University of California

Los Angeles

Application of Latent Dirichlet Allocation in Online Content Generation

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

Yajia Yang

2015

Abstract of the Thesis

Application of Latent Dirichlet Allocation in Online Content Generation

by

Yajia Yang

Master of Science in Statistics

University of California, Los Angeles, 2015

Professor Yingnian Wu, Chair

In this paper, I apply latent dirichlet allocation(LDA) to cluster 100,000 health related articles using the livestrong.com data set. I rst review the previous research progress in topic modeling. Then I introduce how LDA model is constructed. In stead of using simple word counts as model inputs, Part-of-Speech(POS) tagging and Term-Frequency Inverse Document Frequency(tf-idf) transformation are performed in data preprocessing steps in order to improve training e ciency and model interpretability. I further discuss the choices of model parameters, evaluating of model performance and visualization of model outputs from a real world point of view. Finally, I discuss two variations of conventional LDA including paralleled LDA and Online LDA. In addition to a traditional perplexity measure, I discuss how to use cosine similarity and Symmetric Kullback-Leibler Divergence to evaluate clustering performance. Three examples of using LDA outputs as building blocks for more complicated machine learning system are also demonstrated: 1) Cascaded LDA for taxonomy building. 2) In-cluster similarity computing. 3) Auto categorization.

The thesis of Yajia Yang is approved.

Hongquan Xu

Qing Zhou

Yingnian Wu, Committee Chair

University of California, Los Angeles

2015

To my parents . . .

who have been with me every step of the way

through good times and bad

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction to Topic Modeling

1.1 Topic Modeling History

It has been a consistent challenge for human to read and study all the digitized data in various forms and in overwhelming volume. In the big data era, data comes as number, text, image, video or even sound. Therefore, we need a powerful tool to help organize analyze the data. It is necessary to find an algorithm that aims to cluster and annotate large archives of different forms of data with thematic information.

Word count is probably the simplest but most widely used method of summarizing large corpus. However, it gives no thematic information. "tf-idf" is so-far the most well-known scheme in information retrieval. It convert count vectors to a normalized real number that takes word rarity into account. However, in order to achieve a more significant dimensionality reduction in length and also reveal the inter- and intra- documents statistical structure, researchers has proposed other methods, of which Latent Sematic Indexing (LSI)[DDL90] is the most notable. Similar to principal component analysis, LSI uses a singular value decomposition of the tf-idf matrix in order to project the original space to a linear subspace that captures most of the variations. A significant step forward in the field of natural language processing is probabilistic LSI introduced by Hofmann[Hof99]. This approach models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as representation of topics.

However, this approach doesn't describe any probabilistic model at document level and thus leads to two potential issues. First, the model itself tends to overfit as the parameter

size will grow linearly with corpus size. Second, it performs badly for out-of-sample test dataset since the allocation of probabilities to the document outside the training set is not well defined.

As a result of evolution in information retrieval research, Latent Dirichlet Allocation(LDA) is trying to consider mixture model that captures exchangeability of words and documents that previous researches lack of capability to handle. In addition to the basic LDA idea itself, it is also worth noting the evolution of inference procedure of LDA that has been the driving-force behind the popularity of LDA. The first LDA paper[BNJ03] adopted variational method and EM algorithm for parameter estimation. Recently, researchers had proposed more efficient algorithms that significantly reduce estimation time [TNW06][PNI08].Most recently, researchers have proposed a new inference method [MHB12] for LDA that takes advantages of both sparse Gibbs sampling and online stochastic inference.

1.2 Latent Dirichlet Allocation

The basic idea of LDA[BNJ03] is that a document can be viewed as a mixture of a finite number of topics and a topic is a mixture of a finite number of words. Given corpus, LDA model attempts to uncover the following:

- It discovers a set of topics.
- It allocates a set of words to a topic.
- For each document, it associates a specific mixture of topics.

The common vocabulary used in LDA is as follows:

- Word: the basic unit defined to be an item from a vocabulary of size W.
- Document: A sequence of N words denoted by $d = (w_1, w_2, ...w_N)$ where w_n is the N-th word in the sequence.

- Corpus: A collection of M documents denoted by $D = (d_1, d_2, ...d_M)$

LDA models each document d as a multinomial distribution θ over T topics and each topic, Z as a multinomial distribution ϕ over a set of words W. The LDA model assumes a given prior dirichelet distribution on θ, thus allowing the estimation of ϕ separately.

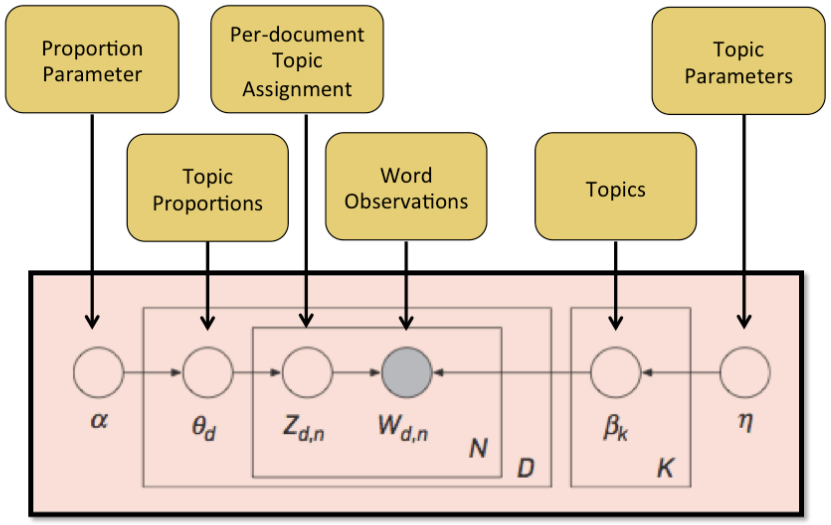
LDA is a generative process for creating a document as presented below:

- Select the number of words N: $N \sim Poisson(\lambda)$
- Select θ from a dirichlet distribution parameterized by $\vec{\alpha}$:$\theta \sim Dir_V(\vec{\alpha})$:
- For each word in the document,$w_n \in W$,do:
    - **a.** Draw a topic $z_n \sim multinomial(\theta)$
    - **b.** Draw a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability ϕ_{z_n}

Figure 1.1 shows the graphical representation of LDA model. α and η denotes the proportion parameter and topic parameter, respectively. These two hyperparameters define the sparsity of dirichelt distribution from where LDA will draw topic proportions(θ_d) and topics(β_k).Further, $z_{d,n}$ is used to denote per document topic assignment which determines word observations($w_{d,n}$). LDA is a generative process that attempts to mimic how humans generate articles(not necessarily articles but it makes more sense in a natural langurage setting). The task of LDA is to back calculate all hidden parameters given words observation. The bottom equation shows the joint distribution of both hidden and observed variables which is the fundamental distribution for constructing LDA inferences.

1.3 Inference Procedure

In LDA, the mixing coefficients for each document and the word-topic distribution are unobserved (namely hidden or latent) and are learned from data using unsupervised learning



$$P(\theta_d, Z_{d,n}, \beta_k, w_{d,n}) = \prod_{d=1}^{D} P(\theta_d) \prod_{k=1}^{K} P(\beta_k) (\prod_{n=1}^{N} P(Z_{d,n} \mid \theta_d) P(w_{d,n} \mid \theta_d, \beta_k))$$

Figure 1.1: The graphical model for Latent Dirichlet Allocation.

methods. Variational Bayes method is widely accepted as faster computationally but collapsed gibbs sampling approach is in principal more accurate. Most recently, a new inference algorithm for LDA called Collapsed Variational Baysian Inference that combines advantages of both was also proposed[TNW06]. Researchers also work on improving the gibbs sampling approach by introducing fast collapsed gibbs sampling[PNI08]. Here I just give a brief review of the advantages and disadvantages of each method.

1.3.1 Variational Bayes [BNJ03](2003)

Standard Variational Bayes(VB) inference upper bounds the negative log marginal likelihood using the variational free energy. Although efficient and easily implemented, VB can potentially lead to very inaccurate results.

