



Review and Implementation of Topic Modeling in Hindi

Santosh Kumar Ray, Amir Ahmad & Ch. Aswani Kumar

To cite this article: Santosh Kumar Ray, Amir Ahmad & Ch. Aswani Kumar (2019) Review and Implementation of Topic Modeling in Hindi, Applied Artificial Intelligence, 33:11, 979-1007, DOI: 10.1080/08839514.2019.1661576

To link to this article: <https://doi.org/10.1080/08839514.2019.1661576>



Published online: 05 Sep 2019.



Submit your article to this journal [↗](#)



Article views: 1793



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 16 View citing articles [↗](#)



Review and Implementation of Topic Modeling in Hindi

Santosh Kumar Ray^a, Amir Ahmad^b, and Ch. Aswani Kumar^c

^aDepartment of IT, Al Khawarizmi International College, Al Ain, UAE; ^bDepartment of IT, College of Information Technology UAE University, Al Ain, UAE; ^cFaculty of Engineering & IT, Vellore Institute of Technology, Vellore, India

ABSTRACT

Due to the widespread usage of electronic devices and the growing popularity of social media, a lot of text data is being generated at the rate never seen before. It is not possible for humans to read all data generated and find what is being discussed in his field of interest. Topic modeling is a technique to identify the topics present in a large set of text documents. In this paper, we have discussed the widely used techniques and tools for topic modeling. There has been a lot of research on topic modeling in English, but there is not much progress in the resource-scarce languages like Hindi despite Hindi being spoken by millions of people across the world. In this paper, we have discussed the challenges faced in developing topic models for Hindi. We have applied Latent Semantic Indexing (LSI), Non-negative Matrix Factorization (NMF), and Latent Dirichlet Allocation (LDA) algorithms for topic modeling in Hindi. The outcomes of the topic model algorithms are usually difficult to interpret for the common user. We have used various visualization techniques to represent the outcomes of topic modeling in a meaningful way. Then we have used the metrics like perplexity and coherence to evaluate the topic models. The results of Topic modeling in Hindi seem to be promising and comparable to some results reported in the literature on English datasets.

Introduction

Introduction of mobile technology and social media has helped the use of the Internet in a way never seen before. The number of Internet users increased from 745 million in 2004 to 4388 million in 2019¹, in a span of just 15 years. The growth of the Internet is not limited to increase in the number of users only; the growth of user-created contents on the Internet is much higher as users are spending more time on the Internet. The Internet, being the most democratic medium of communication, allows users to publish their contents anytime, to express their opinions on any topic. However, it is almost impossible for an individual human to read all the documents of his/her interest. This created the

CONTACT Santosh Kumar Ray  santosh.ray@kic.ac.ae  Department of IT, Al Khawarizmi International College, Al Ain, UAE

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uai.

CCS CONCEPTS

Artificial Intelligence, Natural Language Processing, Information Extraction.

© 2019 Taylor & Francis

need for techniques and tools to analyze the collection of documents/opinions and produce a summarized result about the theme of the document collections. Also, if the summary of the results could be displayed in some visual form, it will make things much easier for the users. Topic modeling is one such unsupervised learning technique that helps users to analyze and understand the themes hidden inside unlabeled text documents. As shown in Figure 1, topic modeling assumes that a document consists of several hidden topics. Though the topic modeling cannot understand the meaning of the words and concepts in a text, it leverages the context around the words to capture the hidden concepts in the document collections and produces the hidden topics in the documents with certain probabilities assigned to terms in each document. Thus, topic modeling can be considered as a clustering problem where the number of topics, like the number of clusters, is a hyperparameter. The difference between the two is that topic modeling group words instead of numeric features. Also, if required, the results produced by topic modeling techniques can be presented in visual forms, such as word clouds or inter topic distance map, to provide a better understanding of the texts in the document collection.

Due to its ability to organize a collection of documents into clusters of topics, topic modeling has been applied in several areas. Zhang et al. (Zhang et al. 2017) used topic modeling approach to develop a healthcare recommender system named *iDoctor* to provide personalized and professionalized guidelines for users to choose medical services. It is a well-known fact that the effects of drugs vary from person to person depending upon the variations in human genes. Wu et al. (Wu et al. 2012) used topic modeling to rank candidate gene-drug relations from

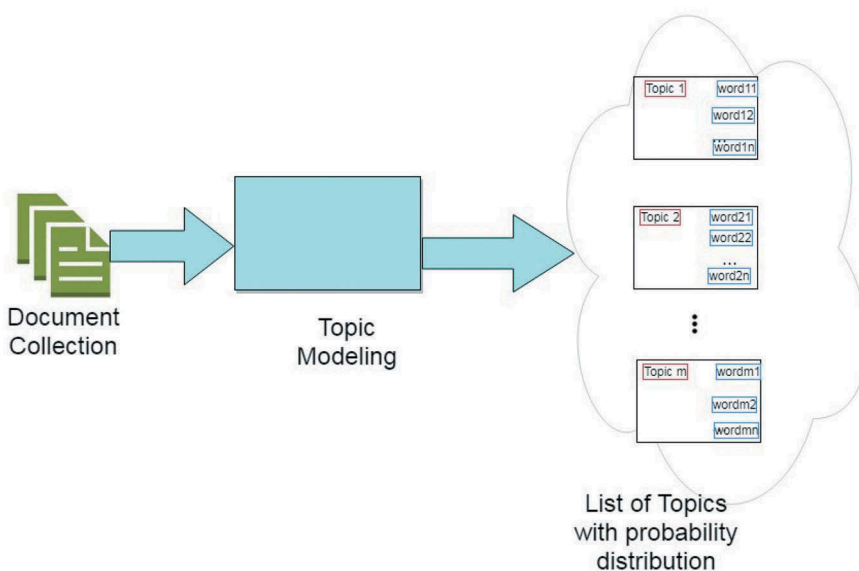


Figure 1. Topic modeling.

MEDLINE abstracts of pharmacogenomics literature. Green and Cross (Greene and Cross 2015) used topic modeling algorithm to analyze the European Parliament plenary from the period 1999–2014. They analyzed the speech contents in the plenary using dynamic topic modeling method based on two layers of matrix factorization. They observed that the political agenda of the European Parliament has evolved significantly over time, is impacted upon by the committee structure of the Parliament and reacts to exogenous events. Topic modeling can help in recommending interesting articles to researchers by offering articles related to area of their research. This will save efforts and time of researchers in finding articles by keyword search through search engines. The problem with searching through search engines is that its output depends on the right formulation of the search query. The way for researchers to find articles is through citations from other articles. By using citations from other articles, the researchers may be at risk of missing some more important articles just because the authors of the original article did not cite them (Musto et al. 2015). Topic modeling has also been used in modeling the evolution of topics in software repositories (Thomas et al. 2011) and mining concepts from software codes (Linstead et al. 2007; Gethers and Poshyvanyk 2010).

Hindi is the fourth most spoken language in the world. Hindi, a language based on the famous Paninian grammar, is known for its syntactic richness. Nevertheless, research and development in the field of topic modeling in Hindi is almost non-existent. At the time of writing this article, a google search of “Hindi Topic Modeling” did not return any article related to research on Hindi topic modeling though there were some researches on applications of multilanguage topic models for machine translation (Kanojia et al. 2015; Kanojia et al. 2016). Therefore, to the best of our knowledge, this work is the first research work on topic modeling on Hindi texts. The aim of this article is to introduce the new researchers to the methods and tools of Topic Modeling in general and to Hindi topic modeling. We have also introduced the visualization techniques and evaluation metric. This will help in creating an easy platform to boost the research in Hindi Topic modeling.

The rest of the article is organized as follow: [Section 2](#) provides a brief description of the three major topic modeling algorithms namely, Latent Semantic Indexing (LSI), Non-negative Matrix Factorization (NMF), and Latent Dirichlet Allocation (LDA). [Section 3](#) discusses the challenges posed by the inherent constraints of the Hindi language and lack of computational linguistic resources for Hindi. [Section 4](#) introduces some popular but easy to use topic modeling tools and programming language library helpful in implementing topic modeling algorithms discussed in [section 2](#) in addition to other related topic models. [Section 5](#) describes the tasks and subtasks in topic modeling in Hindi that includes data collection, preparation, tokenization, topic modeling and producing outputs. [Section 6](#) discusses the results, visualization of results for better interpretation and metrics for evaluation of the topic

modeling algorithms. Finally, [section 7](#) discusses the conclusion and future scopes.

Introduction to Topic Modeling Algorithms

Since the first topic model was proposed in late 1990s, it has received a lot of attention and generated widespread interest among researchers in many research fields. In this section, we will discuss three of the most widely used topic modeling algorithms. There are several variations of these topic modeling algorithms which are essentially based on the major algorithms discussed in this paper. Due to constraints of space, we are providing only a brief introduction of the three major algorithms. More details on these topic modeling algorithms, variational models and their applications can be found in (Alghamdi and Alfalqi 2015).

Latent Semantic Analysis (LSA)

Latent Semantic Analysis, also referred as Latent Semantic Indexing (LSI), is a knowledge representation technique that creates a vector-based presentation of the content of a text (Dumais et al. 1988; (Thomas, Foltz, and Laham 1998). The underlying idea behind LSA is that the aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other. LSA does not require any human constructed dictionary, grammar, parser or any other tool. Its input is only the raw text files. Each document in the corpus is represented as a word count vector of length W where W is the number of words in the corpus dictionary. The dictionary is usually created using the corpus itself. Thus, the corpus can be represented by a matrix, called document-term matrix, of dimension $D \times W$ where D is the number of documents in the corpus. Each cell of the matrix contains the TF-IDF score of the word in the corresponding document. Then LSA uses Singular Value Decomposition (SVD) on the matrix to map documents and terms to a vector space of reduced dimensionality (equal to the number of desired topics), the latent semantic space (Deerwester et al. 1990). This reduced latent semantic space is further used to find similar words and documents by using techniques such as cosine similarity method. LSA model has been used to replicate semantic categorical clustering of words found in certain neuropsychological (Laham 1997), sentence comprehension (Kintsch 1998), selection of reviewers for a paper (Dumais and Nielsen 1992) and research article recommendation (Foltz and Dumais 1992). Demonstration of some of the applications of LSA can be seen on the website².

