DTS 301TC - Assessment 2 Project Report

Jiayuan Zhu
1931097

1 T1

Start by reading the xlsx file by column type.

```
col_types <- c("text", "text", "text", "text", "text", "
    text", "numeric", "numeric", "numeric", "numeric", "
    numeric")
data <- read_excel("TwitterDataset.xlsx", col_types = col
    _types)
```

Then, by checking the data we found three columns, **Favs**, **RTs**, and **Followers**, have missing values, set the missing values to 0.

```
data <- data %>%
  mutate_at(c("Favs", "RTs", "Followers"), function(x)(
      replace_na(x,0)))
```

1.1 T1-1

We created a new column **SumFavsRTs** by adding **Favs** and **RTs**, and then ranked the tweets according to the descending order of the values of a. The top ten rows are the top ten tweets. The Figure 1 shows the top ten tweets, with some columns hidden due to the small size of the position.

```
RankTweets <- data %>%
  mutate(SumFavsRTs = Favs + RTs) %>%
  arrange(desc(SumFavsRTs))

RankTweets[1:10, ]
```

```
# A tibble: 10 × 12
   `Tweet Id`      Date   Hour  User …¹ Nickn…² Tweet…³  Favs   RTs Latit…⁴ Longi…⁵ Follo…⁶ SumFa…⁷
   <chr>           <chr>  <chr> <chr>   <chr>   <chr>   <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 721167832923… 2016… 02:46 "MALUM… maluma  "Y DEL…   2089   449    19.4   -99.1 2525716    2538
 2 721120295143… 2016… 23:37 "MALUM… maluma  "Hoy m…   2006   425    19.4   -99.1 2525565    2431
 3 720732972433… 2016… 21:58 "MALUM… maluma  "AGRAD…   1847   426    19.4   -99.1 2525716    2273
 4 721003913734… 2016… 15:55 "MALUM… maluma  "😇😈 …   1576   373    19.4   -99.1 2525716    1949
 5 720981520425… 2016… 14:26 "MALUM… maluma  "\"MED…   1550   313    19.4   -99.1 2525716    1863
 6 721133353911… 2016… 00:29 "Charl… charlo… "Do I …    853   499    34.1  -118.   349595    1352
 7 721025968278… 2016… 17:22 "Tanya… TanyaB… "I'm s…    477   103    33.8  -117.  1812679     580
 8 720561387030… 2016… 10:36 "MALUM… maluma  "Batma…    356    70    25.8   -80.3 2513119     426
 9 721154718161… 2016… 01:54 "Alfre… Alfred… "Ya bo…    243   153    33.7  -116.  2742081     396
10 721140633197… 2016… 00:58 "Fran\… _Frann… "Prom …    237   100    34.1  -118.      752     337
# … with abbreviated variable names ¹`User Name`, ²Nickname, ³`Tweet content`, ⁴Latitude,
#   ⁵Longitude, ⁶Followers, ⁷SumFavsRTs
```

Figure 1: Top Ten Tweets

1.2 T1-2

We started by selecting the three columns of **User Name**, **Nickname**, and **Followers**. We then removed the duplicate rows as each user may have posted more than one tweets, and ranked the users according to the reverse order of the values of **Followers**. The top ten rows are then taken to get the top ten users, as shown in Figure 2.

```
RankUsers <- data %>%
  select("User Name", "Nickname", "Followers") %>%
  filter(!duplicated(Nickname)) %>%
  arrange(desc(Followers))

RankUsers[1:10, ]
```

```
# A tibble: 10 × 3
   `User Name`       Nickname        Followers
   <chr>             <chr>               <dbl>
 1 WEREVERTUMORRO    werevertumorro    7232508
 2 Joel McHale       joelmchale        3804447
 3 Questlove Gomez   questlove         3627104
 4 Gabrielle Union   itsgabrielleu     3043036
 5 Alfredo Flores    AlfredoFlores     2742081
 6 MALUMA            maluma            2525716
 7 Matthew Ziff      MatthewZiff       2139467
 8 Fernanda Brum     PraFeBrum         2128396
 9 Maxene Magalona   maxenemagalona    2031651
10 CristiandelaFuente iamdelafuente    1965579
```