1.3.2 Collapsed Variational Bayesian Inference[TNW06](2006)

Collapsed Variational Byesian(CVB) is a variational algorithm that models the dependence of the parameters in an exact fashion. However, since the latent variables are already weakly dependent on each other, one can assume the latent variables to be independent and still expect the resulting to be accurate. Figure 1.2[BNJ03] summarize the results of comparison between CVB, Collapsed Gibbs Sampling and VB. CVB converged faster and to significantly better solution than standard VB. CVB also converged faster than Collapsed Gibbs Sampling but is less exact.

1.3.3 Fast Collapsed Gibbs Sampling[PNI08](2008)

Building on top of Collapsed Gibbs sampling, this approach further reduces the time taken for the inner loop of the sampling process by ordering the sampling operation based on the fact that the sampling distribution of interest is frequently skewed. It is worth noting that this method is still exact, namely, no approximation is made and the fast algorithm correctly and exactly samples from the same true posterior distribution as the slower standard Gibbs

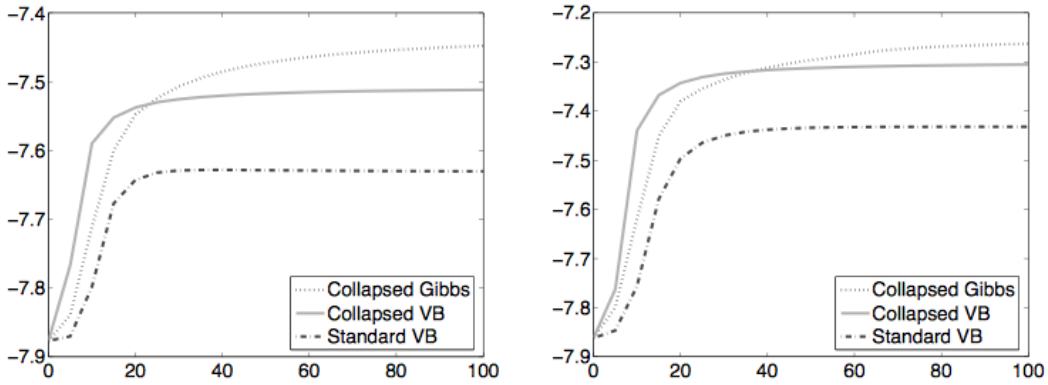


Figure 1.2: Per word log probability as a function of number of iteration

sampling algorithm. CPU time increase linearly with the number of topics K for LDA. The CPU time for FastLDA is significantly less than that for LDA. Further more, we observe that FastLDA CPU time increases slower than linearly with number of topics, indicating a great speedups with increasing number of topics.

1.4 Popular Applications of LDA

1.4.1 Document Classification

In a typical text classification setting, the goal is to classify a document into two or more classes where each class is ideally mutually exclusive. A critical part of document classification problem is the choice of features. Treating individual words as features generate a very large dataset[Joa99]. One way to reduce the feature set is to use an LDA model for dimensionality reduction. Actually, LDA reduces any document to a fixed set of real-value features - the posterior dirichlet parameters associated with documents. Figure 1.3[BNJ03] illustrates a comparison of performance of a binary classification problem between LDA features with all the word features. We can see that there is little reduction in classification performance in using the LDA-based features; in fact in almost all cases the performance is improved.

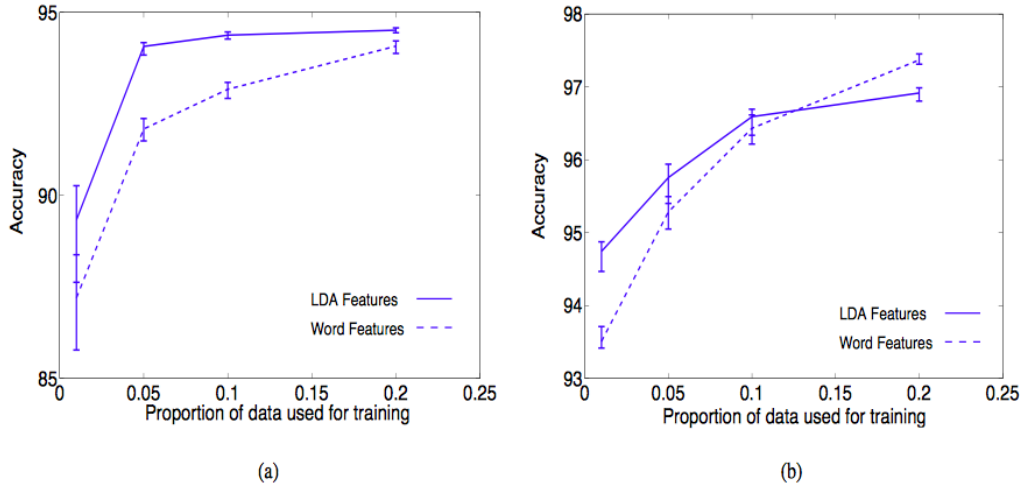


Figure 1.3: Using LDA as a dimension reduction method

1.4.2 Collaborative Filtering

In a collaborative filtering problem setting, one can train with fully observed documents. Then, for each unobserved documents, all but one of the words are shown and the question is to predict what the missing word is. In the LDA model, the probability of the missing word is computed by integrating over the posterior dirichlet distribution:

$$p(w|w_{obs}) = \int \sum p(w|z)p(z|\theta)p(\theta|w_{obs})d\theta \tag{1.1}$$

1.4.3 Computer Vision

Not only in the field of natural language processing, LDA can also be used in the field of computer vision[NWF08][PAN10][WG08]. We can apply LDA to computer vision, specifically for learning natural scene categories. One can model the image as a collection of local patches. Each patch is represented by a codeword from a large vocabulary of codewords. The goal of learning is to find a model that best represents the distribution of these patches in each category of scences.

CHAPTER 2

Latent Dirichlet Allocation for Online Content Generation

2.1 Introduction to Online Content Generation

Everyday, people access and consume various types of online contents including pictures, articles, videos and even voice. It is always a big headache for online publisher to "guess" what is interesting to audiences since their tastes change very rapidly. In early days where there were very few online publishers, company could make millions of dollars by simply keeping track of search keywords on search engines since all you need to do is make sure the title of your contents match what people search exactly. The search results thus became so skewed that certain online publishers will almost always take the top 3 positions among all search results. In 2011, search engine providers started adjusting their ranking algorithm by penalizing low quality content and domain authority. As a result, only the high quality contents from well-established website will get a favorable position in people's search results. Online publisher then started to rethink their business model. Although it is hard to find a common ground for every player in the online content business, many have understood that to decide what kind of content to publish, one needs to take many factors into account: search volume, social popularity, domain authority, competitiveness, ranking difficulty an so on. It is worth noting that it is almost impossible to generate contents in a topic that is unfamiliar to you and get a high rank in search engine since it will take historical specialization into consideration as well.That being said, it is critical to leverage whatever contents online publishers already have in their database and reorganize them into highly related content

group such that when people are searching related keywords, their contents have high chance to show up at a favorable position. The needs of reorganizing online contents fit naturally what LDA provides and thus I apply LDA to cluster and categorize online contents from livestrong.com.

2.2 Data Preprocessing

2.2.1 Introduction to livestrong.com Dataset

Livestrong.com is a content farm that connects individual freelance writers with a wide range of online audiences who are interested in health and weight management tips. The whole site provides over 100,000 health-related articles covering topics including fitness, food, weight loss et al. The average length of each articles is approximately 300 words. The original file is 500Mb in size and contains a lot of numbers, special signs and non ASCII characters which needs to be cleared up before feeding into the model.This step is primarily performed using python regular expressions.

Figure 2.1 shows a manual labeled distribution of all livestrong.com articles. From the figure we know livestrong.com has a pretty skewed coverage on health related topics, for example, 20% of articles are under food and drink bucket.This is actually the current taxonomy used within livestrong.com but there are numerous mis-categorized articles and that is why LDA came into play.