Probabilistic Latent Semantic Analysis (PLSA)/ Non-negative Matrix Factorization (NMF)

Introduced by Thomas Hofmann in 1999, PLSA is a topic modeling method that improves LSA (Hofmann 1999). While LSA is based on linear algebra, the PLSA model uses a more principled approach that is based on a mixture decomposition derived from a latent class model called aspect model (Hofmann, Puzicha, and Jordan 1999). In this model, each document is considered an unordered set of words. This model associates the topics with the document-word pairs. However, the words and documents in the document collection are assumed to be conditionally independent for a topic. The likelihood estimation and model fitting are done with the help of Expectation Maximization algorithm (Dempster, Laird, and Rubin 1977). It identifies and distinguishes different contexts of words without referring to a dictionary. PLSA allows disambiguating words and detecting similarities by grouping words sharing common contexts. Another way to present PLSA is as a matrix factorization approach. Then linear algebra-based algorithms are used to factorize high-dimensional sparse document term matrices into low-dimensional matrixes with non-negative coefficients. PLSA algorithm has been used in auto-annotation of images (Monay and Gatica-Perez 2004) and object categorization (Sivic et al. 2005).

Latent Dirichlet Allocation (LDA)

In order to overcome the issues with PLSA, such as overfitting, a fully generative Bayesian model was proposed in 2003 by Blei et al. (Blei, Ng, and Jordan 2003). This model is one of the most widely unsupervised machine learning frameworks for topic modeling. LDA is an unsupervised learning-based model, it means that LDA does not require any previous training data with training labels. LDA model takes a set of documents and the desired number of topics as the input parameters. This model assumes that each document is a multinomial distribution of topics and each topic is a multinomial distribution of words. So, for a given corpus of documents, LDA model chooses a multinomial distribution ϕ_t for a topic t ($t \in \{1 \dots, T, T \text{ is the number of desired topics}\}$) from a Dirichlet distribution with parameter β . It also chooses another multinomial distribution θ_d for document d ($d \in \{1 \dots, M, M \text{ is the number of documents in the corpus}\}$) from a Dirichlet distribution with parameter α . Several methods have been proposed for better estimation of the parameters of the LDA model; Gibbs sampling (Griffiths and Steyvers 2004), expectation propagation (Minka and John 2002), being some of them. Then, LDA produces the desired number of topics with each topic being a list of words with their probability distributions. By looking at the list of the words, a human can make some guess about the context of the topic. Similarly, by analyzing the probability distribution of the topics of two or more documents, we can compute the similarity between the

two or more documents. This is the reason that the LDA topic models have been used in wide variety of text mining applications such as research citations (Nallapati et al. 2008), role discovery in social networks (McCallum, Wang, and Corrada-Emmanuel 2007), automatic essay grading (Kakkonen, Myller, and Sutinen 2006). One of the limitations of LDA is that it requires a pre-defined number of topics (K). If K is too small, then the topics are more general in nature, and if K is too large, the topics will be overlapping with each other. Another limitation of LDA is that it does not automatically construct a list of stopwords. If required, the list of the stopwords should be provided by the model developer/user.

Each of the models discussed in this section has its own strengths. For example, while LDA is better in learning descriptive topics, LSA is better in creating a semantic representation of documents in a corpus (Stevens et al. 2012). That is why there are several variants of these models specially LDA model for topic modeling and related tasks have been proposed over the years. Jelodar et al. (Jelodar et al. 2017) have provided a detailed survey of various LDA based topic models and their applications.

Challenges to Hindi Topic Modeling

Hindi, one of the two official languages of India, is the fourth most-spoken language in the world after Mandarin, Spanish and English³. Hindi is written using Devanagari script. The most basic unit of writing Hindi is *Akshara*, which can be a combination of consonants and vowels of the language. Words are made of *aksharas*. Words can also be constructed from other words using grammatical constructs called *Sandhi* and *Samaas*. Though Hindi is a syntactically rich language, it has certain inherent characteristics that make the computer-based processing of the documents in this language, from the information retrieval point of view, a very difficult task. In this section, we are presenting some of these challenges.

1. No Capitalization: Some of the text-mining tasks require to identify the names of locations, persons and other proper nouns in an efficient way. Identification of proper nouns is done by special NLP modules called named entity recognizers, which typically exploit the fact that proper nouns in many languages including English are usually started with capital letters. However, the Hindi language does not use the capitalization feature to distinguish proper nouns to other word forms such as common nouns, verbs or adjectives. For example, the Hindi proper name “ममता” (pronounced as Mamta, means affection) can be used in a sentence as a first name, or as a common noun without any change in the aksharas.

2. Lack of uniformity in writing styles: The lack of standardization of spellings of the Hindi words leads to the generation of variants of the same word that are

spelled differently but still refers to the same word with the same meaning, creating a many-to-one ambiguity. In Hindi grammar, there is a rule called *panchmakshar* (fifth letter) that states that if the fifth letter of a class of consonants precedes any of the four remaining letters of the same class, the *Anuswar* (the dot above a letter) can be used in place of that fifth letter. For example, the word Anand (means happiness but it can be the name of a person also) can be spelled as आनंद or आनन्द. Also, it is not uncommon to see the wrong spellings of most frequent words, for example the word “हूँ (pronounced as hoon, means am)” is wrongly written as “हू”. Also, another language Urdu which uses *Nukta* (dot below letters) has a lot of influence on Hindi which causes variations in spellings of words. For example, “कागज़” (pronounced as kagaj, meaning paper, written with one nukta below the third letter) is also spelled as “कागज” (written without nukta) or “कागज़” (with two nuktas).

3. Expressions with multiple words: It is very common in the Hindi language to use a word (or words with similar meaning) consecutively twice in the same sentence. For example, the word कौन (who) is used as कौन-कौन in a plural sense, धीरे (slow) is used as धीरे-धीरे to emphasize low speed, बहुत (many) सारे (all) are combined as बहुत सारे (so many). This type of usage of words can be crucial in the tokenization process, or it can even negatively affect the performance of cross-language NLP applications where translation from one language to another language is needed.

4. Vaalaa morpheme constructs: The “वाला (vaalaa)” Hindi morpheme is frequently used in Hindi as a suffix to construct new words or to modify the verbs in a sentence. It can take different forms according to gender and number form of the base noun. For example, if we add “vaalaa suffix to the word चाय (pronounced as chai, means tea), a new word चायवाला (pronounced as chaiwala, means male tea seller) will be formed. However, if we add “vaali” suffix to the word घर (pronounced as ghar, means house), a new word घरवाली (pronounced as gharwali, means wife) will be formed. This can make the automatic word sense disambiguation task more complex.

5. Lack of NLP resources and tools: Unlike English, there is a scarcity of language processing software tools in the Hindi language. Even as commonly used software as MS excel does not provide enough support to the Hindi language. Some versions of MS excel do not provide CSV UTF-8 format that makes storage of Hindi texts a difficult task. Also, different software uses different encodings for Hindi characters or words. It means that some words may be represented in more than one way, depending on the presence or absence of ZWJ or ZWNJ (Zero Width Joiner or Zero Width Non-Joiner). This causes errors in processing the text by the codes or software. For example, if we have stored Hindi stopwords in a CSV file created using notepad, due to different encodings these stopwords are not identified by the python code due to different encoding of the same word in the CSV file and Python library.

Most of the packages of popular programming languages such as python or R or Java do not support linguistic operations related to Hindi.

Tools and Datasets for Topic Modeling

There are several tools available for topic modeling. Some of them are very easy to use as they have graphical interfaces. Even a nonprogrammer can use them. However, the use of programming language gives more flexibility and power in doing topic modeling. For example, many of the tools with GUI/CLI may not support documents from languages other than English. So, anyone who wants to do topic modeling for other languages may need to use programming languages. Most of the major programming languages like Python, R and Java provide libraries/ classes to support topic modeling. In fact, most of these tools have been developed by using Python and Java. In this section, we are providing a brief introduction to some of these tools.

Mallet (MACHINE Learning for Language Toolkit)

MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text (McCallum 2002). As Mallet is a java-based application, it can be either used as a standalone application, or it can be deployed as the part of a larger application. An easy to use GUI implementation of Mallet tool is available at GitHub⁴. Figure 2 shows the initial screen of the Mallet GUI which asks for locations of input directory (where text dataset is stored) and output directory (where output in mallet file format, csv format and html format) will be stored. The user can decide the number of topics as an output parameter (the default value is 10). Mallet also provides a mechanism to change the parameters through the optional settings button. As seen in Figure 2, there is a default regular expression to tokenize the English texts. For documents in other texts, users need to provide required regular expression to avoid incorrect output. Similarly, the user can set other parameters.

