# EMERGING TOPIC MODELS
# CAMCOS REPORT
# FALL 2011

NEETI MITTAL

Abstract. We review the concept of Latent Dirichlet Allocation (LDA), along with the definitions of Text Mining, Topic, and Topic Modeling. We then extend these results to define *Emerging Topic Models* and *Relative Topic Importance.*

## 1. Introduction

The process of generating information from any set of text data using a machine is called Text Mining. Text mining is becoming more and more important each day as more and more information in terms of data is readily available and it is almost impossible to generate quality information or to make any sense of this data without the use of machines. We are all familiar with the online word search using a search engine. This is a form of text mining. But in this paper, we will discuss another form of text mining, mining that generates the subject or theme behind the given set of documents. We call these subjects or theme Topics. A Topic is a distribution of words in a document over a predetermined vocabulary. For this process of finding topics, we use topic modeling. Topic Modeling is the using of methods to automatically assign words in documents to topics. According to Wikipedia, "In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents." In other words, topic modeling is a self-generating statistical method of placing a distribution of topics over words. There are various kinds of topic models but we focused on Latent Dirichlet Allocation , simply called LDA. LDA was first presented as a graphical model for topic discovery by David Blei, Andrew Ng, and Michael Jordan in 2002 [1]. It is a generative process that defines a joint probability distribution over both the observed and hidden random variables (which will be discussed in next paragraph). Simply put, LDA uncovers the thematic structure hidden in a document. It generates the main ideas behind a set of documents, which we call Topics.

## 2. Latent Diriclet Allocation

In LDA, the term Latent refers to the hidden variable. When we feed the documents in a computer, we observe the words in the documents, but the topics are hidden and hence the topics are latent. Dirichlet is the statistical distribution that is used for this process. Allocation points to allocating topics to the documents. Hence LDA allocates a value to the hidden variable by applying a priori of the values of known variables using Dirichlet distribution.

*Date*: September 28, 2012.
We kindly acknowledge the support of the Woodward Fund.

**Definition 2.1.** The Dirichlet distribution, with parameters $a_1, a_2, \ldots, a_K$, is a multivariate distribution of K random variables, $x_1, x_2, \ldots, x_K$. Its density is

$$\mathrm{Dir}(a_1, a_2, \ldots, a_K) \propto \prod_{i=1}^{K} (x_i)^{a_i - 1}$$

To explain Dirichlet Distribution, we present an example that we found on Wikipedia[1]. Consider an urn containing balls of $K$ different colors. Initially, the urn contains $a_1$ balls of color 1, $a_2$ balls of color 2, and so on. Now perform $N$ draws from the urn, where after each draw, the ball is placed back into the urn with an additional ball of the same color. In the limit as $N$ approaches infinity, the proportions of different colored balls in the urn will be distributed as $\mathrm{Dir}(a_1, a_2, \ldots, a_K)$. Simply put, Dirichlet is a distribution over a distribution (David Blei). Like any other model, LDA model is based on certain assumptions. These assumptions are:

- The topics are Dirichlet distributed over the words.
- The documents are Dirichlet distributed over the topics.
- Order of the documents does not matter. (This is a drawback and is exactly what we address.)
- Order of the words in the documents does not matter.
- The number of topics is assumed known and fixed.
- LDA automatically removes all the common words such as "with," "the," "and," etc.
- It also removes all the numeric values, commas, parentheses, etc.
- It eliminates all the words that are repeated many times in the document.
- We only considered the words with 3 or more letters.