2.2.2 Noun Extraction

The document in itself is usually an unstructured text collection so the first step of processing any text-based dataset is to convert it to a matrix representation. Bag-of-words method is currently a standard in natural language processing that discard the meaning of order in the document and only consider the word itself. Recent research[DBJ12] has proposed some methods that go beyond bag-of-words and take sequences of chapter/paragraph into

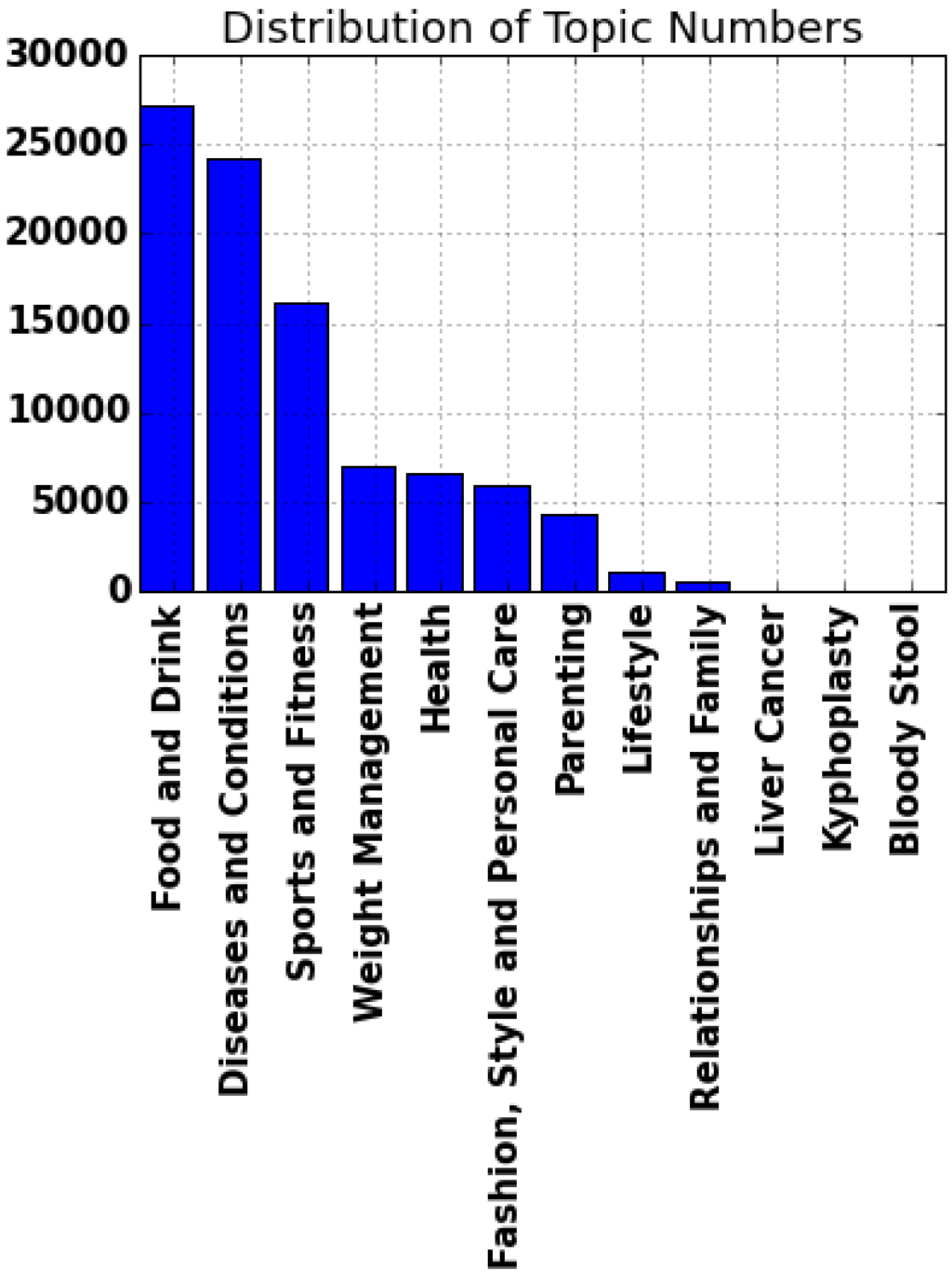


Figure 2.1: Topic Distribution of Livestrong.com Articles

consideration. However, the discussion of using methods other than bag-of-words has beyond the scope of this paper, thus I will only use bag-of-words for any feature extraction throughout this paper. The very first step of data preprocessing is to decide whether all the words of the corpus should be used or just part of it. A common problem of using all the words in the corpus is that it will introduce too much noise (article, adjective, preposition, pronoun, adverb, conjunction, and interjection) that contains very little information in terms of understanding the topics. For example, by using bag of words method, the whole corpus has over 700,000 dimensions. Thus for interpretability and training efficiency purpose, I just extracted all nouns from the original corpus which significantly reduce dimensionality of training set while retains most of the important information. In this step, I use python textblob (`https://textblob.readthedocs.org/en/dev/`) package which generate a blob object and perform Part-of-speech tagging task in a multiprocessing manner.This is not a mandatory step for LDA but according to my experience this step significantly increase the interpretability of final model outputs. Also, depending on the size of corpus and the training time budget, we can keep verbs in the training set as it contains some information as well. Noun extraction step is relatively time-consuming, for my case, this step takes roughly 120 seconds. However, it saves training time later on as the dimensions have shrink to half of original size.

By using noun extraction we are actually performing dimensionality reduction based on part-of-speech tagging, obviously we lose some information in order to speed up the training process. There is no clear cut on what part-of-speech should be used but according to my experience, if there are decent amount of nouns(300-500/document), then the extraction is a pretty good approximation to original document.

An alternative to noun extraction is noun phrase extraction[BC00]. A typical example is the handling of noun phrase "New York". In a noun extraction setting, the algorithm will break out the phrase and annotate them as New[adj] and York[noun] which makes very little sense whereas in a noun phrase extraction setting, the algorithm will definitely mark the two as a whole as a noun phrase.Theoretically noun phrase extraction should deliver better

performance, in this specific problem, I found that noun phrase extractions(using the same python textblob package) tend to group very long word list as a phrase, which will make it hard to interpret when it comes to LDA output. Another practical reason that I didn't use noun phrase extraction is special noun phrase(like 'New York') didn't appear quite often so their are actually not significant improvement on model performance.

2.2.3 Term-frequency Inverse Document Frequency (Tf-idf) Weighting

After noun extraction, a word count function is performed on the corpus to obtain a term-document matrix. Term-document matrix is a good representation of original dataset. However, there are two potential issues if the model takes term-document matrix directly as input: 1) cross-corpus common words will get more weights than it actually deserves. 2) document specific words may receive less weights than it deserves. To avoid these, Tf-idf weight is introduced. It is simply the product of its tf weight and idf weight. Tf weight is defined as the number of word occurrence in a document, sometime we use log normal transformation for smoothing purpose. Idf is inverse document frequency, defined as the log of number of documents over number of documents that contain the word. Tf-idf is the best known weighting scheme in information retrieval. The tf-idf weight increases with the occurrences within a document and the rarity of the term in the collection. It is worth noting that some researchers[Pai13] has further advanced the use of tf-idf by introducing more information to capture term saliency. Again, there might be various type of modifications on top of tf-idf score but for demonstration purpose, I think tf-idf itself is good enough.

This step is important when the corpus are, to some extend, belongs to a common topic. In this case, we know that all livestrong.com articles should be relevant to health. If I simply use the word count as model inputs, words like health and weight will appear in almost all topic representations. This is bad since what we want is some easily distinguishable topics.

$$w_{t,d} = (tf_{t,d}) \times (\log_{10} N/df_t) \tag{2.1}$$

The table 2.1 is an example of converting term-document count matrices to tf-idf matrices. Assume there are 7 documents and each contains some words. By converting term-frequency counts to tf-idf, we obtain a new representation of original corpus. Basically, if a word does not appear in a document, both word count and tf-idf will be zero. If a word is dominant in a document, its weight will be panelized. It is worth noting that there is a potential problem using tf-idf matrix as traing input. If the purpose of building the model is to automatically label future articles, then theoretically we need to recompute the whole tf-idf matrix for every new article because tf-idf is a global dependent measure such that any addition to the original dataset will change the tf-idf score entirely(the idf score part). This can be a very big headache if you have very large training corpus since the computing of tf-idf may take minutes. Fortunately, in most cases, you don't need to label new articles that have comparable size to your original corpus. The result is still reliable if the new ones is not totally out-of-sample. From my experience on livestrong.com corpus, if the amount of new articles are smaller than 5% of original corpus, I can expect a pretty good labeling performance.

