the LDA generative process in reverse and thus infer the underlying topics given the words in a particular document. This algorithm works by sampling each of the variables given the other variables (i.e. the full conditional distribution).

In the case of LDA, the full conditionals are also given in [4] and are as follows *(Note: parameters have been previously described in Section 1)*:

$$p(q_{il} = k|\cdot) \propto \exp(\log(\pi_{ik}) + \log(b_{k,\pi_{il}})) \tag{7}$$

$$p(\pi_i|\cdot) = \texttt{Dir}(\{\alpha_k + \sum_l I(z_{il} = k)\}) \tag{8}$$

$$p(\mathbf{b}_k|\cdot) = \texttt{Dir}(\{\gamma_v + \sum_i \sum_l \mathbf{I}(x_{il} = v, z_{il} = k)\}) \tag{9}$$

Importantly, the number of iterations in the Gibbs sampling algorithm are crucial [6]. At least 1000 iterations of the above-described algorithm are needed to ensure convergence towards a *supposedly* accurate topic distribution for each individual document, and word distribution for each of the chosen topics. Due to this algorithm having a long run-time and not yet being optimised, we made the decision to only run Gibbs sampling for 100 iterations, for 5 topics, for the chosen subset of 100 documents.

3.1.2 LDA implementation by *gensim*

Gensim is a free library for topic modelling with Python. It is robust, efficient and easy to use. Their LDA package gives the option to choose the desired number of topics, the α parameter for the prior Dirichlet distribution of topics, the prior belief for word-distribution and the number of iterations before convergence, among others.

For our purposes, we set *random-state* = 1 for the initial point (for reproducing similar results), and set α to be asymmetric for diversity. This algorithm was applied to both the subset and the full dataset

searching for 5 and 10 topics respectively. No maximum number of iterations was set and the fact that the algorithm converged in a few seconds for the full dataset indicates the efficiency of *gensim*.

3.1.3 LDA Implementation by Scikit-learn

The importance of LDA is indicated by the ease of finding ready-to-use implementations, another of which is offered by *scikit-learn*. One is given the option to choose many parameters, such as the number of topics, priors for topics and the initial seed. Again, this was executed with five topics on the selected subset and on ten topics for the full data set. It should be noted that this implementation had a significantly longer run-time in comparison with the implementation by *gensim*.

3.2 Clustering

3.2.1 Black & White Clustering

LDA produces a probability distribution over the K topics that shows the association of a document with each topic. We define these distributions as

$$\mathbf{P}_i = [p_{i1},\ \ldots,\ p_{iK}],\ i = 1, ..., N_D.$$

for document i. In addition, LDA produces similar distributions for topic-word associations. Thus any document can now be represented as a (stochastic) vector in a N_V-dimensional space. Since $K \ll N_V$, this corresponds to some kind of dimensionality reduction. But how could we use this information for clustering?

One direct approach is to consider each topic as a whole cluster. Under this assumption, in order to partition the dataset, each document is associated only with the topic that corresponds to the highest probability. In other words, for document i we set as

$$\text{Cluster of } i = \arg\max_j \{p_{ij}\}.$$

This approach appears to work but is very naive, since it treats topics in a somewhat flat way. The clustering cannot be meaningful when a particular $\mathbf{P}_i$ does not contain a dominant element. Furthermore, what would happen in the case of a tie?

3.2.2 Clustering using Document-Topic Associations by LDA

To deal with the aforementioned problems, we attempted a more complex approach by using the complete set of document-topics associations as the feature space for clustering. Viewing the set of $\mathbf{P}_i$ lists as a $N_D \times K$ matrix could lead to a casual spectral clustering (or other) problem formulation. Perhaps a simple reordering of lines could unlock the existence of clusters that would correspond to a combination of topics, instead of a single one.