Here is the LDA algorithm in the statistical language R based on the work of Grün and Hornik [2]:

**Algorithm 2.2.**

```
JSS_papers <- read.delim(file="c:\\users\\Neeti\\Desktop\\habitat.txt",header=FALSE, sep= " ")
remove_HTML_markup <-
function(s) {
doc <- htmlTreeParse(s, asText = TRUE, trim = FALSE)
xmlValue(xmlRoot(doc))
}
corpus <- Corpus(VectorSource(sapply(JSS_papers,remove_HTML_markup)))
Sys.setlocale("LC_COLLATE", "C")
JSS_dtm <- DocumentTermMatrix(corpus,control = list(stemming = TRUE, stopwords = TRUE,
                          minWordLength = 3,removeNumbers = TRUE, removePunctuation = TRUE))
dim(JSS_dtm)
summary(col_sums(JSS_dtm))
term_tfidf <-tapply(JSS_dtm$v/row_sums(JSS_dtm)[JSS_dtm$i], JSS_dtm$j, mean)
                             *log2(nDocs(JSS_dtm)/col_sums(JSS_dtm > 0))
summary(term_tfidf)
JSS_dtm <- JSS_dtm[,term_tfidf >= 0.0017]
JSS_dtm <- JSS_dtm[row_sums(JSS_dtm) > 0,]
summary(col_sums(JSS_dtm))
dim(JSS_dtm)

k <- 10
```

[1]www.wikipedia.org

```
SEED <- 2010
jss_TM <-
list(VEM = LDA(JSS_dtm, k = k, control = list(seed = SEED)),
VEM_fixed = LDA(JSS_dtm, k = k,
control = list(estimate.alpha = FALSE, seed = SEED)),
Gibbs = LDA(JSS_dtm, k = k, method = "Gibbs",
control = list(seed = SEED, burnin = 1000,
thin = 100, iter = 1000)),
CTM = CTM(JSS_dtm, k = k,
control = list(seed = SEED,
var = list(tol = 10^-4), em = list(tol = 10^-3))))
sapply(jss_TM[1:2], slot, "alpha")
sapply(jss_TM, function(x)
mean(apply(posterior(x)$topics,
1, function(z) - sum(z * log(z)))))
Topic <- topics(jss_TM[["VEM"]], 1)
Terms <- terms(jss_TM[["VEM"]], 5)
Terms[,1:5]
(topics_v24 <-topics(jss_TM[["VEM"]])[grep("/v24/", JSS_papers)])
most_frequent_v24 <- which.max(tabulate(topics_v24))
terms(jss_TM[["VEM"]], 10)[, most_frequent_v24]
```

Now, to make this LDA picture clear, we will discuss an example. We ran the LDA algorithm over a document about Habitats and looked for 5 topics. Here are the results:

| Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 |
|---|---|---|---|---|
| species .028 | plant .019 | habitat .038 | species .021 | species .076 |
| environment .021 | botanical .019 | population .019 | additional .021 | particular .052 |
| effect .014 | physical .019 | species .019 | cycle .021 | analysis .041 |
| references .014 | zoological .014 | natural .019 | effect .021 | population .030 |
| ecosystem .014 | habitat .013 | trophic .019 | help .020 | abundance .024 |
| - | - | - | - | - |
| - | - | - | - | - |
| - | - | - | - | - |

So there are 5 topics, each of them is distributed over a list of words. Probability of each word under each topic is also presented above. This list consists of the entire vocabulary that comprises our Habitat document, even though the probability of each word varies. For example, the word habitat is listed under Topic 2 as well as Topic 3. But its probability under Topic 2 is 0.013 while under Topic 3 is 0.038. Because all the words are listed under each topic, the sum of the probabilities of all the words under any given topic is 1. Under each Topic, as the probability of the words decreases, their position drops. For example, the word species has the highest probability under Topic 1 and hence it is ranked at first position while the word environment has the second highest probability and hence ranked on second position, and so on. LDA has a perfect way of distinguishing the attributes of subject and defining the subject, depending on how we want to use it.