Figure 2: Top Ten Users

1.3 T1-3

Firstly, we get the Hour from the column **Date** and create a new column **Hour**, then we group all the rows according to the Hour and find the number of rows **Count** in each group. The Figure 3 shows the number of tweets posted in 24 hours.

```
TimeTweets <- data %>%
  mutate(Hour = str_split(Hour,":",simplify=T)[,1]) %>%
  group_by(Hour) %>%
  summarise(Count = n())

ggplot(TimeTweets) +
  geom_line(aes(x = Hour, y = Count, group = 1), color='
      #82B0D2') +
  geom_point(x = TimeTweets$Hour, y = TimeTweets$Count,
      color='#82B0D2') +
  labs(x = 'Hour',
       y = 'Number of Tweets',
       title = " ") +
  theme(plot.title = element_text(hjust = 0.5, size = 15)
      ,
        axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        axis.title = element_text(size = 15))
```

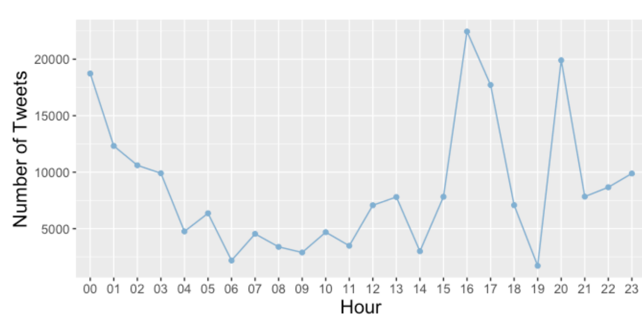

Figure 3: Number of Tweets Posted in a 24 hours

Project in DTS 301TC, Jiayuan Zhu, 1931097
2022.

1.4 T1-4

I'd like to draw a figure to show the number of tweets posted from different states in the US. Firstly, we need to find the state to which the coordinates of each tweet belong. Here, I used package *sf* to do it. Get states map from it and change it to a spatial dataframe, then spatial join with our coordinates.

```
# Get the states map, turn into sf object
US <- st_as_sf(map("state", plot = FALSE, fill = TRUE))

# Make it a spatial dataframe, using the same coordinate system as the US spatial dataframe
CoordsData <- st_as_sf(data, coords = c("Longitude", "Latitude"), crs = st_crs(US))

# Perform a spatial join
sf_use_s2(FALSE)
StateTweets <- st_join(CoordsData, US)
```

Secondly, group each tweet according to the states they belong to and count the number of tweets in each group to get column **Count**.

```
StateTweets <- StateTweets %>%
  drop_na(ID) %>%
  group_by(ID) %>%
  summarise(Count = n()) %>%
  arrange(ID)
```

Finally, the number of tweets is presented on a map of the US states, from light to dark indicating the number of tweets from most to least, as shown in Figure 4.

```
us <- map_data("state")

ggplot() + geom_map(data=us, map=us,
                    aes(x=long, y=lat, map_id=region),
                    fill="#ffffff", color="#ffffff", size=0.15) +
  geom_map(data=StateTweets, map=us,
           aes(fill=Count, map_id=ID),
           color="#ffffff", size=0.15) +
  scale_fill_continuous(low='thistle2',
                        high='darkred',
                        guide='colorbar') +
  labs(x=NULL, y=NULL) +
  theme(axis.ticks = element_blank()) +
  theme(axis.text = element_blank())
```