With that in mind, just using word counts might be a better idea if the corpus is very dynamic and incoming articles might vary a lot in topics.

2.2.4 Sparse Matrix Representation

Next, all the tf-idf score will be stored in a sparse matrix (115,133 rows x 94,851 columns) of which 6,752,514 elements having non-zero values. Only 0.06% of the matrix are non-zeros.As a comparison, if all the words are used to construct the tf-idf matrices, we will obtain a sparse matrix(115,133 rows x 153,806 columns) of which 23,574,292 elements has non-zero values, aka a sparsity of 0.13%. Using sparse matrix instead of original tf-idf matrix will significantly speedup matrix manipulation. This will work only for sparse matrix and will only do little if the matrix is dense. Fortunately, most of the high dimensional problem is under a sparse situation.

Table 2.1: Converting term-document matrices to tf-idf matrices

Documents	weight	food	vitamin	skin	baby	diet
#1	150	73	0	0	0	0
#2	4	57	0	1	0	0
#3	232	227	0	2	1	1
#4	0	10	0	0	0	0
#5	57	0	0	0	0	0
#6	2	0	3	5	5	1
#7	2	0	1	1	1	0

Documents	weight	food	vitamin	skin	baby	diet
#1	5.25	3.18	0	0	0	0
#2	1.21	6.1	0	1	0	0
#3	8.59	2.54	0	1.51	0.25	0.35
#4	0	1.54	0	0	0	0
#5	2.85	0	0	0	0	0
#6	1.51	0	1.9	0.12	5.25	0.88
#7	1.37	0	0.11	4.15	0.25	0

2.3 Training and Evaluating LDA model

In this paper, I used an open source python package gensim to train LDA model.The training process was based on multiprocessing framework in order to speed up the E-M inference process.

2.3.1 perplexity

In convention of language modeling[BNJ03], perplexity on hold-out test set is usually used to measure the generalization performance of predicting models. The perplexity is monotonically decreasing in the likelihood of test set and has equivalent algebraic meaning as the inverse of the geometric mean per-word likelihood. For a test set with D documents, the perplexity is defined as:

$$perplexity(D_{test}) = e^{-\frac{\sum_{d=1}^{M} logP(w_d)}{\sum_{d=1}^{M} N_d}} \tag{2.2}$$

Perplexity is very useful for comparing different topic models in that the lower the score the better the performance. However, if one have decided to use LDA but want to compare the performance of LDA with different parameters,for example the number of topics to use, perplexity will not always give a deterministic conclusion since we know that more topics will almost always give lower perplexity score. Researchers[AGR03] have shown that language model perplexity actually retains a systematic relationship with the achievable precision recall performance. In this problem, whether the clustering result makes sense to human or not is more important than the score itself, so I will not focus on comparing perplexity score in this paper.

2.3.2 The Choice of Dirichelet Priors

As denoted in figure 1.1, there are two hyerparameters α and η that affect sparsity of the document-topic(θ) and topic-word(β) distributions. Both default to a symmetric array with

all elements equal to 1.However, both of them can be set to a prior of your choice.The larger the value the less sparse the document-topic and topic-word distributions are. The array is unnecessarily symmetric, and can be even been a learned prior directly from the data. we can also assign a predefined vector to both parameters if we have any pre-knowledge of the actually distribution. Recent research [WMM09] has shown that some asymmetric dirichlet priors on document-topic distribution can achieve better performance and robustness. There is currently no gold standard about how to choose the hyperparameters, however, by using default vectors, I have already get pretty reasonable results.

2.3.3 The Choice of Number of Topics

The choice of number of topics is more of art than science as well. It depends on how large your corpus is and what is the purpose of using LDA model. Hold-out perplexity can be used as a reference but since the hold-out process is random and the score can be really close in a real situation, I would not use it as a single best way to determine an optimal number. It is important to note that the time complexity of LDA will increase as with number of topics and thus can be a factor to be consider when choosing the number of topics.

In this specific application, the purpose is to cluster the corpus and it needs to make sense to humans so I vary the number of topics from 2 to 20 and finally decide to use 8 topics for further analysis based on how separated any two topic clusters are(I will further discuss this on later chapters). Actually if topic number becomes larger than 20, many clusters will contain topic words that make little sense. Depending on the size of corpus, this number can grow accordingly. Mathematically, there is no limit on the number I choose.

2.3.4 Interpretation of Model Outputs

The output of LDA are document-topic distribution and topic-word distribution. Unlike k-mean clustering, LDA outputs topic information and thus can be used to summarize the topics.

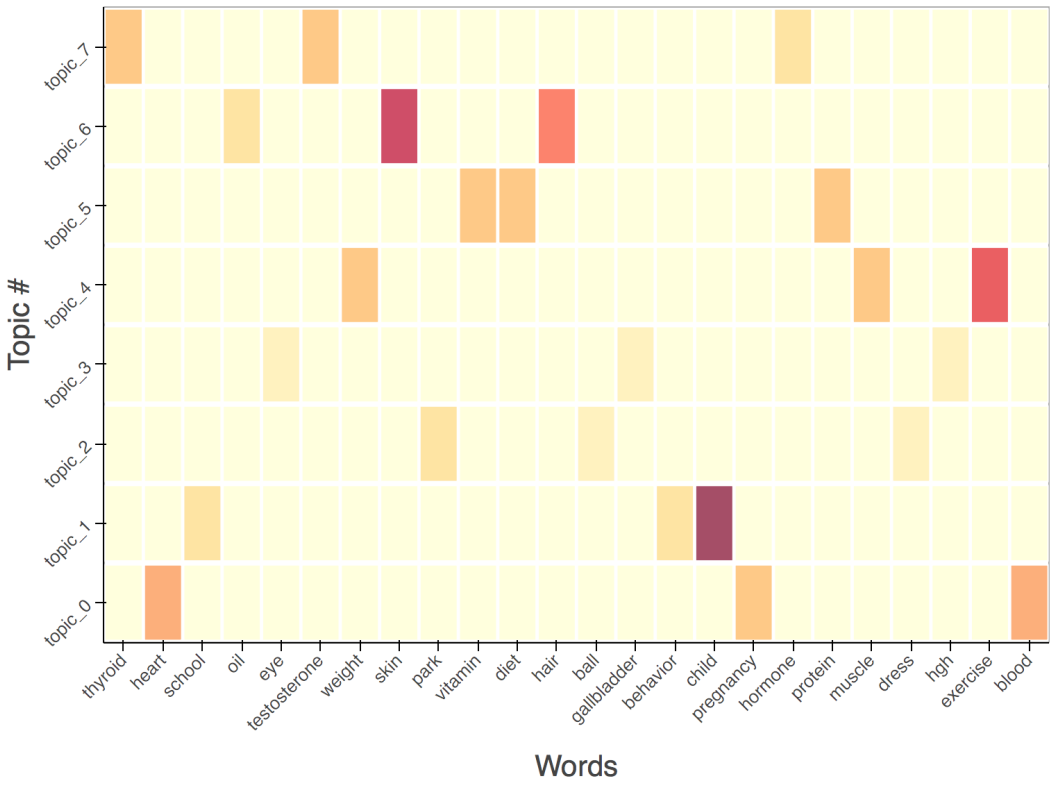


Figure 2.2: The 8-topic LDA model for 115,133 Livestrong.com articles

The figure 2.2 demonstrates a 8-topic LDA model heat-map for all 115,133 Livestrong.com articles. Each row represents a topic that consists of all the words with different probability (Dark yellow means high probability while light yellow means low probability). Each column represents the importance of a word across all topics. We can see that there are very few overlaps of words for different topics, meaning that the clustering algorithm is working well.For simplicity purpose I only choose the top 3 words of each topic but actually LDA assign a probability to each word. From the figure, for example, 'Child' is the most probable word under topic#2.