## 3. Our contribution – Emerging Topic Models

The problem with LDA arises when we want to find time sensitive information out of it. If we have some documents that read on some information and we want to find the most recent information out of them, LDA will fail us. As mentioned earlier, LDA is based on the assumption that the order of words or the order of documents does not matter. This means that all the documents are simply "bag of words." This assumption prevents us from differentiating between a prior knowledge and new information. Further in this paper we will discuss what we implemented to overcome this drawback. We extended David Bleis LDA algorithm to automatically detect the emerging topic. We define an emerging topic as a topic that is more prominent now than it was before. For example if we have some data from a year long period of time and we want to find out if there is any new information in the last month data set, then we will run our algorithm over the documents. This leads us to a definition of the concept of *emerging topic*, which is one of our main contributions.

**Definition 3.1** (Emerging topic)**.** If $a_i$ is the prominence of topic $i$ in the entire data set and $A_i$ is the prominence of the same topic in a recent part of the data, then we define that topic $i$ is an *emerging* topic if

$$\frac{A_i}{a_i} > 1.$$

Now, we discuss an example to differentiate between a search engine, LDA algorithm and our algorithm. We know that Prof. Saleem, Prof. Foster, and Prof. Koev; all three of them have taught Numerical Analysis. We also know that Prof. Saleem has taught calculus based analysis, while Prof. Foster and Prof. Koev have taught Matrix based Numerical Analysis. Let us assume that Prof. Koev calls his class Numerical Linear Algebra instead of Numerical Analysis. If we look up for "Numerical Analysis" using a search engine, Prof. Saleems and Prof. Fosters classes will show up but because Prof. Koev calls his class Numerical Linear Algebra, his class will not come up in the search results. While on the other hand, a topic search will not only discover Prof. Koevs class, but it will also place Prof. Koevs and Prof. Fosters class under same topic and differentiate it from Prof. Saleems class. This is so because topic search places a distribution over the attributes. Hence it can differentiate between different attributes, matrix and calculus, in the documents. Now say, Prof. Foster publishes a paper and we want to find out about that paper. LDA does not give us any time sensitive information. It will simply show all the publications by Prof. Foster, but not in orderly fashion. But our algorithm will do this job. It will detect the newly published paper. Further, we will discuss how this works. Our program is written in statistical language R, using LDA package topicmodels by Grun and Hornik [2]. Our algorithm takes three inputs-A document (which we suspect having an emerging topic), an estimated number of topics ($K$), the percentage of "recent" data (which is the "last month" in the above example). The program eliminates all the common words and the words that are repeated many times. Special characters and numeric values are also discarded. The program places a distribution over all K topics and also gives us their prominence in the document set. We define these prominence as $a_1, a_2, \ldots, a_K$. For each topic $i$, we compute its prominence in the recent part of the documents and call it $A_i$. Then the ratio $A_i/a_i$ is computed and sorted in the decreasing order. Topics for which this ratio is greater than 1 are the suggested emerging topics.

Then we go back to the document set and run LDA over them, excluding the "recent" part. We do this this to find out the old topics and hence to exclude those old topics from the suggested new topics. Our algorithm also displays a graph of the suggested emerging topics. Here is our algorithm:

**Algorithm 3.2.**

```
# Read in data.
# Data is in 2 files:  TrainingData.txt, TestingData.txt

TrainingData=read.delim(file="c:\\users\\Neeti\\Desktop\\Train.txt",header=FALSE, sep= " ")
```

```
TestingData=read.delim(file="c:\\users\\Neeti\\Desktop\\Test.txt",header=FALSE, sep= " ")

# Format data.

remove_HTML_markup <-
function(s) {
doc <- htmlTreeParse(s, asText = TRUE, trim = FALSE)
xmlValue(xmlRoot(doc))
}

corpus.training <- Corpus(VectorSource(sapply(TrainingData,remove_HTML_markup)))
corpus.testing <- Corpus(VectorSource(sapply(TestingData,remove_HTML_markup)))

Sys.setlocale("LC_COLLATE", "C")
training_dtm <- DocumentTermMatrix(corpus.training,
control = list( stopwords = TRUE,
minWordLength = 3,
removeNumbers = TRUE,
removePunctuation = TRUE))

testing_dtm <- DocumentTermMatrix(corpus.testing,
control = list( stopwords = TRUE,
minWordLength = 3,
removeNumbers = TRUE,
removePunctuation = TRUE))

dim(training_dtm)
dim(testing_dtm)

summary(col_sums(training_dtm))
term_tfidf.training <-tapply(training_dtm$v/row_sums(training_dtm)[training_dtm$i],
                                        training_dtm$j, mean)
                                      *log2(nDocs(training_dtm)/col_sums(training_dtm > 0))
summary.training = summary(term_tfidf.training)

summary(col_sums(testing_dtm))
term_tfidf.testing <-tapply(testing_dtm$v/row_sums(testing_dtm)[testing_dtm$i],
                                         testing_dtm$j, mean)
                                       *log2(nDocs(testing_dtm)/col_sums(testing_dtm > 0))
summary.testing = summary(term_tfidf.testing)

# Only include terms which have a tf-idf value of at least some cutoff, ensuring that the
# very frequent terms are omitted.  In this case, we use a cutoff of 90% of the median.

training_dtm <- training_dtm[,term_tfidf.training >= 0.90*summary.training["Median"]]
training_dtm <- training_dtm[row_sums(training_dtm) > 0,]
summary(col_sums(training_dtm))
dim(training_dtm)
```

```
testing_dtm <- testing_dtm[,term_tfidf.testing >= 0.90*summary.testing["Median"]]
testing_dtm <- testing_dtm[row_sums(testing_dtm) > 0,]
summary(col_sums(testing_dtm))
dim(testing_dtm)

# Run on training set.

k <- 10
SEED <- 2010
TM1 <-
list(VEM = LDA(training_dtm, k = k, control = list(seed = SEED)),
VEM_fixed = LDA(training_dtm, k = k,
control = list(estimate.alpha = FALSE, seed = SEED)),