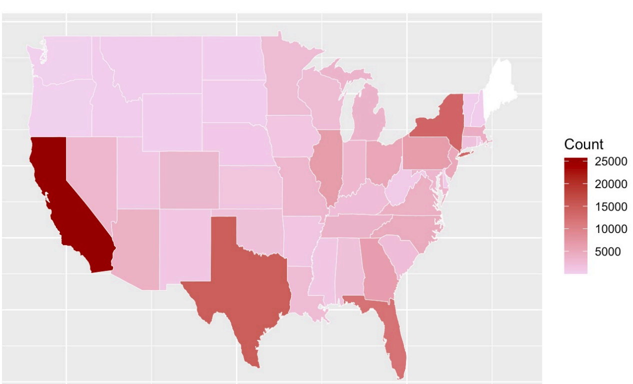

Figure 4: Number of Tweets Posted from Different States

2 T2

2.1 T2-1

To facilitate data mining, a function has been created for removing URLs, Emojis, mentions, hashtags and so on. Then it is applied on **Tweet content**. The Figure 5 shows some samples of tweet content

```
clean_tweets <- function(x) {
  x %>%
    # Remove URLs
    str_remove_all(" ?(f|ht)(tp)(s?)(://)(.*)[.|/](.*)")
      %>%
    # Remove Emojis
    str_remove_all('[:emoji:]') %>%
    # Remove mentions
    str_remove_all("@[[:alnum:]_]{4,}") %>%
    # Remove hashtags
    str_remove_all("#[[:alnum:]_]+") %>%
    # Replace "&" character reference with "and"
    str_replace_all("&amp;", "and") %>%
    # Remove puntucation
    str_remove_all("[[:punct:]]") %>%
    # Remove "RT: "
    str_remove_all("RT:? ") %>%
    # Replace any newline characters with a space
    str_replace_all("\\\n", " ") %>%
    # Make everything lowercase
    str_to_lower() %>%
    # Remove multiple spaces by single space
    str_squish()
}

d_clean <- data %>%
  mutate_at(c("Tweet content"), clean_tweets)

head(d_clean$`Tweet content`)
```

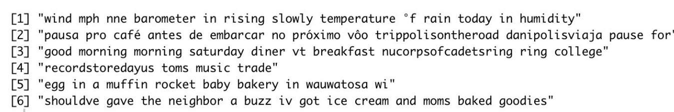
[1] "wind mph nne barometer in rising slowly temperature °f rain today in humidity"
[2] "pausa pro café antes de embarcar no próximo vôo trippolisontheroad danipolisviaja pause for"
[3] "good morning morning saturday diner vt breakfast nucorpsofcadetsring ring college"
[4] "recordstoredayus toms music trade"
[5] "egg in a muffin rocket baby bakery in wauwatosa wi"
[6] "shouldve gave the neighbor a buzz iv got ice cream and moms baked goodies"

Figure 5: Samples of tweet content

2.2 T2-2

I'm going to do topic modeling in T3, so I'm going to remove the numbers, split into tokens and remove stop words for column **Tweet content** next. Some samples of tokens are shown in Figure 6.

```
d_token <- d_clean %>%
  # Remove numbers
  mutate(`Tweet content` = gsub("[[:digit:]]", '', `Tweet content`)) %>%
  # Split into tokens
  unnest_tokens(word, `Tweet content`) %>%
  # Remove stop words
  anti_join(stop_words)
```

3 T3

3.1 Introduction

3.1.1 Background. Topic modelling is a technique designed to extract potential information from large datasets [1]. Topic models show enormous promise as a means of collecting potential insight

```
 [1] "wind"          "mph"                  "nne"                 "barometer"
 [5] "rising"        "slowly"               "temperature"         "rain"
 [9] "humidity"      "pausa"                "pro"                 "café"
[13] "antes"         "de"                   "embarcar"            "próximo"
[17] "vôo"           "trippolisontheroad"   "danipolisviaja"      "pause"
[21] "morning"       "morning"              "saturday"            "diner"
[25] "vt"            "breakfast"            "nucorpsofcadetsring" "ring"
[29] "college"       "recordstoredayus"     "toms"                "music"
[33] "trade"         "egg"                  "muffin"              "rocket"
[37] "baby"          "bakery"               "wauwatosa"           "wi"
[41] "shouldve"      "neighbor"             "buzz"                "iv"
[45] "ice"           "cream"                "moms"                "baked"
[49] "goodies"       "ct"
```