As mentioned before, performing part-of-speech tagging on original dataset is very useful in that the final topic representation will be much more interpretable.

The model can be used to label either training corpus or new corpus. For each document, the model will give a probability distribution over topics and the topic with the highest

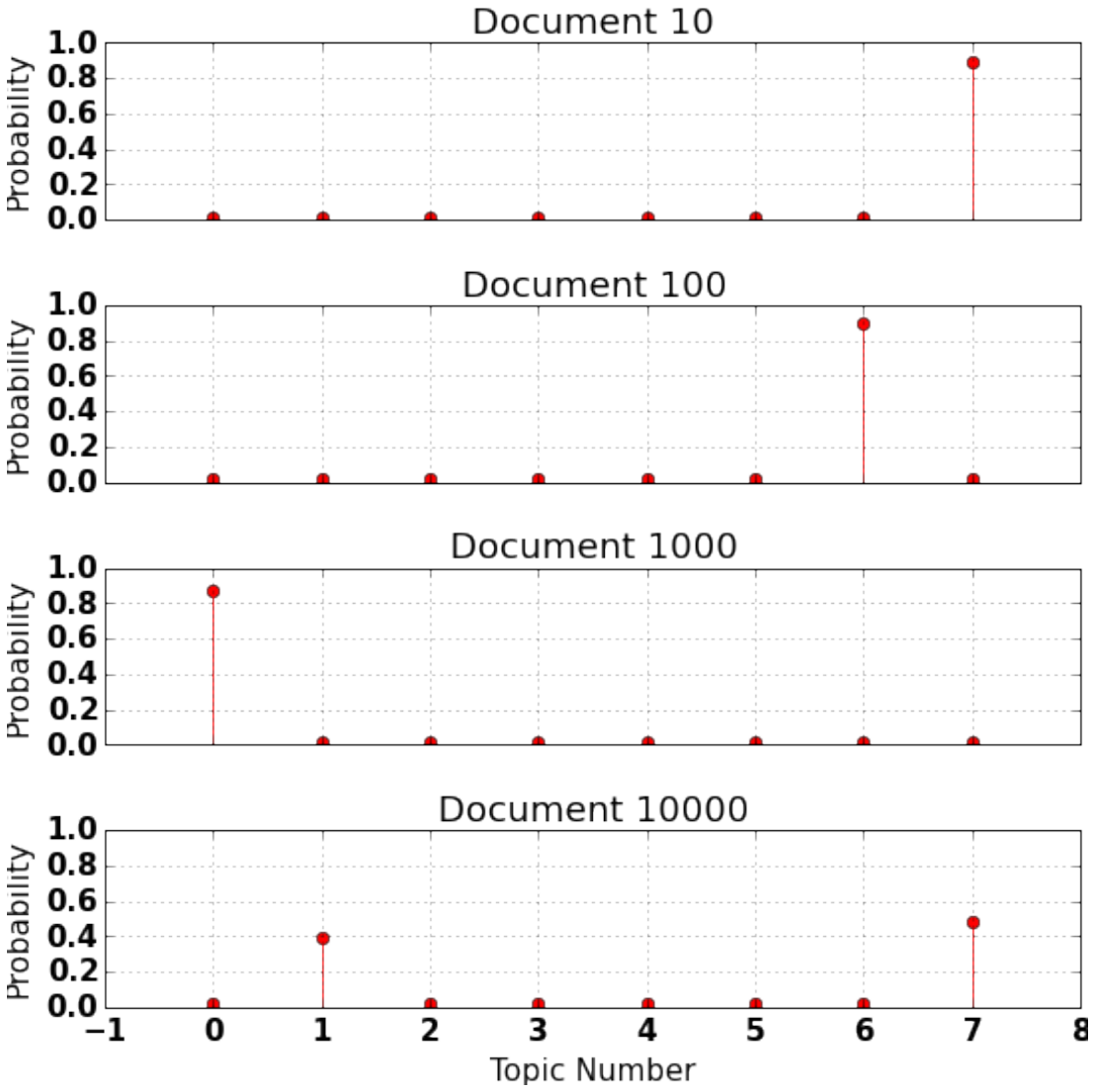


Figure 2.3: Topic Distribution of 4 arbitrary documents

Table 2.2: Topic Representation of 8-topic LDA

Food	**Weight**	**Skin**	**Parenting**	**Disease**	**Sports**	**Drug**	**Unknown**
diet	exercise	skin	child	heart	park	thyroid	gallbladder
vitamin	**weight**	hair	behavior	blood	dress	testosterone	eye
protein	muscle	oil	school	pregnancy	ball	hormone	hgh
blood	training	acne	baby	pain	lake	medication	vision
sugar	body	**water**	time	pressure	trail	side	bile
weight	workout	cause	family	cancer	game	gland	retina
body	leg	infection	toddler	cause	**water**	drug	bilirubin

probability is usually selected as a topic label for that document. As shown in figure 2.3, I randomly select 4 documents(the corpus has been randomly shuffled) from original corpus and use the LDA to evaluate the topic numbers. We can see that each document has been assigned a dominant number and it is clear which topic a document should belong to. This is good for most of the application but sometimes we may want a document to be more like a mixture of different topics. In this case, we can use $\vec{\alpha}$ to control the sparsity of topic distribution as aforementioned. By simply specifying a larger $\vec{\alpha}$, we will force LDA to favor more mixed topics.

Table 2.2 shows the top 7 words within each topic. The topic words(food,weight, skin, parenting, disease sports,drug and unknown) is manually labeled. It is worth noting that there is topic "Unknown" because the top words under this topic seems to be random combinations. Actually the total number of articles under "unknown" topic is just 66, accounting for less than 0.1% of the corpus, we might view them as random noise. One word can also appear in two topic representations. For example, the word weight are top words both in FOOD and WEIGHT topics.

New topic distribution, shown in figure 3.2, is plotted. The total number of articles under FOOD topic is roughly 45,000, meaning that the LDA model "think" some of previous human defined topics is inaccurate. The overall distribution has similar skewed shape as with the

original one.