Stanford Topic Modeling Toolbox

Stanford Topic Modeling Toolbox⁵ is a topic model tool developed at Stanford NLP group by Daniel Ramage and Evan Rosen (Ramage et al. 2009). This tool is developed using Scala and requires scripts written in Scala to run the topic models. Its graphical interface allows to import text data, manipulate the imported data using Scala Scripts and find useful information such as word usage across topics. Figure 3 shows the GUI of this tool. Though it is still in use, Stanford NLP group no longer updates and supports this tool. This makes this

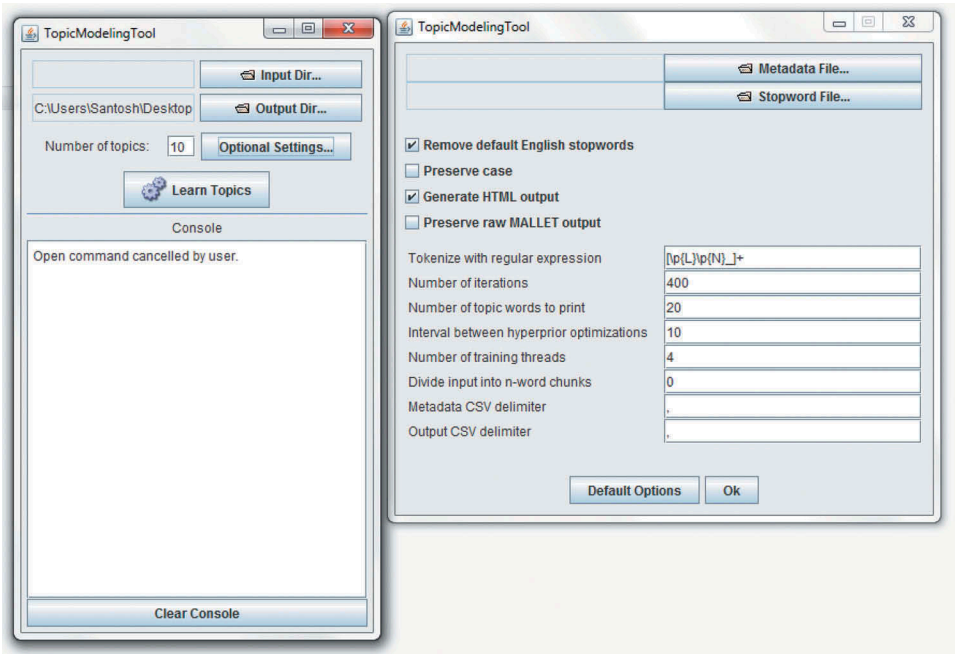


Figure 2. Mallet topic modeling tool.

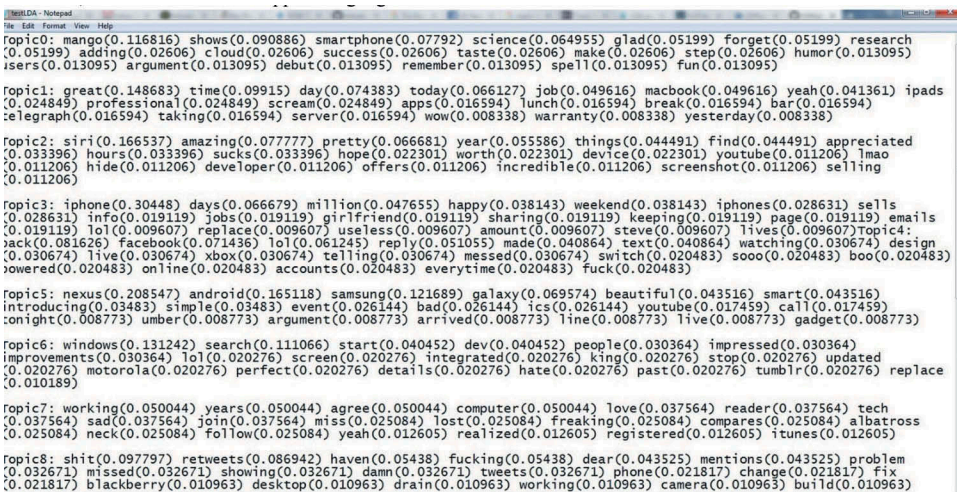


Figure 3. Stanford topic modeling toolbox.

tool less attractive. That is why we have also not used this tool in our experiments.

jLDADMM and STTM

jLDADMM is java implementation of the Latent Dirichlet Allocation topic model and the one-topic-per-document Dirichlet Multinomial Mixture model (i.e. a mixture of unigrams), using collapsed Gibbs sampling. jLDADMM supplies a document clustering evaluation to compare topic models, using two common metrics of Purity and normalized mutual information (NMI) (Nguyen 2018). jLDADMM is freely available and can be downloaded from GitHub⁶. The downloaded package contains jLDADMM tool as a .jar file, which can be run from the command line. This command line-based tool provides options for



```

topic0: mango(0.116816) shows(0.090886) smartphone(0.07792) science(0.064955) glad(0.05199) forget(0.05199) research
(0.05199) adding(0.02606) cloud(0.02606) success(0.02606) taste(0.02606) make(0.02606) step(0.02606) humor(0.013095)
isers(0.013095) argument(0.013095) debut(0.013095) remember(0.013095) spell(0.013095) fun(0.013095)

topic1: great(0.148683) time(0.09915) day(0.074383) today(0.066127) job(0.049616) macbook(0.049616) yeah(0.041361) ipads
(0.024849) professional(0.024849) scream(0.024849) apps(0.016594) lunch(0.016594) break(0.016594) bar(0.016594)
telegraph(0.016594) taking(0.016594) server(0.016594) wow(0.008338) warranty(0.008338) yesterday(0.008338)

topic2: siri(0.166537) amazing(0.077777) pretty(0.066681) year(0.055586) things(0.044491) find(0.044491) appreciated
(0.033396) hours(0.033396) sucks(0.033396) hope(0.022301) worth(0.022301) device(0.022301) youtube(0.011206) imao
(0.011206) hide(0.011206) developer(0.011206) offers(0.011206) incredible(0.011206) screenshot(0.011206) selling
(0.011206)

topic3: iphone(0.30448) days(0.066679) million(0.047655) happy(0.038143) weekend(0.038143) iphones(0.028631) sells
(0.028631) info(0.019119) jobs(0.019119) girlfriend(0.019119) sharing(0.019119) keeping(0.019119) page(0.019119) emails
(0.019119) lol(0.009607) replace(0.009607) useless(0.009607) amount(0.009607) steve(0.009607) lives(0.009607) Topic4:
jack(0.081626) facebook(0.071436) lol(0.061245) reply(0.051055) made(0.040864) text(0.040864) watching(0.030674) design
(0.030674) live(0.030674) xbox(0.030674) telling(0.030674) messed(0.030674) switch(0.020483) sooo(0.020483) boo(0.020483)
lowered(0.020483) online(0.020483) accounts(0.020483) everytime(0.020483) fuck(0.020483)

topic5: nexus(0.208547) android(0.165118) samsung(0.121689) galaxy(0.069574) beautiful(0.043516) smart(0.043516)
introducing(0.03483) simple(0.03483) event(0.026144) bad(0.026144) ics(0.026144) youtube(0.017459) call(0.017459)
tonight(0.008773) umber(0.008773) argument(0.008773) arrived(0.008773) line(0.008773) live(0.008773) gadget(0.008773)

topic6: windows(0.131242) search(0.111066) start(0.040452) dev(0.040452) people(0.030364) impressed(0.030364)
improvements(0.030364) lol(0.020276) screen(0.020276) integrated(0.020276) king(0.020276) stop(0.020276) updated
(0.020276) motorola(0.020276) perfect(0.020276) details(0.020276) hate(0.020276) past(0.020276) tumblr(0.020276) replace
(0.010189)

topic7: working(0.050044) years(0.050044) agree(0.050044) computer(0.050044) love(0.037564) reader(0.037564) tech
(0.037564) sad(0.037564) join(0.037564) miss(0.025084) lost(0.025084) freaking(0.025084) compares(0.025084) albatross
(0.025084) neck(0.025084) follow(0.025084) yeah(0.012605) realized(0.012605) registered(0.012605) itunes(0.012605)

topic8: shit(0.097797) retweets(0.086942) haven(0.05438) fucking(0.05438) dear(0.043525) mentions(0.043525) problem
(0.032671) missed(0.032671) showing(0.032671) damn(0.032671) tweets(0.032671) phone(0.021817) change(0.021817) fix
(0.021817) blackberry(0.010963) desktop(0.010963) drain(0.010963) working(0.010963) camera(0.010963) build(0.010963)

```

Figure 4. The sample output of jLDADMM.

several parameters such as the name of the model (LDA or DMM), location of the corpus, number of topics, number of words per topic, number of iterations and so on. Many of these parameters have default values. Each line of the input text file is considered as one document. Figure 4 shows the output of jLDADMM for LDA model on the sample corpus provided with the tool itself. After running 2000 iterations (default), it produces an output that contains 15 topics, each topic containing default 20 words. However, this model does not support languages like Hindi.

STTM (Short Text Topic Modeling) is also a Java-based topic modeling tool. It implements LDA topic model in addition to many short text topic models such as Dirichlet Multinomial Mixture, BiTerm Topic Model and other models (Qiang et al. 2018). They have used LF-DMM and LF-LDA model from the jLDADMM tool itself. STTM is available to download from the website⁷. This tool provides six datasets namely SearchSnippets, StackOverflow, Biomedical, Tweet, GoogleNews and Pascal_Flickr. This tool also provides methods to compare and evaluate topic models by computing metrics like coherence, purity, and NMI.

HLTA

HLTA (Hierarchical Latent Tree Analysis) tool is a Java implementation of the hierarchical topic detection algorithm (Chen et al. 2016). HLTA takes text files (.txt and pdf) as input and converts them to bag-of-words representation. Then, a model is built from the bag-of-words representation using hierarchical latent tree model. The leaves of the output of the model represent the presence/absence of words in a document. The internal nodes of the tree are binary latent variables, with those at the lowest latent level representing word co-occurrence

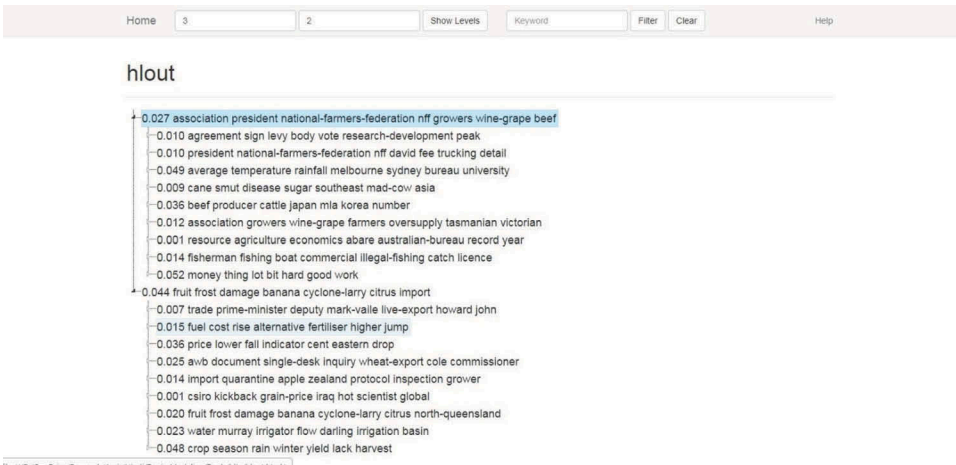


Figure 5. Sample output of HLTATool on a text document.

patterns and those at higher levels representing co-occurrence of patterns at the level below. [Figure 5](#) shows a sample output of the HLTATool on the *rural.txt* file (part of nltk package in Python). HLTATool is available to download from [GitHub](#)⁸.