Figure 6: Samples of tokens

from the text data [10], bioinformatics data [6], social data [4], and environmental data [2]. There is a large amount of textual data on social media sites such as Twitter, and topical modelling of Twitter can yield potential information to better analyse the data, for example to establish public opinion on different issues. It is found that there is a link between the frequency of words in Twitter and polls on politics [7]. Analysing tweets over a period of time can provide insight into what is happening during that time, as people are inclined to tweet content that is pertinent to themselves and their surroundings [12].

3.1.2 Review. Papadimitriou et al. [8] first proposed a thematic model called Latent Semantic Analysis, which was later optimised as Probabilistic Latent Semantics Analysis. In 2010, Weng et al. [13] found that Latent Dirichlet Allocation (LDA) produced good results for topic modeling in tweets, and Sotiropoulos et al. [11] conducted a similar study using LDA on directed sentiment on topics related to two US telecommunications companies. With the development of deep learning in recent years, some neural network-based methods are also used for topic modeling [5]. Also, in topic modelling for social media, the analysis is not limited to the use of plain text messages sent by users. Qiu et al. [9] do topic modeling by analysing the user's behavioural information, namely "post", "retweet", "reply", and "mention". And Steinskog et al. [12] utilized aggregation of tweets sharing authors or hashtags.

3.2 Methodology

3.2.1 Model Section. Here, I have chosen Latent Dirichlet Allocation (LDA) for topic modelling. LDA is a model for latent semantic analysis based on Bayesian learning, proposed by Blei et al. in 2002 [3]. It is assumed that each text is represented by a multinomial distribution of topics, and each topic is represented by a multinomial distribution of words. in particular, it is assumed that the prior distribution of the topic distribution of the text is a Dirichlet distribution, and the prior distribution of the word distribution of the topic is also a Dirichlet distribution. the import of the prior distribution allows LDA to better cope with The introduction of the prior distribution enables LDA to better cope with the phenomenon of over-fitting in topic model learning.

3.2.2 Model Description. The process of generating a collection of texts for LDA is as follows: first, a random topic distribution is generated for a text, then a random topic is generated at each position in the text based on the topic distribution, then a random word is generated at that position based on the word distribution for that topic, and then the whole text is generated at the last position in the text, and then the process is repeated for all texts. LDA models cannot be solved directly with inference, and are usually solved using Gibbs Sampline and Variational EM Algorithm, the former being the Monte Carlo Method and the latter being the Approximation Algorithm.

The process of generating an LDA text set is as follows. Given a set of words W, a set of texts D, a set of topics Z, and hyper-parameters α and β of the Dirichlet distribution.

(1) Generating word distributions for topics.
Randomly generate word distributions for K topics. The specific procedure is as follows, following the Dirichlet distribution $Dir(\beta)$ randomly generates a parameter vector φ_k, $\varphi_k \sim Dir(\beta)$, as the word distribution of topic z_k, $p(w|z_k)$, $w \in W$, $k = 1, 2, ..., K$.
(2) Generating topic distributions for text.
Randomly generate topic distributions for M text. The specific procedure is as follows, following the Dirichlet distribution $Dir(\alpha)$ randomly generates a parameter vector θ_m, $\theta_m \sim Dir(\alpha)$, as the word distribution of topic $\mathrm{w_m}$, $p(z|\mathrm{w_m})$, $m = 1, 2, ..., M$.
(3) Generating word series for text.
Randomly generate N_m words for M texts. The word $w_{mn}(n = 1, 2, ..., N_m$ for text $\mathrm{w_m}(\mathrm{m} = 1, 2, ..., \mathrm{M})$ is generated as follows.
  (a) First according to the multinomial distribution $\mathrm{Mult}(\theta_\mathrm{m})$ generate a random topic z_{mn}, $z_{mn} \sim \mathrm{Mult}(\theta_\mathrm{m})$.
  (b) Then according to the multinomial distribution $\mathrm{Mult}(\varphi_{\mathrm{z_{mn}}})$ generate a random word w_{mn}, $w_{mn} \sim \mathrm{Mult}(\varphi_{\mathrm{z_{mn}}})$.