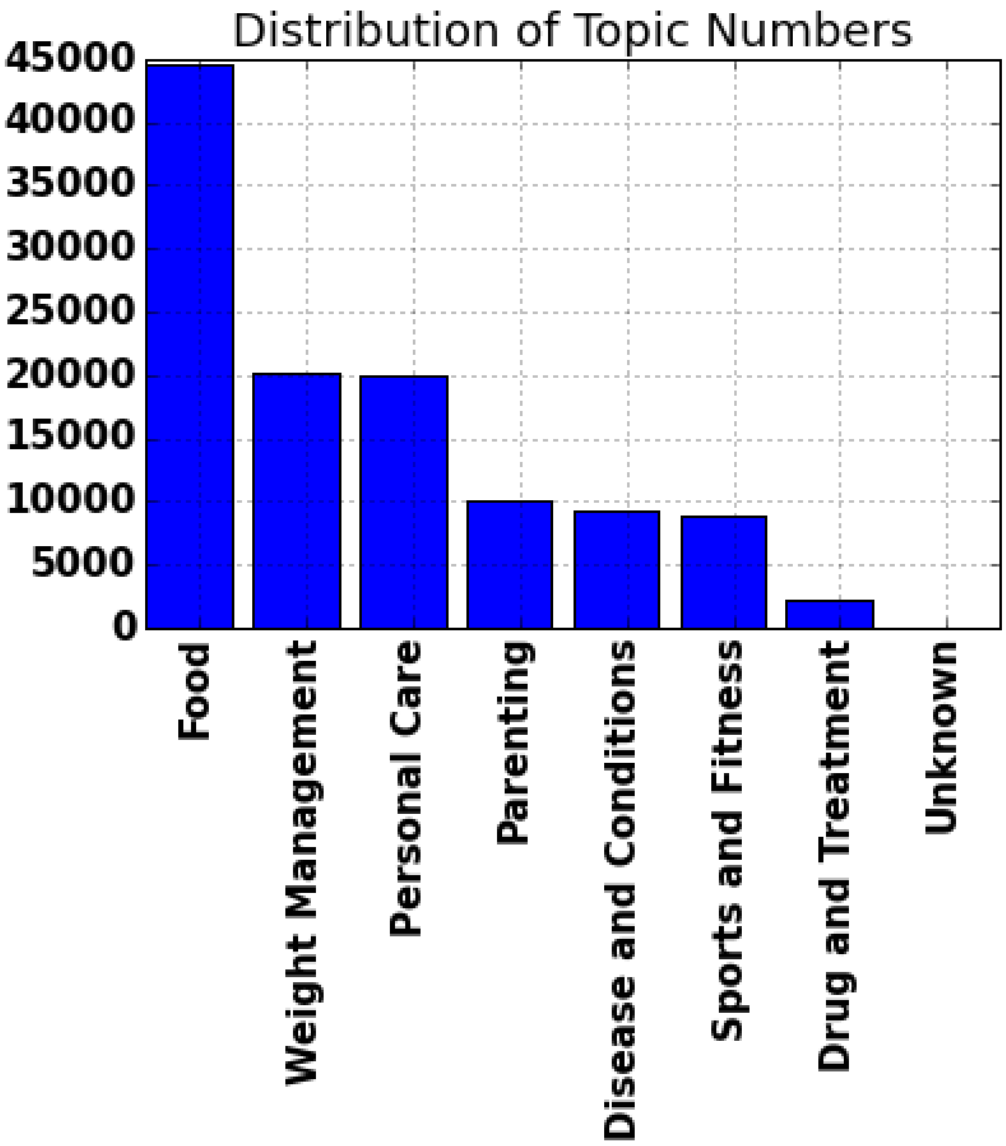


Figure 2.4: Topic Distribution after LDA

CHAPTER 3

Discussion

Till now I have demonstrated how the LDA model is applied to roughly 100,000 articles. In this chapter, I will talk about more technical details on how to speed up the training process and even work on datasets that is streaming.

3.1 Parallelized LDA

The online variational Bayes LDA training in gensim (`http://rare-technologies.com/multicore-lda-in-python-from-over-night-to-over-lunch/`) conceptually consists of Expectation-Maximization iterations, with alternating E-steps and Msteps. In Estep we perform inference on a chunk of documents. The sufficient statistics collected in E-step are accumulated and used to update the model in M-step. In terms of improving the performance of the algorithm, one can quite easily distribute the E-step to several workers and by this possibly speed up the whole LDA model training, as shown in figure 3.1.Researchers have proposed different strategies to solve the scalability problem that LDA faces, for example in [LZC11], data placement, pipeline processing,word bundling and priority-based scheduling are proposed as a solution. However, here we just focus on EM algorithm and use it as a datapoint to show how parallelized computing can speed up the training process.

To quantify the improvements of parallelized LDA, I plot training time against different number of workers, shown in figure 3.2. To train a 8-topic LDA model for 115,133 articles, the running time using 7 cores is roughly 160s while it takes roughly 500s under 1 core situation. The parallelized LDA is very useful especially when the train size is large.

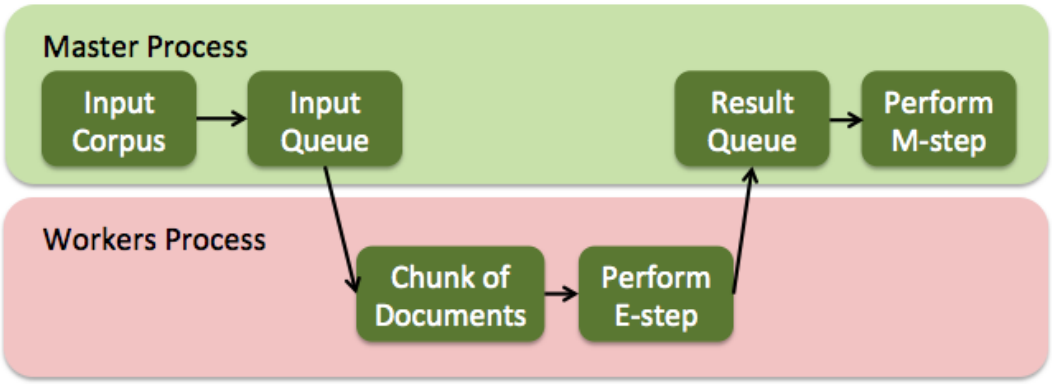


Figure 3.1: The conceptual flow of multicore LDA

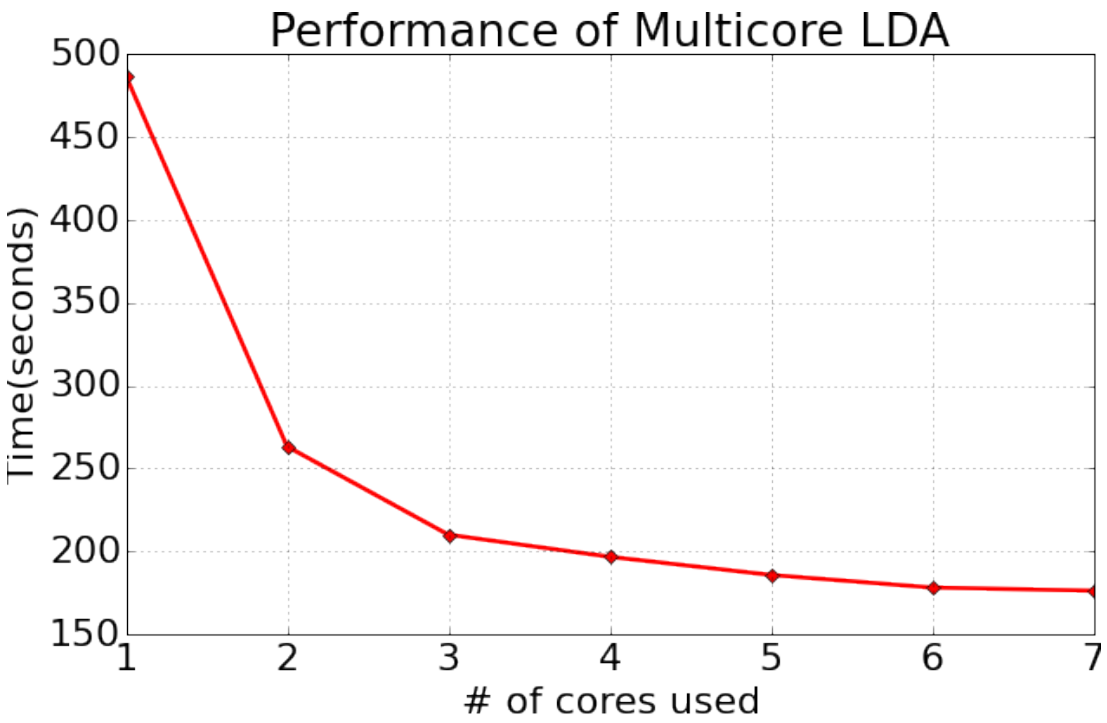


Figure 3.2: Training time v.s. number of workers

3.2 Online Updating

Another practical concern is that real world data can arrive in a stream but conventional batch variational bayes for LDA is not able to handle this efficiently. Recent research [HBB10] has proposed a new on-line LDA method where the parameters can get updated in a stream manner.

Online LDA is based on online stochastic optimization with a natural gradient step, which converges to a local optimum of the variational objective function. Researchers also show that online LDA finds topic models as good as or better than those found with batch variational bayes algorithm, and in a fraction of the time.

This method is extremely useful for livestrong.com as the freelance writers generate roughly 1, 000 new articles per day and all of them needs to be categorized into corresponding topics on a daily basis. As the model evolves with incoming articles there is no need to worry about the inaccuracy of taxonomy.