Python Libraries

Python is a hugely popular programming language. It provides several libraries that help in performing NLP tasks such as document summarization and topic modeling. The most widely used python package for topic modeling is *Genism*⁹. It provides implementations of major topic algorithms such as LSI, LDA, Random Projections, Hierarchical Dirichlet Process (HDP), word2vec deep learning, etc (Rehurek and Sojka 2010). Implementations of all the models in this package are independent of memory size, which allows it to handle even large size text corpora. The second relevant library for Topic modeling from Python is *Bigartm* library¹⁰ which is based on a technique called Additive Regularization of topic models (ARTM) (Vorontsov and Potapenko 2014). *Bigartm* allows to develop, train and test various models like PLSA and ARTM. It also implements several quality measures of topic models like matrix sparsity, perplexity, topic mass, coherence, etc. There are some other packages in python, which though do not provide topic modeling functions but provide supplementary functions of topic modeling. These packages include nltk, NumPy, Panda, Udpipes, Polyglot, Sklearn, Bokeh and Matplotlib.

R Packages

Like Python, R is also a very popular programming language especially for data analysis and visualization. Many of the popular topic models have been implemented in both R and Python. Here, we will mention the names of some packages in R that help in topic modeling or related tasks. The *textMineR*¹¹ package provides a wrapper for several topic modeling algorithms including LSA, LDA, and Correlated Topic Models (CTM). In addition, *textMineR* has utility functions for topic models including R-squared for probabilistic topic models, probabilistic coherence and topic labeling. *Udpipe* is another useful R package (implemented in Python also) that does tokenization, parts of speech tagging, lemmatization and dependency parsing of raw text. This package provides direct access to language models trained on more than 50 languages including Hindi. This helps to prepare data for topic modeling in other non-English languages too. Another package in R named *Topicmodels*¹² provides interface to C implementations of LDA and CTM. It also provides methods to calculate the perplexity and log-likelihood of the models. The *lda*¹³ package in R implements LDA and related models such as sLDA, corrLDA and the mixed-membership stochastic block model.

Topic Modeling in Hindi

In this section, we will describe the implementation of topic modeling algorithms discussed in the previous sections, but on Hindi texts only. Though the mathematical details of each topic modeling algorithm vary, the overall modeling implementation, evaluation, and implementation follow the sequence of steps discussed below.

Input: *A set of raw text documents*

Output: *List of topics with the probability distribution of terms in each topic*

Step1: Read the raw documents and remove unnecessary words

Step2: Tokenize the texts of the documents

Step 3: Preprocess the tokens (removal of stop words, punctuation marks, numbers, lemmatization)

Step4: Create a document-term matrix

Step5: Apply topic modeling algorithm

Step6: Evaluate modeling algorithm

Step7: Visualization of results

Data Collection and Preparation

To conduct the research discussed in this article, we adopted the dataset provided at the GitHub website¹⁴. This dataset contains news articles scraped from the websites of two Hindi newspapers named *Amar Ujala* and *Navbharat*

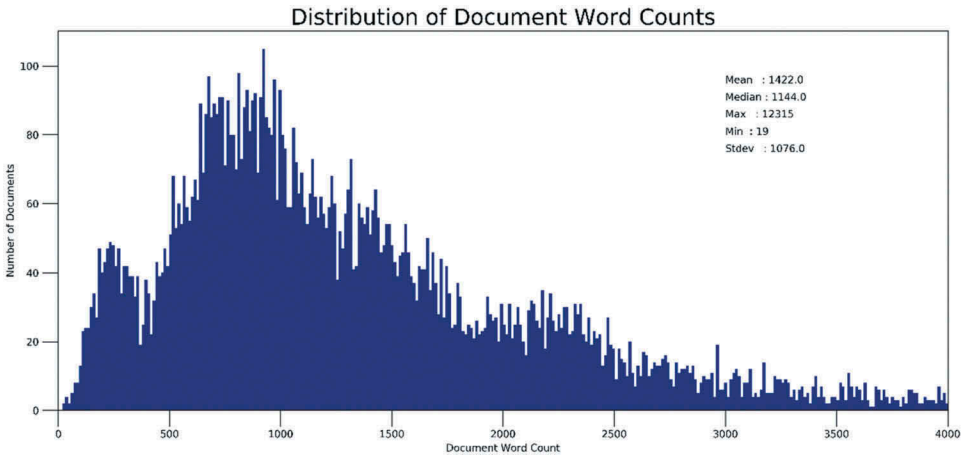


Figure 6. Statistics of the dataset for topic modeling.

Times. This dataset of approximately 38MB size contains articles related to business, education, entertainment, fashion, lifestyle, spirituality, technology, astrology, and sports. The original dataset consists of more than 3 million words distributed over 15 text files. Each text file contains several articles related to a specific field only. We divided this dataset into 10,400 documents. The statistics of the dataset is shown in Figure 6. On average, each document contains 1144 words. Most of the Hindi newspapers and magazines use several English words with Devanagari script. For example, many of the newspapers use the word “फिल्म (film)” instead of the Hindi word “chalachitra.” In fact, the use of such words is so popular that the plural forms of these words are not simple transliterations of the original English words in Devanagari script. Rather, these words are pluralized in the way plural form of a Hindi word is derived. For example, the word “films” is not written as “फिल्म्स”; rather they are written as “फिल्में” or “फिल्मों”. For the purpose of topic modeling, we did not translate these English words in Hindi in Devanagari Scripts as the changes may affect the outcome of the topic modeling. As some frequently occurring words such as “हैं (are)”, “है (is)”, “था (was)” do not contribute in determining the topic in the documents, we have removed these words from the dataset. These words are called stopwords and they are insignificant in computation of topics in the documents. For languages like English, there are many standard sets of stopwords. In fact, many of the packages in the modern programming languages such as Python and R have lists of in-built stopwords, which can be used for major text processing tasks. However, there is no such standard set for Hindi Language. Through Google search, we found few lists of stopwords in Hindi. However, these lists were either insufficient or contained several words with wrong spellings. Therefore, we prepared our own list of stopwords containing about 1000 words. As discussed in Section 3, people use different spellings of the

same words in Hindi language (e.g. “चाहिए” and “चाहिये” both pronounced as “Chahiye,” means should or want). Therefore, we have added all possible, but correct and acceptable, spellings of these words in the list of stopwords. Due to unavailability of efficient stemmer for Hindi language, we had to include all inflectional and derivational forms of a stop word in the list (e.g., डालना, डालनी, डालने, डालता, डालती, डालते). As numbers and digits do not contribute much in topic modeling, we removed all the numeric entities from the datasets. We also noted the presence of English words in the dataset through their frequencies were very low. We removed all such words from the datasets.

Experiment for Hindi Topic Modeling

Once the unnecessary words were removed from the dataset, we lemmatized the documents using *nlTK* package of Python. In fact, we implemented all the modeling algorithms discussed in this article using Python libraries such *nlTK*, *NumPy*, *Pandas*, *Genism*, *Sklearn*, *Bokeh* and *Matplotlib*. Then, we created a dictionary from the document collection using *genism* package. The dictionary created is nothing but a collection of unique terms in the document collection. This dictionary was then used to create a document-term matrix. This document-term matrix is used by each of the models discussed in this article. For all kinds of topic models, we need to provide the number of topics and number of words in each topic as an input to modeling algorithm. We will present the results of each topic model for 15 topics with each topic consisting of 10 words. For such a large dataset, 15 topics are not enough. But considering the limitations of space in representing the optimal results, we are representing the results of each model with 15 topics only. Later, we will determine the optimal number of topics based on something called *coherence*.

Results

The result of a topic modeling algorithm is traditionally represented as a list of topics. Each topic consists of a number of words. Each word in a topic is assigned some number that indicates statistical significance of that word in the topic. Depending on the topic model used, this number may indicate either probability distribution or some other semantic similarity of the word in the topic. For LSI model, we will represent the result in the traditional way, that is, each topic will consist of words and some number will be associated with each number. Though each model computes the statistical distributions, for the remaining two models we will drop the numbers representing distributions to make results more presentable and meaningful to the reader. There are other alternatives packages (e.g. *Sklearn*) in python for topic modeling, we used *genism* package of Python to run the models because it is one of the most widely used packages for the topic and vector space modeling tasks. Also, *genism* package has specific classes on LSI