Gibbs = LDA(training_dtm, k = k, method = "Gibbs",
control = list(seed = SEED, burnin = 1000,
thin = 100, iter = 1000)),
CTM = CTM(training_dtm, k = k,
control = list(seed = SEED,
var = list(tol = 10^-4), em = list(tol = 10^-3))))

alphas.train = sapply(TM1[1:2], slot, "alpha")

sapply(TM1, function(x)
mean(apply(posterior(x)$topics,
1, function(z) - sum(z * log(z)))))

Topic1 <- topics(TM1[["VEM"]], 1)
Terms1 <- terms(TM1[["VEM"]], 5)
Terms1[,1:5]

# Run on test set.

perplexity(object=TM1$VEM)
perplexity(object=TM1$VEM, newdata=testing_dtm)

LDAcontrolinitial = list(seed=999,
estimate.alpha=TRUE,
estimate.beta=TRUE)

initialmodel = LDA(training_dtm, k = k, control = LDAcontrolinitial)

LDAcontrol = list(seed = 134,
 estimate.alpha=FALSE,
 alpha = alphas.train[1],
 estimate.beta=FALSE,
 initialize="model")
```

```
k <-10
SEED <- 2010
TM2 <-
list(VEM = LDA(testing_dtm, k = k, control = LDAcontrol, model=initialmodel),
VEM_fixed = LDA(testing_dtm, k = k,
control = list(estimate.alpha = FALSE, seed = SEED)),
Gibbs = LDA(testing_dtm, k = k, method = "Gibbs",
control = list(seed = SEED, burnin = 1000,
thin = 100, iter = 10000)),
CTM = CTM(testing_dtm, k = k,
control = list(seed = SEED,
var = list(tol = 10^-4), em = list(tol = 10^-3))))

gamma1 = sapply(TM1[1], slot, "gamma")
gamma2 = sapply(TM2[1], slot, "gamma")

gamma1.mat = matrix(gamma1, ncol=k)
topic.sums.1 = apply(gamma1.mat,2,sum)
topic.importance.1 = topic.sums.1 / sum(topic.sums.1)
topic.labels.1 = seq(1:k)

# Relabel topics in order of importance.
plot(topic.importance.1)
topic.importance.1.sorted1 = sort(topic.importance.1,
decreasing=TRUE, index.return=TRUE)
plot(topic.importance.1.sorted1$x)
topic.importance.1.sorted1$ix

gamma2.mat = matrix(gamma2, ncol=k)
topic.sums.2 = apply(gamma2.mat,2,sum)

topic.importance.2 = topic.sums.2 / sum(topic.sums.2)

topic.labels.2 = seq(1:k)

topic.importance.2.sorted1 = topic.importance.2[topic.importance.1.sorted1$ix]

beta2 = sapply(TM2[1], slot, "beta")
beta2.mat = matrix(beta2, ncol=k, byrow=TRUE)
exp(beta2.mat[,1])

head(sort(exp(beta2.mat[,1]),decreasing=TRUE))

plot(topic.importance.1.sorted1$x, col="black", xlab = "Topic Number",
ylab = "Relative Importance", main="Relative Importance of Topics",
pch=2,
ylim = c(min(topic.importance.1, topic.importance.2)-0.01,
                        max(topic.importance.1, topic.importance.2)+0.01))
```