Such clustering was implemented, using the LDA output that gives the likelihood of each document being associated with each topic as the input for a clustering algorithm. K-Means is a robust clustering method, and is well-suited to this task. It is easily used by `Scikit-learn`, the state-of-the-art package for statistical learning with Python. K-Means starts by randomly selecting K subjects as centroids of the clusters and assigning every other element to the same cluster as the closest centroid. The centroid of each cluster is then updated iteratively, and the remaining points re-assigned until the objective function cannot get any lower.

The first step towards clustering our bird species documents by topic (as established by LDA) was to assess the similarity between the topic assignments for each document. Several similarity metrics were investigated, such as the Euclidean distance, but ultimately the cosine similarity between each document was used (normalised dot product), as it yielded the optimum clustering of our data for interpretation. As

stated before, some dimensionality reduction was required before the data could be parsed to the K-means clustering algorithm. Multi-Dimensional Scaling (MDS) was used to reduce the dimensions of our data to two, allowing this to be used as the clustering input. An example of the clusters achieved using this method is shown in figure 3.

3.2.3 Benchmark Technique: K-means

Evaluation in unsupervised learning is often difficult as there is no ground-truth knowledge. Our case is even more complex as we have no prior knowledge for the existence of clusters within this set of documents. In order to obtain a point of reference for evaluating our results, we began with a standard clustering method, namely a *K-means* approach.

K-Means can by applied directly to the TF-IDF matrix already calculated. The only parameter of high importance is the number of clusters, which, for reasons of comparison, is set to K, the number of topics by LDA. K-Means is a computationally fast algorithm and is, as we will demonstrate in the next section, efficient for accurate clustering.

Moreover, we used K-Means in a more complex way. As the TF-IDF matrix is large but sparse, we first attempted dimensionality reduction with Singular Value Decomposition. To this direction, we applied SVD with TF-IDF, keeping only the 50 (1250 for the full dataset) largest singular values to transform the data, and ultimately input that to the K-Means algorithm.

4 Results and Discussion

4.1 Comparison of different LDA implementations

LDA, like every probabilistic model, does not have one specific output. The results depend on the selection of the initial parameters, the success of convergence and, of course, the intermediate probabilistic values. As a result, there is no clear way to measure the precision and accuracy of the algorithm. However, in order to estimate the effectiveness of our implementation, we will compare the resulting document-topic distributions with those obtained by *scikit-learn* and *gensim*.

One usual way to compare two probability distributions is the *Kullback - Leibler* divergence. Assume that $P(\cdot)$ and $Q(\cdot)$ are two distributions defined on the same space $\mathcal{D}$ (for our purposes, the K topics), this metric is defined as

$$D_{\mathrm{KL}}(P\|Q) = -\sum_{x\in\mathcal{D}} P(x)\log\left(\frac{Q(x)}{P(x)}\right).$$

Larger numbers of this metric indicate greater deviations whereas smaller numbers show similarity; in the case of identical distributions, the KL divergence is equal to zero, which is the lowest possible value.

In our case, we obtain a set of N_D distributions, instead of a single one. Thus, by taking the average D_{KL} over the set of documents we can calculate an estimate of the dissociation between two LDA implementations. For a more rational comparison, we calculated D_{KL} among the following approaches: two runs of our implementation for 100 iterations, two by gensim and two by scikit, after selecting different initial points. For the cases of gensim and scikit, which are both computationally fast, we obtained the distributions after convergence, having run on the subset of 100 documents, searching for 5 topics.

The results of this comparison are summarised in Table 1. Each row compares the average KL divergence of the method mentioned on the left with each other approach. The two instances of our implementation seem to behave similarly, as those values are the lowest; this holds for gensim as well. Interestingly, the two approaches by scikit seem to differ greatly and also deviate from the other four cases. It should, therefore, be noted that we observe substantially low numbers and since this statistic occurs as an average of 100 values, it could be susceptible to outliers. Once again, this is just an attempt for comparison.