3.3 Topic Similarity

How to evaluate the performance of LDA is still under debate. Popular metrics include hold-out perplexity, log loss et al. Although these measures make sense mathematically, I would like to look at the problem in a more practical and intuitive way. The whole idea of clustering text is to minimize the cross-topic 'similarity' and maximize in-topic 'similarity'. Since the output of LDA is a list of words associated with probability, I applied two popular similarity calculation methods on topic distributions

3.3.1 Cosine Similarity

Cosine similarity (`https://en.wikipedia.org/wiki/Cosine_similarity`) is quite popular in quantify similarity between high dimensional vectors.It has the following formula:

$$similarity = cos(\theta) = \frac{AB}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{3.1}$$

A geometric explanation of cosine similarity is the angle between the two vectors in a high dimensional space. This measure is insensitive to the increase of dimension and is thus considered a good choice of similarity score. However, since LDA output also associate probability with words, I decide to use symmetric Kullback-Leibler divergence as the measure for topic similarity

3.3.2 Symmetric Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence (`https://en.wikipedia.org/wiki/KullbackLeibler_divergence`) is an asymmetric measure of the difference between two distribution P and Q with the following mathematical formula:

$$D_{KL-Asymmetric}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{3.2}$$

It is evident that KL divergence is an asymmetric measure but we want to treat different topics equally. A simple fix is to take average of two asymmetric KL divergence scores as follows:

$$D_{KL-Symmetric}(P\|Q) = \frac{1}{2}(\sum_i P(i) \log \frac{P(i)}{Q(i)} + \sum_i Q(i) \log \frac{Q(i)}{P(i)}) \tag{3.3}$$

From the formula, we know for KL divergence, the higher the divergence value the more separated the two topics. Here I run LDA algorithm 16 times but pass different number of topics each time and use the output topic representations to plot an array of confusion matrices,as shown in figure 3.3. Ideally, all, a topic should be similar only to itself, meaning

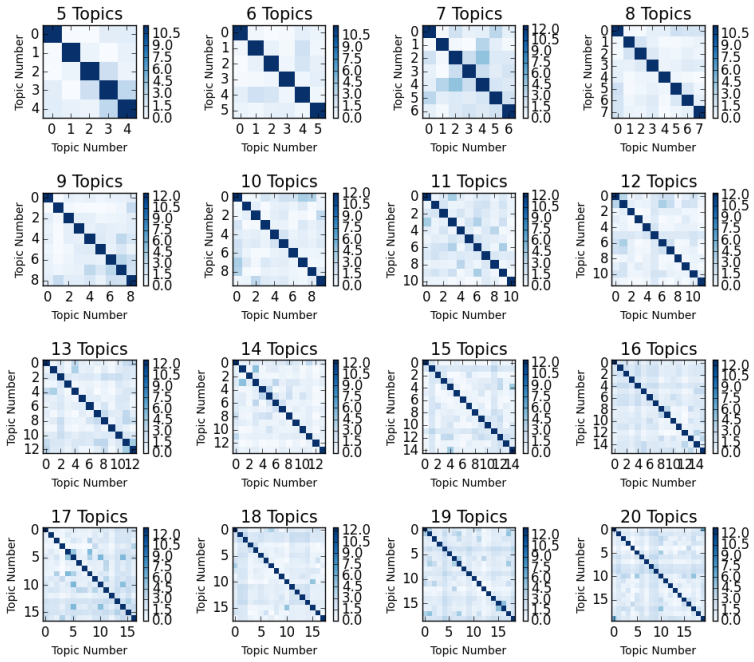


Figure 3.3: KL divergence score on 16 different topic models

that all the dark color should be in the diagonal position and by looking at this figure set I can have a better understanding of how good a LDA model performs. For example, LDA model with 7 topics is not a good candidate but a 8 topic model is much better.As we increase the total number of topics, topics become less and less distinguishable and the number of articles under some topics become very small. Although the choice of number of topics is flexible, I would recommend the number of be less than 50 in order for the model output to make sense to human.

CHAPTER 4

Real World LDA Examples

In addition to its original usage as a general purpose clustering algorithm, LDA can also be used as components for more sophisticated models. Here I gave three use cases from my own experiences.

4.1 Smart Online Content Generation System

In online content business, it is crucial to understand the popular and trending topics on both search engine and social platforms such facebook, twitter and pinterest. However, as mentioned in previous chapters, simply knowing the search volume and social popularity is not enough for an online publisher to succeed any more. At least three other factors should be considered before any decision should be made on investment for contents: domain authority, competition and difficulty. I will not go too deep on these factors but one should keep in mind that the search engine has evolved to be smart enough to pick only the good quality content from well known online publishers to present to users. Relevancy of urls on a page is a very important factor for page ranking and thus the contents should be organized in the most efficient way possible.

The very first step is to construct a topic universe that is closely related to the business. Usually people will use google analytics to collect top search queries that drive traffic to the site. This is a reasonable way but sometimes it could be misleading. Following this method, you only know what is currently the most popular search phrases on your site but will never have an idea how to expand to a larger phrase universe. And since the tastes of audience

changes so fast, one may end up getting less and less traffic although it is still the best content provider for certain search queries. Thus, I take advantage of LDA output and use the topic representations to expand the query universe. Further, by running LDA multiple times, articles can be clustered in a cascaded manner and one can use the output of LDA to refresh a large scale taxonomy.

In figure 4.1, I put LDA in the context of a automated online content generation system. The input of the system is simply a domain name, a list of its competitors' domain name and a list of popular phrases. The system will search online to find related contents and cluster them into topics. Next, the system will expand the topic representation to a larger phrase universe using google autocompletes. Depending on how large you want your final phrase universe to be, combinations of expansion strategies can be used to achieve any decent size of phrase universe. After that, the system will query various social data venders and generate social popularity score. At the same time, if the client needs, the system will also scrape google trends api to download relateive search volume for all the search phrases in the universe. Sequentially, seasonality score and novelty score based on search volume and patterns can be computed. For each search phrase, the system will gather a collection of related contents and pass into pre-trained LDA model to generate category labels. Finally a category filter will be incorporated into all the reports. The process is highly automated and is designed to save time for search engine optimization process and to facilitate content generation process.

4.1.1 Cascaded LDA for taxonomy building

This is a very practical case in many online content generation business where they have a lot of contents to organize and manage and need an accurate taxonomy to group contents. This is important since google search engine will panelize a website if the content is not well organized and your content will have very little chance to position well in google search results. Previously, companies may need to hire several senior editors or experienced writers to manually go through all the documents and come up with their own taxonomy. How-

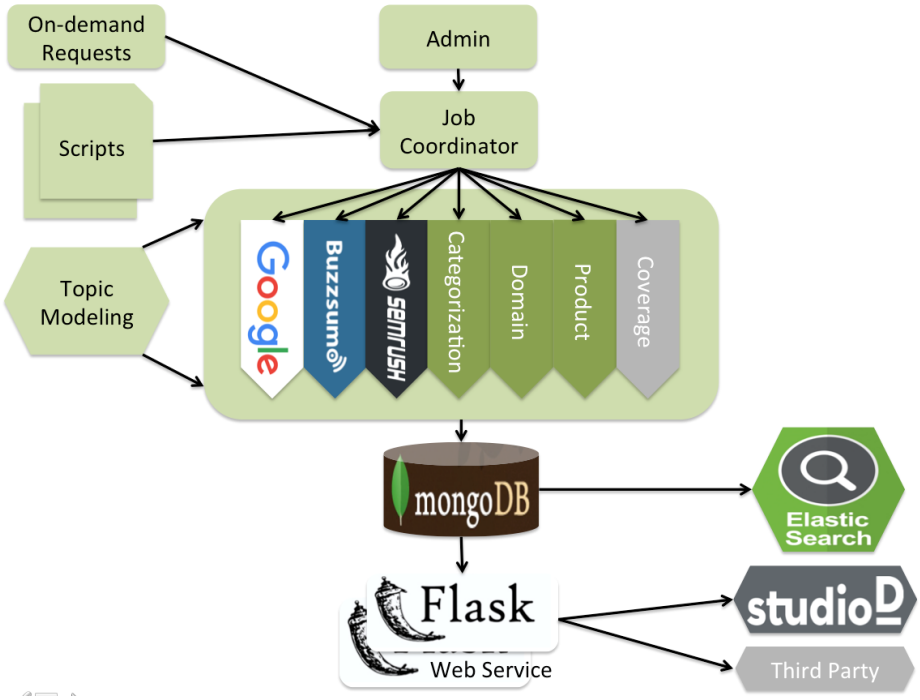


Figure 4.1: The schematic diagram of smart content generation system

ever, this is a painful process. First, each person will definitely propose different taxonomy. Second, the process can take up to several months since you may have millions of articles.