The we get the text $\mathrm{w_m}$, a sequence of words $\mathrm{w_m} = (\mathrm{w_{m1}}, \mathrm{w_{m2}}, ..., \mathrm{w_{mN_m}})$, corresponding to the implicit sequence of topics $\mathrm{z_m} = (\mathrm{z_{m1}}, \mathrm{z_{m2}}, ..., \mathrm{z_{mN_m}})$.

The learning of LDA, also known as parameter estimation, is a complex optimisation problem. It is difficult to solve exactly and can only be solved approximately. In the experiments I use Gibbs sampling, which has the advantage of being simple to implement and the disadvantage of having a potentially large number of iterations.

3.2.3 Process & Novelty. In the data mining process, we first removed some noise, namely words that appear less frequently. Then we found the most suitable number of topics K by tuning the hyper-parameters, and do LDA on the **words** to find the topics.

3.3 Experiments

Firstly, words that occur less than the *minimumFrequency* are removed to reduce noise. Then, in order to be able to use the data with function **LDA** in package **topicmodels**, the data needs to be transformed into DocumentTermMatrix types.

```r
# Set the minimum frequency of words
minimumFrequency <- 20
# Change to corpus
corpus <- VCorpus(VectorSource(d_token$word))
# Get DocumentTermMatrix
DTM <- DocumentTermMatrix(corpus,
                          control =
                          list(bounds =
                          list(global = c(
                              minimumFrequency, Inf))))
```

```
# Remove the duplicate
DTM = DTM[unique(DTM$i), ]
```

Next, I used function **FindTopicsNumber** in package **ladatuning** to find the best number of topic in this task. The Figure 7 shows the performances of choosing different number K of topics between 4 to 10. It indicates that the performance gets better as K increases and still tends to increase. However, if $K > 10$, the computational effort increases and the visualisation is not favourable. So in the next step we set $K = 10$.

```
# Find the best K
result <- ldatuning::FindTopicsNumber(
  DTM,
  topics = seq(from = 4, to = 12, by = 2),
  control = list(seed = 12345),
  verbose = TRUE
)

FindTopicsNumber_plot(result)
```

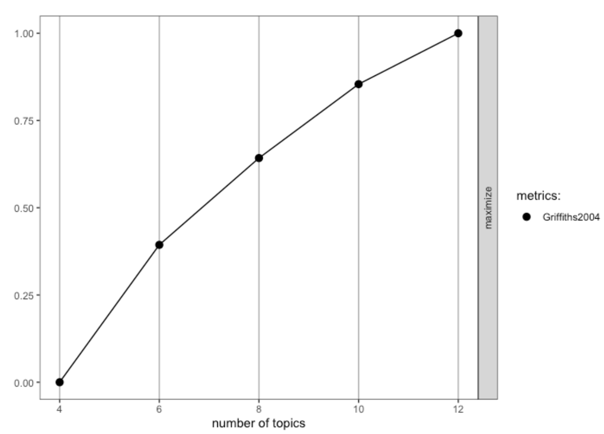

Figure 7: Top 15 Words of Each Topic

Next, the Gibbs Sampline and VEM methods of LDA were used to fit the data set respectively and get two models.

```
# Number of topics
K <- 10

set.seed(12345)
# Compute the LDA model, inference via 1000 iterations of
    Gibbs sampling
GibbsModel <- LDA(DTM, K, method="Gibbs", control=list(
    iter = 1000, verbose = 200))

# Compute the LDA model using VEM approach
VEMModel <- LDA(DTM, K, method="VEM")
```

3.4 Evaluation

To find the best model, we calculated the perplexity of each model.