Table 1: **KL Divergence of different LDAs**

	ourLDA-1	*ourLDA-2*	*gensimLDA-1*	*gensimLDA-2*	*skLDA-1*	*skLDA-2*
ourLDA-1	0.0	0.596	0.567	0.612	3.891	3.755
ourLDA-2	0.494	0.0	0.744	0.806	3.974	3.962
gensimLDA-1	0.655	0.898	0.0	0.545	3.852	4.351
gensimLDA-2	0.72	1.024	0.519	0.0	4.175	4.303
skLDA-1	1.766	1.973	1.73	1.9	0.0	4.941
skLDA-2	1.596	1.825	2.046	2.004	4.971	0.0

Calculated on the subset with 100 documents, for 5 topics. The two approaches for the cases of gensim and scikit correspond to different seeds. Smaller values indicate stronger similarities. Thus, diagonal values are zero.

4.2 Evaluation of LDA Ability for Clustering

Figure 3 presents the output of the K-Means clustering algorithm for our implementation of LDA using Gibbs sampling. The results are encouraging, as one can clearly observe an approximately equal grouping of texts into 5 clusters, which correspond to 5 underlying groups (Topic 0 - Topic 4). Notably, these groups do not correspond to the 5 topics chosen for the document-topics matrix, but to an underlying *topic/theme* which may or may not be interpretable from the perspective of human understanding.

Ideally, we would have hoped to discover hidden patterns linking the various Wikipedia articles, and hence bird species, such as groupings by prey versus predator or by relevance to socio-cultural contexts. However, drawing conclusions in this case is a challenging – *if not impossible* – task, particularly due to one key limitation of the algorithm: the fact that it ran for only 100 iterations and therefore the document-topic and topic-word matrices may have not converged to stable values. Finally, while we can overcome this limitation given more time, this still does not guarantee that the clusters will have any relevance to human understanding.

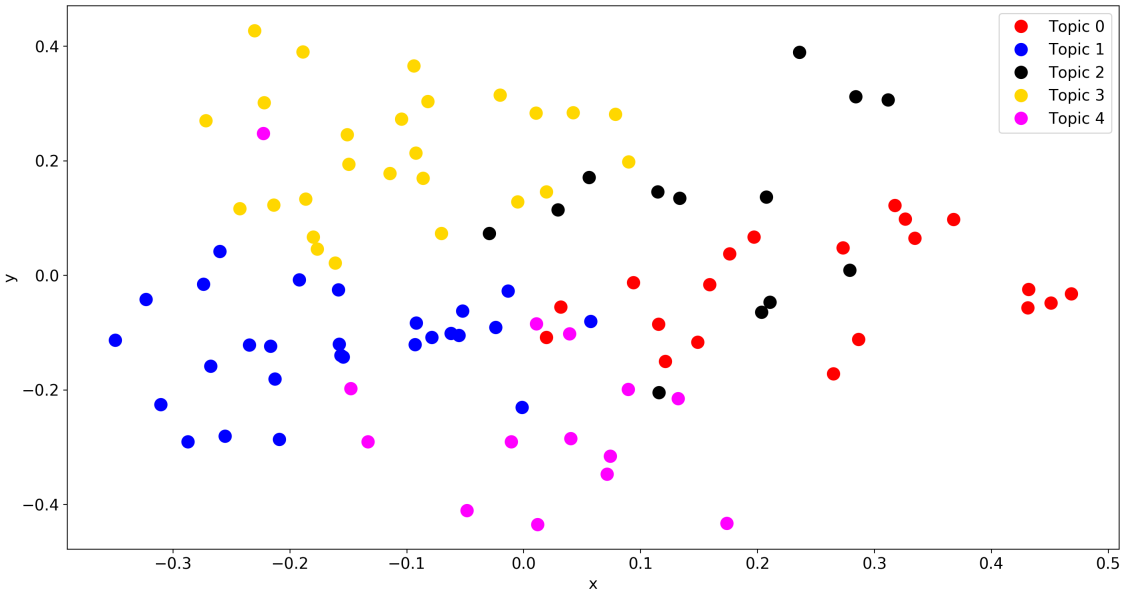

Figure 3: **Document-Topic Clusters**. Topic clustering with K-Means using cosine similarity for 100 uniformly-sampled Wikipedia articles