LSI Model:
 [(0, '-0.360**साल' + -0.348**रुपये' + -0.314**टैक्स' + -0.220**फिल्म' + -0.146**पर्सेंट' + -0.141**भासा' + -0.128**समकाम' + -0.127**लाख' + -0.118**कहा' + -0.117**करोड़'), (1, '-0.786**फिल्म' + 0.285**टैक्स' + 0.208**रुपये' + -0.100**दिलीप' + -0.098**टीस' + -0.098**किरदार' + 0.097**पर्सेंट' + -0.086**फिल्मो' + 0.075**लाख' + 0.075**दिलव'), (2, '0.375**टीस' + -0.365**फिल्म' + 0.334**भासा' + 0.252**बैक' + -0.239**टैक्स' + 0.224**गोल' + 0.195**भासावीच' + 0.187**मिक्ट' + 0.123**बै' + 0.122**गीत'), (3, '-0.319**गणेशजी' + -0.285**लाभ' + -0.218**समाड' + -0.199**संभावना' + -0.196**गव' + -0.179**गशि' + -0.169**स्वास्थ्य' + -0.168**मानसिक' + -0.145**सर्व' + -0.139**गोश'), (4, '0.591**टैक्स' + -0.279**साल' + -0.247**रुपये' + -0.208**पद' + -0.133**इलकस' + 0.123**टीस' + -0.117**क्वालिफिकेशल' + 0.114**फिल्म' + -0.111**कॉलेज' + 0.106**दिलव'), (5, '0.457**फोव' + -0.239**साल' + 0.206**रुपये' + 0.206**कंपनी' + 0.177**कैमरा' + 0.173**स्मार्टफोव' + -0.167**कॉलेज' + -0.150**टैक्स' + 0.141**गोबी' + -0.132**कटऑफ' + -0.307**कॉलेज' + -0.302**फोव' + -0.249**कटऑफ' + -0.223**ऑवर्स' + -0.221**स्टूडेंट्स' + -0.211**एडमिशन' + -0.171**टैक्स' + -0.154**कोर्स' + -0.125**कॉलेजो'), (7, '-0.384**टैक्स' + -0.254**पद' + -0.252**फोव' + 0.181**फंड' + 0.178**पर्सेंट' + 0.175**बैक' + 0.166**आकेंट' + -0.137**क्वालिफिकेशल' + 0.123**फंडस' + 0.122**गोल्ड'), (8, '-0.515**रुपये' + 0.341**साल' + 0.273**फंड' + 0.201**फंडस' + -0.175**कॉलेज' + 0.162**दिलव' + 0.162**विशेष' + -0.149**कटऑफ' + -0.135**लाख' + -0.134**ऑवर्स'), (9, '0.525**मिक्ट' + 0.371**बजकर' + 0.268**गोल' + 0.147**गोश' + 0.141**वखन' + -0.134**कहा' + 0.133**संवत्' + -0.130**ऑलिंपिक' + 0.129**कप' + 0.129**फंड'), (10, '-0.285**बैक' + 0.237**घर' + 0.216**लुक' + 0.200**दिस' + 0.137**आकेंट' + -0.137**गणेशजी' + 0.133**स्टाइल' + -0.133**फिल्म' + 0.126**कप' + 0.126**शी'), (11, '-0.477**बैक' + 0.228**फंड' + 0.189**फंडस' + -0.169**समकाम' + 0.167**रुपये' + 0.161**गणेशजी' + 0.155**दिलव' + 0.138**पर्सेंट' + 0.131**विशेष' + 0.122**आकेंट'), (12, '-0.413**बैक' + 0.231**साल' + -0.179**लुक' + -0.171**दिस' + 0.170**कंपनी' + 0.163**गशि' + -0.157**गणेशजी' + 0.148**समाड' + -0.145**कई' + -0.136**अकाउंट'), (13, '-0.367**समाड' + -0.305**गशि' + 0.284**गणेशजी' + 0.267**कंपनी' + 0.162**मिक्ट' + -0.134**व्यक्ति' + 0.129**समकाम' + -0.124**रुपये' + 0.119**बजकर' + -0.113**फंड'), (14, '-0.420**समकाम' + 0.237**गोल्ड' + 0.221**सियु' + 0.196**साल' + 0.191**ऑलिंपिक' + -0.178**टीस' + 0.137**बैक' + 0.134**फोव' + -0.129**समाड' + 0.124**फिसदी)]]

Figure 7. Output of LSI model.

	0	1	2	3	4	5	6	7	8	9
Topic # 01	फेसबुक	ऐप	फोन	कंपनी	मोबाइल	स्मार्टफोन	गूगल	वीडियो	यूजर्स	लॉन्च
Topic # 02	सरकार	करोड़	भारत	देश	फीसदी	लाख	प्रतिशत	साल	कहा	बैंक
Topic # 03	घर	जीवन	शरीर	व्यक्ति	भगवान	मन	रूप	मां	मंदिर	हो
Topic # 04	गेम	क्वालिफिचे	कंडोम	आसन	मॉक	दर्द	सेक्स	कमर	अभ्यास	कार्यक्षेत्र
Topic # 05	धन	एडमिशन	दिल्ली	स्कूलों	डाटा	नक्षत्र	ऑनर्स	एडमिशन	कॉलेज	स्कूल
Topic # 06	लुक	पानी	ड्रेस	मैसेज	शॉपिंग	स्टाइल	कलर	बालों	फैशन	रंग
Topic # 07	फिल्म	रिलीज	साल	कमाई	फिल्मों	किरदार	कहा	खान	फिल्में	शूटिंग
Topic # 08	शो	कहा	टीम	टेस्ट	कपिल	मित्रों	क्रिकेट	कोहली	ऑस्ट्रेलिय	शर्मा
Topic # 09	भारत	टीम	मैच	भारतीय	ओलिंपिक	जीत	खेल	फाइनल	कहा	खिलाडी
Topic # 10	मिनट	बजकर	गोल	करण	योग	संवत्	सौर	राष्ट्रीय	प्रातः	पेनल्टी
Topic # 11	रुपये	टैक्स	बैंक	पर्सेंट	कंपनी	साल	फंड	रिटर्न	मार्केट	गोल्ड
Topic # 12	सेक्स	महिलाओं	साल	मूलाबिक	घर	लाइफ	शादी	बच्चों	बच्चे	प्यार
Topic # 13	होली	तेल	मिर्च	कप	प्याज	चाय	गोवा	फीड	नमक	मिनट
Topic # 14	लाभ	वर्ष	गणेशजी	योग	स्वास्थ्य	संभावना	मानसिक	मन	राशि	आर्थिक
Topic # 15	साल	ऑफ	स्टूडेंट्स	ऑनलाइन	बोर्ड	परीक्षा	रुपये	दिल्ली	पद	कोर्स

Figure 8. Output of LDA model.

and LDA models which can be used directly and thus eliminating the need for implementation of the models from scratch. Besides, this package has classes to compute the coherence score of the models.

LSI Model

As shown in Figure 7, LSI produces 15 topics numbered 0–14 with each topic containing 10 words. Though numbers are not easy to interpret, by seeing the words in each topic, a human can make an easy guess about the topic. For example, topic 1 is about movies, topic 4 is about astrology, and topic 8 is about education.

LDA Model

As discussed earlier, in order to make the representation of topics more understandable to readers, we have presented the result of LDA topic model without

```

Topic 0:
[स्पष्ट, तास्व, कयोड, कीमत, मार्केट, गोलड, सप्रकार, गिपट, हजारा, दाम, साल, रेज, फीसदी, टन, दिल्ली]
Topic 1:
[फिल्म, प्रितीज, किरदार, फिल्मों, फिल्में, शूटिंग, गोल, कमाई, खान, बॉलिवुड, कयोड, अक्षय, सीन, विदेशक, कपूर]
Topic 2:
[टीम, भारत, मैच, भारतीय, गोल, जीत, ओलिंपिक, खेल, वे, ऑस्ट्रेलिया, डॉकी, फाइलत, खिताबी, मुकाबले, वर्चोटर]
Topic 3:
[गणेशजी, लाभ, मन, मानसिक, स्वास्थ्य, शारीरिक, मित्रों, संभावना, रूप, स्वर्च, धन, प्रवास, सलाह, योग, वृद्धि]
Topic 4:
[पद, साल, क्वालिफिकेशन, अप्लीकेशन, ऑनलाइन, एज, कंप्यूटर, लिमिटे, वेबसाइट, पोस्ट, साईंस, एक्सपीरियंस, फीस, डिग्री, डेट]
Topic 5:
[फोन, कैमरा, स्मार्टफोन, मोटी, जीवी, ऐप, स्क्रीन, डिस्टो, बैटरी, ऐडॉबड, कार्ड, परफॉर्मेंस, ऐप्स, फ्रंट, मेगापिक्सल]
Topic 6:
[कॉलेज, कटऑफ, ऑनर्स, एडमिशन, स्टूडेंट्स, कोर्स, कॉलेजों, डीयू, कोर्सेज, बीकॉम, बीए, यूनिवर्सिटी, इंविगिन, दिल्ली, ऑफ]
Topic 7:
[टैक्स, इनकम, रिटर्न, सप्रकार, पसेंट, तास्व, जीएसटी, छूट, सर्विस, डिपार्टमेंट, फाइल, बजट, प्रकम, जानकारी, लागू]
Topic 8:
[फंड, फंड्स, निवेश, रिटर्न, पसेंट, इक्विटी, साल, म्यूचुअल, मार्केट, पोर्टफोलियो, पैसा, शेयर, टर्म, इनवेस्टर्स, डेट]
Topic 9:
[मिनट, बजकर, गोल, योग, नक्षत्र, संवत्, करण, प्रातः, मास, मुस्लिम, राष्ट्रीय, शक, विक्रम, चन्द्रमा, सौर]
Topic 10:
[साल, दिल्ली, बोर्ड, घर, सप्रकार, परीक्षा, ची, टी, स्कूल, सेक्स, बच्चों, फीसदी, शो, देश, बच्चे]
Topic 11:
[बैंक, लोन, बैंकों, अकाउंट, कार्ड, ब्याज, गेट, टर, मित्रव, फीसदी, ऑफ, सप्रकार, पेमेंट, जमा, स्टेट]
Topic 12:
[लुक, ट्रेस, स्टाइल, क्लर, कैरी, मार्केट, फेशन, ट्रेसेज, जूमी, ट्रेड, मेकअप, डिजाइन, क्लर्स, बालों, शॉपिंग]
Topic 13:
[समाह, माशि, संभावना, लाभ, व्यक्ति, जीवन, शुभ, शनि, सूर्य, आर्थिक, बुध, संबंध, वर्ष, नौकरी, घर]
Topic 14:
[कंपनी, शेयर, पसेंट, कंपनियों, ब्रोथ, भारत, स्मार्टफोन, मार्केट, कयोड, उम्मीद, लॉन्च, इंडिया, बिजनेस, सेगमेंट, चील्ड]

```

Figure 9. Output of NMF model.

statistical information of each word in a topic. As we can see from the Figure 8, topic#01 is about technology, topic#04 is about lifestyle and topic#08 is about entertainment.