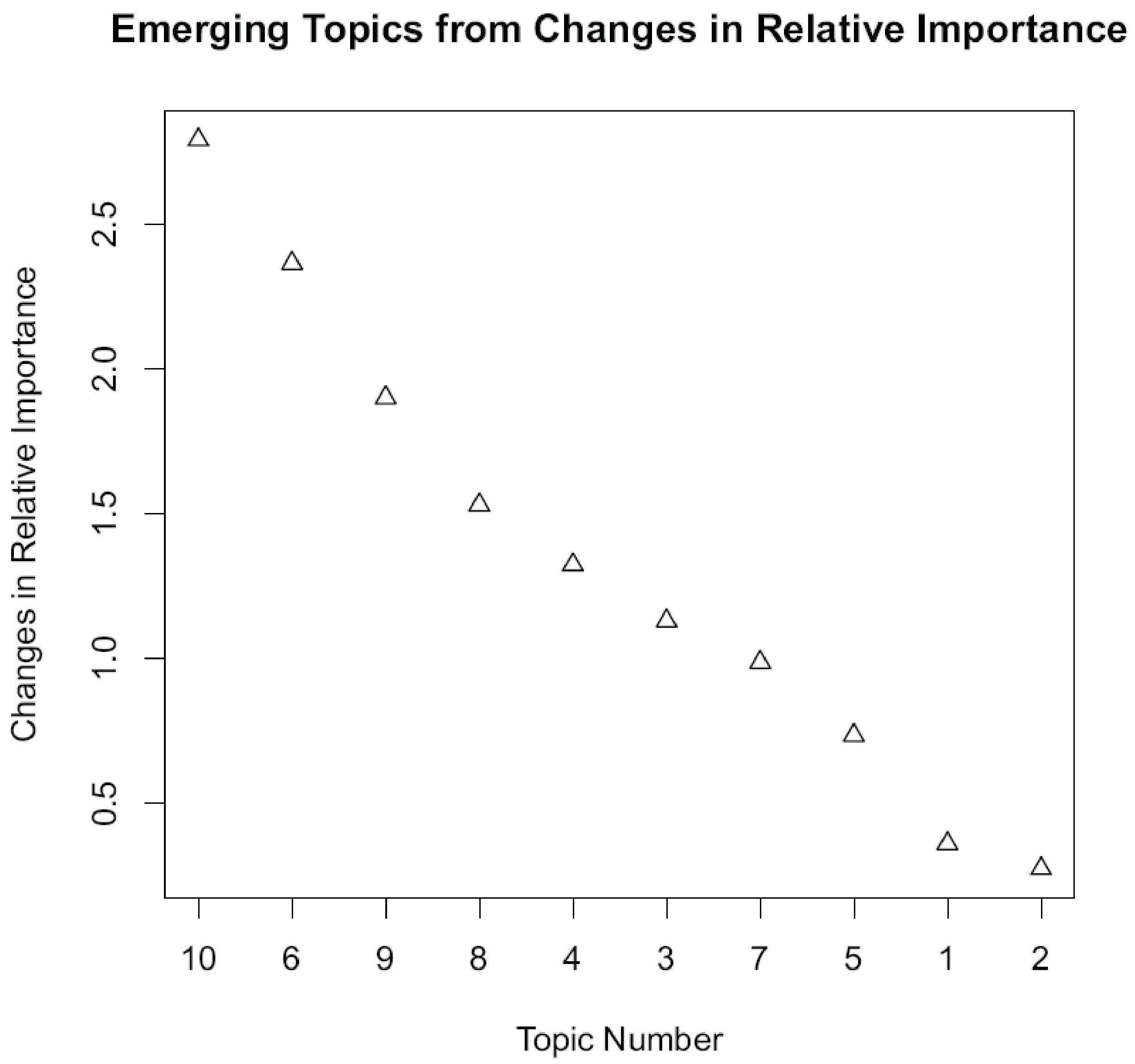
```
points(1:k, topic.importance.2.sorted1, col="red", pch=1)
legend("topright", legend=c("Training Set", "Testing Set"),
text.col=c("black","red"), pch=c(2,1), col=c("black","red"))

# Look at emerging topics.


# Method 1:  Look at absolute changes in relative importance.

importance.changes = topic.importance.2.sorted1 / topic.importance.1.sorted1$x

importance.changes.sort = sort(importance.changes, decreasing = TRUE, index.return = TRUE)

plot(1:k, importance.changes.sort$x, col="black",
xaxt = "n",
main = "Emerging Topics from Changes in Relative Importance",
                        ylab="Changes in Relative Importance",
                        pch=2, xlab = "Topic Number")

axis(side = 1, at=1:k, labels = importance.changes.sort$ix, xlab="Topic Number")
```