In order to compare the different approaches for clustering, we conducted an evaluation using the metrics defined in equation 6. In total, nine techniques were attempted: the three implementations of LDA were

used directly for clustering (*BW* cases) or as features for K-Means. Moreover, as described in Section 3.2.3, K-Means was used directly (standalone) on the TF-IDF matrix, or after dimensionality reduction with SVD. This yields a total of eight techniques. For clarity, and to give an idea of irrelevant results, we also evaluated the clustering that occurs by randomly assigning each subject to a cluster, since, this technique should yield the worse performance.

All of the nine methods mentioned above were evaluated on the subset of 100 documents, however, due to lack of resources and high complexity, the full dataset was tested only with the implementations by gensim and scikit. Table 2 presents all these measurements, for both datasets (*100 docs* and *Full set*) and the two metrics.

In regards to the small dataset, we observe many variations. Our approach seems to achieve an intermediate performance between *gensim* and *scikit-learn*. Notably, *gensim* and *scikit-learn* outputs fail to improve/converge when clustered with K-Means, irrespective of the number of iterations. When considering the full data set, the results can be interpreted from a different perspective. For instance, it looks like *gensim* failed to produce meaningful clustering as the distances within the clusters are similar to those between different clusters. One key observation is the accuracy of K-Means. Either applied directly on the TF-IDF data, or after a compression of the feature space, K-Means achieved by far the best performance, in every scenario.

Table 2: **Evaluation of Different Clustering Approaches**

method	Within-Clusters		Inter-cluster	
	100 docs	Full set	100 docs	Full set
LDA - BW	5.018	-	5.691	-
LDA + KMeans	5.848	-	6.304	-
gensim LDA - BW	4.882	68.225	6.838	69.754
gensim LDA + KMeans	4.882	68.227	5.061	69.757
Scikit LDA - BW	6.444	73.318	6.020	77.092
Scikit LDA + KMeans	6.444	56.112	6.692	59.009
KMeans	**1.3078**	**22.877**	9.162	148.294
SVD + KMeans	1.309	43.994	**13.452**	**209.921**
Random	6.283	56.147	7.337	56.200

The within-clusters and inter-cluster deviations are used as defined in Eq. 6. Successful clustering should yield low values for the first metric and larger for the latter. Starting from the top, the first two approaches are those implemented by us, and were not possible to run on the full dataset. Bold indicates the best approach for each metric and on each dataset. Every approach produced 5 topics for the small dataset and 10 for the large.

5 Conclusions

Creating semantics for text documents for clustering is an interesting and challenging problem. We examined the use of LDA for the extraction of topics for texts seen as bags-of-words, and their partition to clusters. We implemented LDA and used two, freely available implementations to find topics hidden in the corpus and partition the documents accordingly. The lack of ground-truth knowledge and the non-deterministic nature of the algorithm made it difficult to examine the performance and efficacy of the partition.

Overall, our approach produced results that were in agreement with other, state-of-the-art solutions. The two metrics used for evaluating clusters, however, indicate that topics alone fail to produce good clusters. A reason for this may be that documents may share multiple topics and thus an accurate partition cannot be achieved. We did not, however, acquire sufficient evidence to reject the hypothesis that latent semantic analysis can be used for effective clustering on articles about bird species. With more time we would have liked to pursue improved pre-processing, more suitable evaluation metrics and optimal selection of parameters.

References

[1] Charu C Aggarwal and ChengXiang Zhai. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer, 2012.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Adrian E Raftery and Steven Lewis. How many iterations in the gibbs sampler? Technical report, WASHINGTON UNIV SEATTLE DEPT OF STATISTICS, 1991.

[7] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.