NMF Model

As *genism* package does not support the NMF topic model, we used *sklearn* package in Python to run this topic model. As shown in Figure 9, NMF model selects 15 topics, each comprising of 10 words. Topic 0 is about businesses, topic 6 is about education and topic 12 is about fashion and lifestyle.

Topic Modeling Using Mallet Tool

Finally, we are presenting the result of the topic modeling using Mallet tool. We are including the results of this tool to demonstrate the use of this tool on Hindi topic modeling for those users who do not prefer to write codes. The default regular expression of the Mallet tool will not support Hindi topic modeling. We used regular expression “[\p{L}\p{M}]+” for tokenizing the Hindi texts in the Mallet tool. Figure 10 shows the output of the Mallet tool. As seen from the result topic 0 is about entertainment, topic 4 is about sports and topic 8 is about technology.

0. फिल्म शो रिलीज साल किरदार फिल्मों खान कपिल शूटिंग कमाई
1. मिनट योग बजकर लाभ वर्ष गणेशजी राशि जीवन मन घर
2. सेक्स महिलाओं मुताबिक पुरुषों महिलाएं लाइफ रिसर्च पार्टनर पुरुष देखिए
3. टैक्स रुपये बैंक साल सरकार पैसेट करोड़ कंपनी लाख फ्रीसदी
4. टीम भारत मैच भारतीय ओलिंपिक गोल जीत खेल फाइनल मिनट
5. लुक ड्रेस स्टाइल मार्केट घर रुपये पानी साल शॉपिंग कलर
6. दो देश तीन चार हाथ महिला डी घंटे पांच खुशी
7. व्यक्ति घर जीवन शरीर मन पढ़ें राम रूप पिता गुरु
8. फोन कंपनी स्मार्टफोन ऐप लॉन्च रुपये फेसबुक जीबी मोबाइल कैमरा
9. साल स्टूडेंट्स परीक्षा कोर्स दिल्ली पद बोर्ड कॉलेज ऑफ रुपये

Figure 10. Output of mallet tool.

Visualization and Evaluation of Topic Modeling

In this section, we will discuss the methods to visualize the results of topic modeling. We will also present some interesting aspects of the results. Then we will evaluate the results of topic modeling through metrics like perplexity and coherence.

Topic Visualization

As it is evident from the results in the previous section, the output of a topic model is not easily interpretable to users. In order to help the common users to interpret the results of topic modeling in a better way, visualization of results is a good option. As discussed in [section 1](#), a document may consist of more than one topic. However, usually one topic is dominant in that document. The user may be interested in knowing what topic is dominant in the specific document. For example, we applied the LDA topic model on a document containing information related to the business world. [Figure 11](#) shows the results of topic modeling on the document *AmarUjala_business.txt*. It is clear from the result that though there are multiple topics contained in this document, the most prominent is related to finance (tax, bank, company, etc). The contribution of finance in this topic is about 75%.

In social media, users may be interested in a set of tweets discussing a specific topic that is trending. This can be modeled as a topic modeling problem. Each tweet can be treated as a document and the trend as a topic. Then we can find out what are the tweets discussing a specific trend. In the context of topic modeling, this has been illustrated in [Figure 12](#). [Figure 12](#) shows the documents containing the most dominant topic in the previous result (from [Figure 11](#)). Numbers in the bracket indicates the number of words from a document contributing to the specific topic.

DOC :AmarUjala_business.txt

संस्था की मांग है कि बिना आईएसआई मार्क वाले आरओ फिल्टर की बिक्री पर तत्काल रोक लगनी चाहिए। सहारा इंडिया परिवार के मुखिया सुब्रत रॉय सहारा को सुप्रीम कोर्ट से राहत मिल गई है। सराफा बाजार में बुधवार को उथल-पुथल जारी है। जहां सोना घरेलू बाजार में सस्ता हुआ है वहीं चांदी महंगी हो गई है। सोना 20 रुपये सस्ता हो गया है, जिसके साथ ही इसका दाम 28787 रुपये प्रति 10 ग्राम हो गया है। वहीं चांदी ने उछाल मारी है, जो कि चांदी 159 रुपये महंगी हो गई है। चांदी का दाम 42281 रुपये प्रति किलोग्राम पहुंच गया है। अगर आपने ई-कॉमर्स कंपनियों से कोई सामान खरीदा है और वह सामान खरीदारी के दौरान ऑनलाइन दिखने वाले सामान की तरह नहीं है या फिर आपको दी गई जानकारी से खरीदा जाने वाला सामान मेल नहीं खा रहा है या ऑनलाइन खरीदे गए सामान को वापस करने में दिक्कत हो रही है तो आपको परेशान होने की जरूरत नहीं है। आपकी समस्या का निवारण महज 100 रुपये में हो जाएगा। कानून के बड़े बड़े जानकार हैं कॉर्पोरेट कंपनियों से मुझ करगने में आपकी मदद करेंगे।

Top topics in this doc (% words in doc assigned to this topic)

- (75%) टैक्स रुपये बैंक साल सरकार पर्सेंट करोड़ कंपनी लाख फीसदी ...
- (12%) फोन कंपनी स्मार्टफोन ऐप लॉन्च रुपये फेसबुक जीबी मोबाइल कैमरा ...
- (5%) दो देश तीन चार हाथ महिला डी घंटे पांच खुशी ...
- (3%) व्यक्ति घर जीवन शरीर मन पढ़ें राम रूप पिता गुरु ...
- (1%) सेक्स महिलाओं मुताबिक पुरुषों महिलाएं लाइफ रिसर्च पार्टनर पुरुष देखिए ...
- (1%) मिनट योग बजकर लाभ वर्ष गणेशजी राशि जीवन मन घर ...
- (1%) फिल्म शो रिलीज साल किरदार फिल्मों खान कपिल शूटिंग कमाई ...
- (1%) साल स्टूडेंट्स परीक्षा कोर्स दिल्ली पद बोर्ड कॉलेज ऑफ रुपये ...
- (0%) लुक ट्रेस स्टाइल मार्केट घर रुपये पानी साल शॉपिंग कलर ...
- (0%) टीम भारत मैच भारतीय ओलंपिक गोल जीत खेल फाइनल मिनट ...

Figure 11. Dominant topic in a document.

TOPIC : टैक्स रुपये बैंक साल सरकार पर्सेंट करोड़ कंपनी लाख फीसदी ...

top-ranked docs in this topic (#words in doc assigned to this topic)

1. (168905) NBT_business.txt
2. (47643) AmarUjala_business.txt
3. (7383) NBT_education.txt
4. (3373) NBT_lifestyle.txt
5. (1724) AmarUjala_technology.txt
6. (1345) NBT_tech.txt
7. (1258) NBT_sports.txt
8. (887) NBT_movie-masti.txt
9. (822) NBT_astro.txt
10. (481) AmarUjala_education.txt
11. (396) AmarUjala_entertainment.txt
12. (172) AmarUjala_18-plus.txt
13. (83) AmarUjala_fashion.txt
14. (61) AmarUjala_spirituality.txt
15. (56) AmarUjala_lifestyle.txt

[Index]

Figure 12. List of documents containing a specific topic.

left side. Similar chart and its zoom version for term topic distribution are shown in [Figure 14](#).

A better alternative is to produce an interactive visual diagram. We used *pyLDAVis* package to produce an interactive visual diagram for the LDA model ([Figure 15](#)). This package produces an interactive chart in which topics are shown on the left side while words in the topic are shown on the right side. These visual charts give many useful insights into the topics. Each topic is represented by a bubble. The size of the bubble indicates the relevance of the topic represented by it. Larger the size of the bubble, larger is the relevance of the topic in the corpus. Also, topics closer on the plot are more similar than the topics which are farther from them. The interactive plot allows to select any bubble. Alternatively, a topic can be selected by entering the topic number at the textbox provided at the top of the plot. When we select a topic, we can see the most representative words for the selected topic (shown in red color at the bottom of the [Figure 15](#)). User can adjust the weight of each property using the slider (by changing the value of λ). Here λ is a weight parameter that decides the relevance of a word in a topic (Chuang, Manning, and Heer 2012; Sievert and Shirley 2014). This provides user certain flexibility to define his own topic.

Evaluation

In this subsection, we shall discuss the evaluation of topic model algorithms. There are many approaches to evaluate Topic models. Though none of them can match the accuracy of human-in-the-loop approach of evaluation of model (Chang et al. 2009), we need to try theoretical evaluation models as human evaluation has certain limitations in terms of time and complexity. In this section, we will use two most-used methods for topic evaluation of topic models: perplexity and coherence. However, before discussing these two metrics of evaluation, we will decide what will be the optimal number of the topics for our datasets. This is a crucial aspect as computations of both metrics need the number of topics in advance. Therefore, we experimented to find the relation between the number of topics and coherence and to find the optimal number of topics for the maximum coherence. We have calculated the coherence score (CV) for LDA and LSI models using the number of topics as the changing parameter. We changed the number of topics from 1 to 200 and calculated coherence for each number. In [Figure 16](#), the graph labeled as “C” represents results for LDA model and that labeled as “O” represents LSI. From the figure, we can see that if we select 15 topics for our dataset, it will produce the maximum coherence for both models. Therefore, for the computation of both evaluation metrics, perplexity and coherence, we have used 15 as the number of topics parameters for all the functions.