Perplexity is a measure of how well a probability distribution or probability model predicts a sample. The lower the perplexity, the better the model clusters. The perplexity is calculated as follows

$$perplexity(D) = exp(-\frac{\sum \log p(w)}{\sum_{d=1}^{M} N_d})$$

	Gibbs Sampline	Variational EM
Perplexity	1816.72	2054.069

Table 1: Perplexity of Two Models

where $\sum_{d=1}^{M} N_d$ is the sum of all words in the test set, namely the total length of the test set. p(w) refers to the probability of occurrence of each word in the test set:

$$p(w) = p(z|d) * p(w|z)$$

where $p(z|d)$ represents the probability of occurrence of each topic in a document and $p(w|z)$ represents the probability of occurrence of each word in the dictionary under a topic.

```
perplexity(VEMModel, DTM)
perplexity(GibbsModel, DTM)
```

The result shown in Table 1 indicates that **GibbsModel** has the lower perplexity, meaning that it performed better. Thus, we used **GibbsModel** for the next steps.

Then, we calculate the words of each topic. For For clearer visualisation, we plot the 15 words with highest beta value of each topic in Figure 8.

```
# apply auto tidy using tidy and use beta as per-topic-
    per-word probabilities
topic <- tidy(GibbsModel,matrix = "beta")

# choose 15 words with highest beta from each topic
top_terms <- topic %>%
  group_by(topic) %>%
  top_n(15,beta) %>%
  ungroup() %>%
  arrange(topic,-beta)

# plot the topic and words for easy interpretation
plot_topic <- top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  theme(axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 5),
        axis.title = element_text(size = 15)) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip() +
  scale_x_reordered()

plot_topic
```

In order to see more clearly the connections between the different topics, we have generated a page with the results of the model clustering for analysis, as shown in Figure 9.

```
topicmodels2LDAvis <- function(x, ...){
  post <- posterior(x)
  if (ncol(post[["topics"]]) < 3) stop("The model must
      contain > 2 topics")
  mat <- x@wordassignments
  createJSON(
    phi = post[["terms"]],
    theta = post[["topics"]],
    vocab = colnames(post[["terms"]]),
    doc.length = slam::row_sums(mat, na.rm = TRUE),
    term.frequency = slam::col_sums(mat, na.rm = TRUE)
  )
}
```