Cascaded LDA will give you a decent hierarchical taxonomy especially when you need a category at a granular level and at a very low cost as well. The process will be like this: starting from the first level clusters and for each cluster running the same process to get the second level clusters and so on. Depending on the actual number of articles to be clustered, LDA algorithm can go very deep to construct a pretty comprehensive hierarchical taxonomy.For example, for the livestrong.com corpus, I trained a 8-topic LDA model as the first layer. And within each topic, I trained another LDA model as the second layer.Hierarchically supervised latent Dirichlet allocation [PWE11] has also been proposed with a focus on the improvement of out-of-sample prediction accuracy. With more information available(accurate labels), those model can perform significantly better.

4.1.2 In-cluster Similarity

Article similarity is a very useful measure for applications like de-duplication and article recommendations. De-duplication is widely used in many domains. For online user-generated-content, it is highly possible that different user will post similar or highly related contents at the same time. For applications like facebook and twitter, this is not a problem and it even helps generate trending topics. But for business like reddit and slickdeals, this is a big headache since they want users to be all under a big post in stead of posting separately. For online content recommendation, calculating content similarity is also important. For business like Amazon, this is probably not a big issue because everyone log into their account before any user behavior happens. However, for many online shopping websites, people tend to browse as a guest and the company will not have any information about the users(or at least take extra efforts if not super expensive). The best guess of what a user might click on next is the most similar content to whatever s/he is consuming right now. However, the computation process becomes very expensive as size of corpus grows since you need one-vs-all evaluation across the whole corpus. For websites like ehow.com and wikihow.com, they have millions of content pieces on the site and probably millions of user on the site as well. It is unacceptable to compute one-vs-all similarity score and because of this an effective off-line clustering algorithm is very important that it will, up to a certain probability confidence interval, that what you compute within the cluster is similar to what you compute among the whole corpus. One way to solve this problem is by using LDA to cluster the articles first and then only compute the in-cluster similarity. Although, the in-cluster similarity equals cross-corpus similarity, from a practical point of view, this will give pretty good similarity score while dramatically reduce the computation time. Although not always the case, ideally a good clustering algorithm should give a local minimum score that is close to global minimum.

4.1.3 Auto Categorization

Another application that leverages the power of LDA is to combine the output of LDA with human knowledge. In many problem settings, we might not be able to get correctly labeled corpus or even unlabeled corpus. It is economically unacceptable to hire many editors or experience writers to read through all articles and come up with their own way of categorization. They might probably give very different categories to the same article given their different educational background. To make sure the categorization process is consistent over time, a reliable model that delivers reasonable category information for each articles is necessary. Since the output topic is represented by a list of probability weighted words. We can manually label the topic and then use that for any other supervised learning tasks such classification. This can be easily handled by any college graduates since it is just a list of 20-30 related words. In the domain of search engine optimization, the number of search phrases are overwhelmingly large and people need a filter to screen out the phrases from a different field. This is important because different online publisher may focus on totally different topics but the search phrases can be from anywhere. We can algorithmically categorize all phrases by applying a trained LDA model and label the phrases with topics. Although the label LDA gives is pre-defined, we can take advantage of online LDA to further expand the training dataset as long as one can find highly related contents to a specific topic. Recent research[LOZ15] has shown that with a proper adjustment to the original LDA algorithm, one can use LDA for labeled supervised learning as well.

CHAPTER 5

Conclusion

In this paper, I first introduce the history of topic modeling and the state-of-the-art inference methods including Variational Bayes, collapsed gibbs sampling,Collapsed Variational Bayesian Inference and Fast Collapsed Gibbs Sampling. Then I briefly introduce the history of online content business and why LDA is a perfect solution for some of problems in current online content generation. I applied LDA to cluster 100,000 health related articles using gensim LDA solver and discuss the choice of LDA parameters and model outputs.A comparison between cosine similarity measure and KL divergence measure is given as a reference for evaluating model performance. Finally, I discuss three real world use cases of LDA model a components for more sophisticated systems including cascaded LDA, in-cluster similarity computing and auto-categorization using supervised LDA.LDA is a powerful tool in machine learning, although its major application is in natural language processing field, many other use cases can be found in computer vision machine translation and beyond.

References

[AGR03] Leif Azzopardi, Mark Girolami, and Keith van Risjbergen. "Investigating the relationship between language model perplexity and IR precision-recall measures." In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 369–370. ACM, 2003.

[BC00] Ken Barker and Nadia Cornacchia. "Using noun phrase heads to extract document keyphrases." In *Advances in Artificial Intelligence*, pp. 40–52. Springer, 2000.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. "Latent dirichlet allocation." *the Journal of machine Learning research*, **3**:993–1022, 2003.

[DBJ12] Lan Du, Wray Buntine, Huidong Jin, and Changyou Chen. "Sequential latent Dirichlet allocation." *Knowledge and information systems*, **31**(3):475–503, 2012.

[DDL90] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. "Indexing by latent semantic analysis." *JAsIs*, **41**(6):391–407, 1990.

[HBB10] Matthew Hoffman, Francis R Bach, and David M Blei. "Online learning for latent dirichlet allocation." In *advances in neural information processing systems*, pp. 856–864, 2010.

[Hof99] Thomas Hofmann. "Probabilistic latent semantic indexing." In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57. ACM, 1999.

[Joa99] Thorsten Joachims. "Making large scale SVM learning practical." Technical report, Universität Dortmund, 1999.

[LOZ15] Ximing Li, Jihong Ouyang, Xiaotang Zhou, You Lu, and Yanhui Liu. "Supervised labeled latent Dirichlet allocation for document categorization." *Applied Intelligence*, **42**(3):581–593, 2015.

[LZC11] Zhiyuan Liu, Yuzhou Zhang, Edward Y Chang, and Maosong Sun. "Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing." *ACM Transactions on Intelligent Systems and Technology (TIST)*, **2**(3):26, 2011.

[MHB12] David Mimno, Matt Hoffman, and David Blei. "Sparse stochastic inference for latent Dirichlet allocation." *arXiv preprint arXiv:1206.6425*, 2012.

[NWF08] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. "Unsupervised learning of human action categories using spatial-temporal words." *International journal of computer vision*, **79**(3):299–318, 2008.

[Pai13] Jiaul H Paik. "A novel TF-IDF weighting scheme for effective ranking." In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 343–352. ACM, 2013.

[PAN10] Duangmanee Putthividhy, Hagai T Attias, and Srikantan S Nagarajan. "Topic regression multi-modal latent dirichlet allocation for image annotation." In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3408–3415. IEEE, 2010.

[PNI08] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. "Fast collapsed gibbs sampling for latent dirichlet allocation." In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 569–577. ACM, 2008.

[PWE11] Adler J Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. "Hierarchically supervised latent Dirichlet allocation." In *Advances in Neural Information Processing Systems*, pp. 2609–2617, 2011.

[TNW06] Yee W Teh, David Newman, and Max Welling. "A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation." In *Advances in neural information processing systems*, pp. 1353–1360, 2006.

[WG08] Xiaogang Wang and Eric Grimson. "Spatial latent dirichlet allocation." In *Advances in neural information processing systems*, pp. 1577–1584, 2008.

[WMM09] Hanna M Wallach, David M Mimno, and Andrew McCallum. "Rethinking LDA: Why priors matter." In *Advances in neural information processing systems*, pp. 1973–1981, 2009.