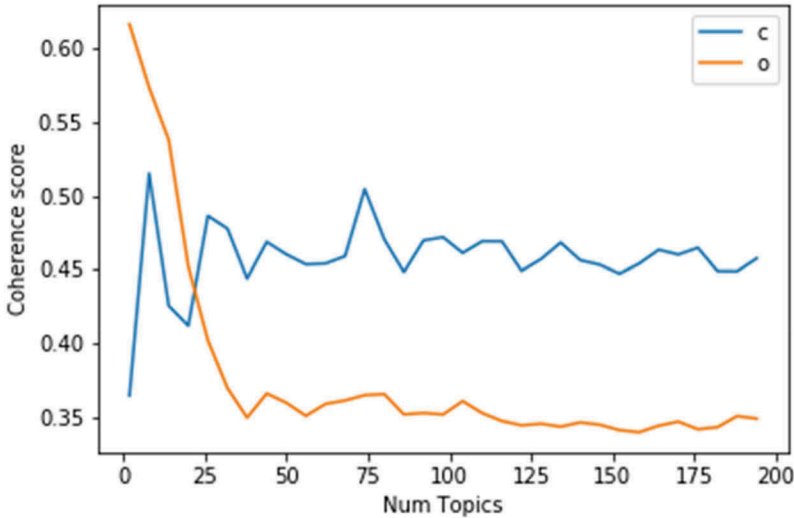


Figure 16. Coherence of LSI and LDA models.

Perplexity

The perplexity of a language model tries to find how effectively a trained model handles an unknown dataset. To compute the perplexity of a language model, the dataset is divided into a training set and a test set. Then the log-likelihood of the unseen documents is calculated using Equation (1)

$$L(w) = \sum_d \log_2 p(w_d | \phi, \alpha) \quad (1)$$

where w_d is a set of unseen documents, Φ is a set of topics and α is a hyperparameter for the topic distribution of documents. The higher value of loglikelihood indicates that the language model is better. Perplexity is calculated using Equation (2)

$$\text{Perplexity}(w) = 2^{\frac{-L(w)}{\text{count of tokens}}} \quad (2)$$

Perplexity is the generative probability of the evaluation document. The lower the value of the perplexity, the better is model. The perplexity is calculated on completely unseen documents. To calculate perplexity, we used a different dataset downloaded from the Internet¹⁵. It consists of text related to automobiles, entertainment, lifestyle, sports, and technology collected from news websites Jagran, NDTV, Hindustan, Navbharat, Ptraika and Sanjeevni. We took the first 10,000 words of each file as a test dataset to calculate the perplexity of the models. Though both *sklearn* and *gensim* have implemented methods to compute perplexity, we used *gensim* package to calculate the perplexity of LDA model. We used *log_perplexity* method from the package *gensim* which returns per-word likelihood bound, using a chunk of documents. This value can be used to calculate the perplexity of the model. The result of our evaluation of LDA

Table 1. Perplexity of LDA model.

Text	per-word likelihood bound (w)	Perplexity (2^{-w})
Auto_text_final	-9.393	672.314
Entertainment_text_final	-9.459	703.609
Lifestyle_text_final	-10.299	1260.303
Sports_text_final	-8.715	420.19
Tech_text_final	-9.655	806.236

model is shown in Table 1. As seen in the Table 1, the perplexity of the datasets in our experiment ranged between 420 and 1260. To see the things in the proper perspective, we will cite some results on English datasets. In their experiment on the evaluation of topic modeling on news and blog data, Newman et al (Newman, Bonilla, and Buntine 2011) calculated perplexity using LDA model. Depending on the type of the document (baseball, drama, health and legal), the perplexity in their experiment ranged between 5000 and 10,000. Similarly, in the research made by Gildea and Hofmann (Gildea and Hofmann 1999), the perplexity of topic model was 829.1 on TDT-1 corpus and 621.1 on Wall Street journal dataset. It shows that perplexity of the LDA model on Hindi dataset is fairly good.

The perplexity of PLSA/ NMF model can be calculated using *bigartm* package. However, as perplexity is considered a poor indicator of the quality of the topics (Newman et al. 2010), we are not calculating the perplexity of the other models.

Coherence

Another commonly used method for evaluation of the topic model is coherence. A set of facts or statements are said to be coherent if they support each other. However, it is not easy to quantify the coherence of a fact (Luc. and Hartmann 2003). One way to quantify the coherence of a topic is to measure the degree of semantic similarity between its high scoring words. To compute the coherence of a topic model, the top n -frequently occurring words in each topic are selected. Then, pairwise scores for each of the words selected above are calculated. These all pairwise scores are then aggregated to calculate the final coherence score for a given topic as shown in Equation (3).

$$Coherence = \sum_{i < j} score(w_i, w_j) \quad (3)$$

There are many coherence measures discussed in the literature. For example, Newman et al. (Newman et al. 2010) proposed an automatic coherence measure called UCI measure (or CV measure) to rate topics for their understandability. This coherence measure treats words as facts and restricts to be always based on comparing word pairs. Some other researchers also proposed to measure topic coherence on the basis of word statistics (Stevens et al. 2012; Lau, Newman, and Baldwin 2014; Mimno et al. 2011). However, in order to evaluate topic models for Hindi documents, we used the CV and

UMass measures, respectively the best measure and the fastest in the literature (Röder, Both, and Hinneburg 2015). The CV measure is an extrinsic measure that relies on external sources such as Wikipedia. The CV measure calculates Pointwise Mutual Information (PMI) using Equation (4)

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j) + \varepsilon}{P(w_i)P(w_j)} \quad (4)$$

where $P(w_i)$ is the probability of finding the word w_i in a random document of the trained model, $P(w_i, w_j)$ is the probability of finding both words w_i and w_j in a random document of the trained model. The CV measure then combines PMI with indirect cosine measure and a Boolean slide window over some external corpus (such as Wikipedia) to produce coherence score. The UMass measure is an intrinsic coherence measure developed by Mimno et al. (Mimno et al. 2011) that relies only upon word co-occurrence statistics gathered from the corpus being modeled and does not depend on an external reference corpus. If we assume $D(v)$ be the number of documents containing words v and $D(v, v')$ be the number of documents containing both words v and v' , then UMass score is computed using the Equation (5)

$$C(t; V^t) = \sum_{m=2}^M \sum_{l=1}^{m-1} \log \frac{D(v_m^{(t)}, v_l^{(t)}) + 1}{D(V_l^{(t)})} \quad (5)$$

where $V^{(t)} = (v_1^{(t)}, \dots, v_M^{(t)})$ is a list of the M most probable words in topic t . A smoothing count of 1 is included to avoid taking the logarithm of zero. As the pairwise score used by the UMass measure is not symmetric, the order of the words in the topic is significant. We used the *coherencemodel* class of *genism* package to compute the coherence scores for LSI, LDA and NMF model. However, instead of computing the coherence score for each topic in the corpus, we have computed the coherence score averaged over all the topics. The result of coherence for these models is shown in Table 2. As we can see that the NMF model is producing better coherence for datasets in Hindi followed by LDA and LSI. Other researchers have computed the coherence score for topic models. But most of the evaluations reported in the literature is on one model only. Here we will compare our results with the results of Stevens et al (Stevens et al. 2012) because they have computed the coherence for all the three models discussed in this article. They calculated

Table 2. Coherence of topic models.

Model	Coherence (CV)	Coherence (Umass)
LSI	0.48500212319154484	-4.118163394061179
LDA	0.6627575907691058	-2.8087249703284565
NMF	0.7975828206735476	-1.6282018183805842

the coherence using CV (UCI) and UMass with the number of topics ranging from 1 to 500. In the topic modeling experiment conducted by them on English dataset (92,600 New York Times articles from 2003), the performance of each of these three models (values and order of performance) is same as that of ours. For example, the coherence score (Umass) of all the models in their experiment varied between an approximate range of -1 to -3 . Also, NMF model performed little better than LDA model which was better than LSI.

Conclusion and Future Scope

Topic Modeling has been studied for more than two decades, but it is gaining more popularity with the increase of online activity of academic, business and common people. People are getting interested in hearing what is happening in the field of their interest. Topic modeling is the right technique to provide users about the happenings in their field of interest. In this article, we have discussed the fundamentals and applications of major topic modeling algorithms. A researcher in any field requires tools and techniques to implement his/her ideas. In this article, we discussed various programming language libraries and tools that can be helpful to researchers in the field of topic modeling.

Hindi being one the major languages of the world deserves more research on NLP tasks such as topic modeling. However, there is a serious shortage of software and tools to support research in topic modeling in Hindi. Most of the software available today do not provide supports to NLP tasks in Hindi. In this article, we have implemented the three major topic models using Python libraries. We have attempted to create resources for topic modeling in Hindi. But there is still a lot of scope for improvement. For example, when creating a list of stopwords, we had to store all the derivational and inflectional forms of the stopwords. Availability of efficient stemmer and lemmatizer can make this task easier. We have evaluated the performance of topic modeling algorithms through metrics such as perplexity and coherence. We have also used Python libraries to present results of topic models in a more efficient way.

Notes

1. <https://www.statista.com/statistics/617136/digital-population-worldwide/>, accessed on 17 February 2019.
2. <http://lsa.colorado.edu>.
3. <https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world/>, accessed on 17 February 2019.
4. <https://github.com/senderle/topic-modeling-tool>.
5. <https://nlp.stanford.edu/software/tmt/tmt-0.4/>.
6. <https://github.com/datquocnguyen/jLDADMM>.
7. <https://github.com/qiang2100/STTM>.

8. <https://github.com/kmpoon/hlta>.
9. <https://pypi.org/project/gensim/>.
10. <http://docs.bigartm.org/en/stable/intro.html>.
11. <https://cran.r-project.org/web/packages/textmineR/index.html>.
12. <https://cran.r-project.org/web/packages/topicmodels/index.html>.
13. <https://cran.r-project.org/web/packages/lda/index.html>.
14. newspapers data from <https://github.com/singhya/TopicModels/tree/master/DataCollection>.
15. <https://www.kaggle.com/pk13055/code-mixed-hindienglish-dataset>.