We discuss an example to make this picture clear. We collected some character merchandise sales reports and ran our algorithm over it in order to find the latest consumer preference. We have some idea that the new trend is reflected in the last 10% (recent) of the reports. We assumed that there are 10 topics. Here are the LDA results (first half of our algorithm):

| Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Star Wars |
|---|---|---|---|---|---|
| Jedi | Star | Details | War | Star | Mouse Family |
| Hoodies | Bowtique | Mouse | Jedi | Custom | |
| Price | Details | Gifts | Bags | Photo | |
| Bottles | Jacket | Jedi | Cards | Jedi | |
| Canvas | Product | Product | Retail | Approx | |

| Topic 6 | Topic 7 | Topic 8 | Topic 9 | Topic 10 |
|---|---|---|---|---|
| Gifts | Trick | Lightsaber | Details | Jacket |
| Stickers | Shirt | Product | Jacket | Land |
| Tshirt | Everminnie | Site | Dark | Force |
| Demothis | Jer | Jacket | Droid | Evermickey |
| Mickey | Star | Acceptance | Mickey | Everminnie |

Clearly, these are topics from the Star Wars and the Disney.
Here is the Relative importance graph which suggests emerging topics:

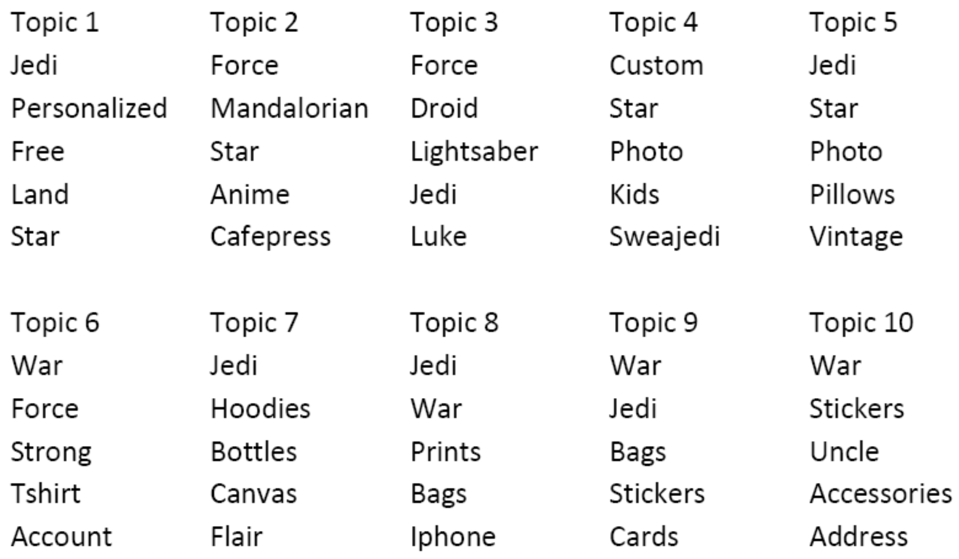


Here are the programs suggested emerging topics:

| Topic 10 | Topic 6 | Topic 9 | Topic 8 | Topic 4 | Topic 3 |
|---|---|---|---|---|---|
| Jacket | Gifts | Details | Lightsaber | War | Details |
| Land | Stickers | Jacket | Product | Jedi | Mouse |
| Force | Tshirt | Dark | Site | Bags | Gifts |
| Evermickey | Dempthis | Droid | Jacket | Cards | Jedi |
| Everminnie | Mickey | Mickey | Acceptance | Retail | Product |

Star Wars
Disney

Since our potential emerging topics are from both sales reports, we don't know which topic is our emerging topic. In order to find out the emerging topics, we took the first 90% of the sales reports (i.e., the documents excluding the recent set) and ran LDA over it. Here are the topics from the first 90% of the reports:



| Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 |
|---|---|---|---|---|
| Jedi | Force | Force | Custom | Jedi |
| Personalized | Mandalorian | Droid | Star | Star |
| Free | Star | Lightsaber | Photo | Photo |
| Land | Anime | Jedi | Kids | Pillows |
| Star | Cafepress | Luke | Sweajedi | Vintage |

| Topic 6 | Topic 7 | Topic 8 | Topic 9 | Topic 10 |
|---|---|---|---|---|
| War | Jedi | Jedi | War | War |
| Force | Hoodies | War | Jedi | Stickers |
| Strong | Bottles | Prints | Bags | Uncle |
| Tshirt | Canvas | Bags | Stickers | Accessories |
| Account | Flair | Iphone | Cards | Address |

As we can see, all of the topics are Star Wars topics. Now we compare these topics against the potential emerging topics from before, and we can discard Star Wars topics as the emerging topics. Since we have discarded Star Wars as emerging topics, the remaining topics, Topics 10, 6, and 3 are the emerging topics. That is, the Disney topics are our emerging topics. Hence, we conclude that our algorithm successfully detects emerging topic(s). This reflects that we were somewhat successful in achieving our goal of detecting emerging topics. However, in future we would like to work on defining the mathematical meaning of the relative importance ratio. Also currently, we have to try with different number of $K$ values, i.e., the number of topics and see what works the best. This is very time consuming and not an effective way of predicting the number of topics in a document (set). We would like to find a way to clearly define the value of $K$.

## 4. References

(1) D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993–1022, January 2003.

(2) Bettina Grün, Kurt Hornik, topicmodels: An R Package for Fitting Topic Models, Journal of Statistical Software, Volume 40, Issue 13, May 2011.