References

- Alghamdi, R., and K. Alfalqi. 2015. A survey of topic modeling in text mining. *International Journal of Advanced Computer Science and Applications* 6 (1):147–53. doi:10.14569/IJACSA.2015.060121.
- Blei, D., A. Y. Ng, and M. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022. doi:10.1162/jmlr.2003.3.4-5.993.
- Chang, J., J. Boyd-Graber, S. Gerrish, C. Wang, and D. M. Blei. 2009. Reading tea leaves: How humans interpret topic models. *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, 288–96. Canada: Vancouver. <https://dl.acm.org/citation.cfm?id=2984126>.
- Chen, P., N. L. Zhang, T. Liu, L. K. M. Poon, Z. Chen, and F. Khawar. 2016. Latent tree models for hierarchical topic detection. <https://arxiv.org/pdf/1605.06650.pdf>.
- Chuang, J., C. D. Manning, and J. Heer. 2012. Termite. *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '12*, 74. New York: ACM Press. doi:10.1145/2254556.2254572.
- Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41 (6):391–407. doi:10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASII>3.0.CO;2-9.
- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)* 39. <http://web.mit.edu/6.435/www/Dempster77.pdf>.
- Dumais, S. T., G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. Using latent semantic analysis to improve access to textual information. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '88*, 281–85. New York: ACM Press. doi:10.1145/57167.57214.
- Dumais, S. T., and J. Nielsen. 1992. Automating the assignment of submitted manuscripts to reviewers. *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '92*, 233–44. New York: ACM Press. doi:10.1145/133160.133205.
- Foltz, P. W., and S. T. Dumais. 1992. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM* 35 (12):51–60. doi:10.1145/138859.138866.
- Gethers, M., and D. Poshyvaryk. 2010. Using relational topic models to capture coupling among classes in object-oriented software systems. *2010 IEEE International Conference on Software Maintenance*, 1–10. Timisoara, Romania: IEEE. doi:10.1109/ICSM.2010.5609687.

- Gildea, D., and T. Hofmann. 1999. Topic-based language models using EM. *Proceedings of Eurospeech*, Budapest, Hungary, 2167–70. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.833>.
- Greene, D., and J. P. Cross. May 2015. Unveiling the political agenda of the European parliament plenary: A topical analysis. <http://arxiv.org/abs/1505.07302>.
- Griffiths, T. L., and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101 (Supplement 1):5228–35. doi:10.1073/pnas.0307752101.
- Hofmann, T. 1999. Probabilistic Latent Semantic Indexing. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '99*, 50–57. New York: ACM Press. doi:10.1145/312624.312649.
- Hofmann, T., J. Puzicha, and M. I. Jordan. 1999. Unsupervised learning from dyadic data. *Advances in Neural Information Processing System* 11:466–72. <https://dl.acm.org/citation.cfm?id=340706>.
- Jelodar, H., Y. Wang, C. Yuan, and X. Feng. 2017. Latent Dirichlet Allocation (LDA) and topic modeling: models, applications, a survey. *ArXiv. Applications* 78(11):15169–15211. doi:10.16288/j.ycz.17-199.
- Kakkonen, T., N. Myller, and E. Sutinen. 2006. *Applying latent dirichlet allocation to automatic essay grading*, 110–20. Berlin, Heidelberg: Springer. doi:10.1007/11816508_13.
- Kanojia, D., A. Joshi, P. Bhattacharyya, and M. J. Carman. 2015. Using multilingual topic models for improved alignment in English-Hindi MT. *Proceedings of the 12th International Conference on Natural Language Processing*, 308–15. Trivandrum, India. https://www.cse.iitb.ac.in/~adityaj/multilingtopicmodels_icon15.pdf.
- Kanojia, D., A. Joshi, P. Bhattacharyya, and M. J. Carman. 2016. That'll do fine!: A coarse lexical resource for English-Hindi MT, using polylingual topic models. *LREC* 2199–203. http://www.lrec-conf.org/proceedings/lrec2016/pdf/531_Paper.pdf
- Kintsch, W. 1998. *Comprehension: A paradigm for cognition*. New York: Cambridge University Press.
- Laham, D. 1997. Latent semantic analysis approaches to categorization. *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, edited by M. G. Shafto and P. Langley, 979. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc. <http://lsa.colorado.edu/papers/categories.pdf>.
- Lau, J. H., D. Newman, and T. Baldwin. 2014. Machine reading tea leaves: automatically evaluating topic coherence and topic model quality. *14th Conference of the European Chapter of the Association for Computational Linguistics*, 530–39. Stroudsburg, PA: Association for Computational Linguistics. doi:10.3115/v1/E14-1056.
- Linstead, E., P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. 2007. Mining concepts from code with probabilistic topic models. *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering - ASE '07*, 461–64. New York: ACM Press. doi:10.1145/1321631.1321709.
- Luc, B., and S. Hartmann. 2003. *Bayesian epistemology*. Oxford: Clarendon Press.
- McCallum, A., X. Wang, and A. Corrada-Emmanuel. 2007. Topic and role discovery in social networks with experiments on Enron and academic email. *Journal of Artificial Intelligence Research* 30 (October):249–72. doi:10.1613/jair.2229.
- McCallum, A. K. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- Mimno, D., H. M. Wallach, E. Talley, M. Leenders, and M. Andrew 2011. Optimizing semantic coherence in topic models. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 262–72. Edinburgh, Scotland: Association for Computational Linguistics. <https://dl.acm.org/citation.cfm?id=2145462>.

- Minka, T., and L. John. 2002. Expectation-propagation for the generative aspect model. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 352–59. Alberta, Canada: Morgan Kaufmann Publishers. <https://dl.acm.org/citation.cfm?id=2073918>.
- Monay, F., and D. Gatica-Perez. 2004. PLSA-based image auto-annotation: constraining the latent space. *Proceedings of the 12th Annual ACM International Conference on Multimedia - MULTIMEDIA '04*, 348–51. New York: ACM Press. doi:10.1145/1027527.1027608.
- Musto, C., G. Semeraro, M. De Gemmis, and P. Lops. 2015. Word embedding techniques for content-based recommender systems: An empirical evaluation. *CEUR Workshop Proceedings*, 1441:448–56. New York: ACM Press. doi:10.1145/2020408.2020480.
- Nallapati, R. M., A. Ahmed, E. P. Xing, and W. W. Cohen. 2008. Joint latent topic models for text and citations. *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 08*, 542–50. New York: ACM Press. doi:10.1145/1401890.1401957.
- Newman, D., E. V. Bonilla, and W. Buntine. 2011. Improving topic coherence with regularized topic models. *NIPS'11 Proceedings of the 24th International Conference on Neural Information Processing Systems*, 496–504. Granada, Spain. <http://papers.nips.cc/paper/4291-improving-topic-coherence-with-regularized-topic-models.pdf>.
- Newman, D., J. Lau, K. Grieser, and T. Baldwin. 2010. Automatic evaluation of topic coherence. *Human Language Technologies: The* doi:10.1186/s12888-015-0700-x.
- Nguyen, D. Q. 2018. JLDADMM: A Java Package for the LDA and DMM Topic Models. *ArXiv Preprint ArXiv:1808.03835*. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.695.8156>. <https://arxiv.org/abs/1808.03835>
- Qiang, J., L. Yun, Y. Yuan, W. Liu, and W. Xindong. 2018. STTM: A tool for short text topic modeling. *ArXiv Preprint ArXiv:1808.02215*. <https://arxiv.org/abs/1808.02215>
- Ramage, D., E. Rosen, J. Chuang, D. A. Manning, and C. D. McFarland. 2009. Topic modeling for the social sciences. In *NIPS 2009 workshop on applications for topic models*, 1–4. Whistler, Canada: Neural Information Processing System Foundationworkshop location.
- Rehurek, R., and P. Sojka. 2010. Software framework for topic modelling with large corpora. *LREC Workshop on New Challenges for NLP Frameworks*, 45–50. <http://numdam.org>.
- Röder, M., A. Both, and A. Hinneburg. 2015. Exploring the space of topic coherence measures. *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*, 399–408. New York: ACM Press. doi:10.1145/2684822.2685324.
- Sievert, C., and K. Shirley. 2014. LDAvis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 63–70. Stroudsburg, PA: Association for Computational Linguistics. doi:10.3115/v1/W14-3110.
- Sivic, J., B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. 2005. Discovering object categories in image collections. *Proceedings of the tenth International Conference on computer vision*, Beijing, China. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.175.1180>.
- Stevens, K., P. Kegelmeyer, D. Andrzejewski, and D. Buttler. 2012. Exploring topic coherence over many models and many topics. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Jeju Island, Korea, 952–61. <https://aclanthology.info/papers/D12-1087/d12-1087>.
- Thomas, L. K., P. W. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes* 25:259–284. <http://lsa.colorado.edu/papers/dp1.LSAintro.pdf>

- Thomas, S. W., B. Adams, A. E. Hassan, and D. Blostein. 2011. Modeling the evolution of topics in source code histories. *Proceeding of the 8th Working Conference on Mining Software Repositories - MSR '11*, 173–82. New York: ACM Press. doi:[10.1145/1985441.1985467](https://doi.org/10.1145/1985441.1985467).
- Vorontsov, K., and A. Potapenko. 2014. Tutorial on probabilistic topic modeling: additive regularization for stochastic matrix factorization. In *Analysis of images, social networks and texts.*, ed. R. Yavorsky, D. Ignatov, M. Khachay, A. Panchenko, and N. Konstantinova, 29–46. Cham: Springer. doi:[10.1007/978-3-319-12580-0_3](https://doi.org/10.1007/978-3-319-12580-0_3).
- Wu, Y., M. Liu, W. Jim Zheng, Z. Zhao, and X. Hua 2012. Ranking gene-drug relationships in biomedical literature using latent dirichlet allocation. *Pacific Symposium on Biocomputing*, Hawaii, USA, 422–33. <http://www.ncbi.nlm.nih.gov/pubmed/22174297>.
- Zhang, Y., M. Chen, D. Huang, W. Di, and L. Yong. 2017. IDoctor: personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems* 66 (January):30–35. doi:[10.1016/J.FUTURE.2015.12.001](https://doi.org/10.1016/J.FUTURE.2015.12.001).