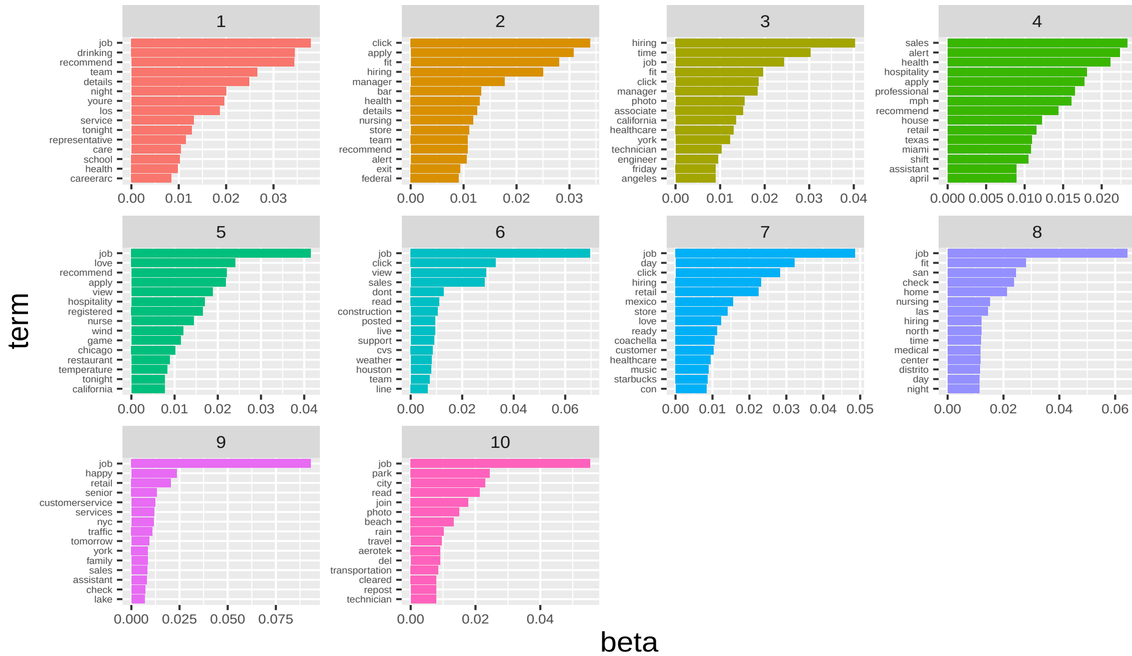

Figure 8: Top 15 Words of Each Topic

```

serVis(topicmodels2LDAvis(topicModel))
```

The visualisation results show that *job* is the most dominant topic in these topics amount the tweets, while the partitioning between different topics is not very obvious, indicating that the performance of the mod is bad and needs further improvement.

3.5 Conclusion

In this project, I used **LDA** for topic models for tweet dataset. However, the **LDA** has many shortcomings in topic modelling.

- The number of topics K is fixed that we need to tuning it for better performance. Also, as K increases, the consumption time increases.
- Topic distribution cannot capture correlations since it uses the Dirichlet distribution to model changes in topic proportions.
- The bag of words is used and the sentence structure information is not used.
- Most of the tweets are short texts, which have little semantic information and are sparse. **LDA** is not suitable it.

In order to improve the accuracy of topic modelling, structural information between sentences should also be used. Therefore, in the future there is a need to make more use of sentence structure information rather than just using the bag-of-words method.

REFERENCES

[1] David M. Blei. 2012. Probabilistic Topic Models. *Commun. ACM* 55, 4 (apr 2012), 77–84. https://doi.org/10.1145/2133806.2133826

[2] Yogesh Girdhar, Philippe Giguere, and Gregory Dudek. 2013. Autonomous adaptive underwater exploration using online topic modeling. In *Experimental Robotics*. Springer, 789–802.

[3] Tom Griffiths. 2002. Gibbs sampling in the generative model of latent dirichlet allocation. (2002).

[4] Liangjie Hong and Brian D Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*. 80–88.

[5] Suhyeon Kim, Haecheong Park, and Junghye Lee. 2020. Word2vec-based latent semantic analysis (W2V-LSA) for topic modeling: A study on blockchain technology trend analysis. *Expert Systems with Applications* 152 (2020), 113401.

[6] Lin Liu, Lin Tang, Wen Dong, Shaowen Yao, and Wei Zhou. 2016. An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus* 5, 1 (2016), 1–22.

[7] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Fourth international AAAI conference on weblogs and social media*.

[8] Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. 1998. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 159–168.

[9] Minghui Qiu, Feida Zhu, and Jing Jiang. 2013. It is not just what we say, but how we say them: Lda-based behavior-topic model. In *Proceedings of the 2013 SIAM international conference on data mining*. SIAM, 794–802.

[10] Daniel Ramage, Evan Rosen, Jason Chuang, Christopher D Manning, and Daniel A McFarland. [n. d.]. Topic Modeling for the Social Sciences. ([n. d.]), 4.

[11] Dionisios N Sotiropoulos, Chris D Kounavis, Panos Kourouthanassis, and George M Giaglis. 2014. What drives social sentiment? An entropic measure-based clustering approach towards identifying factors that influence social sentiment polarity. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*. IEEE, 361–373.

[12] Asbjørn Steinskog, Jonas Therkelsen, and Björn Gambäck. 2017. Twitter topic modeling by tweet aggregation. In *Proceedings of the 21st nordic conference on computational linguistics*. 77–86.

[13] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*. 261–270.

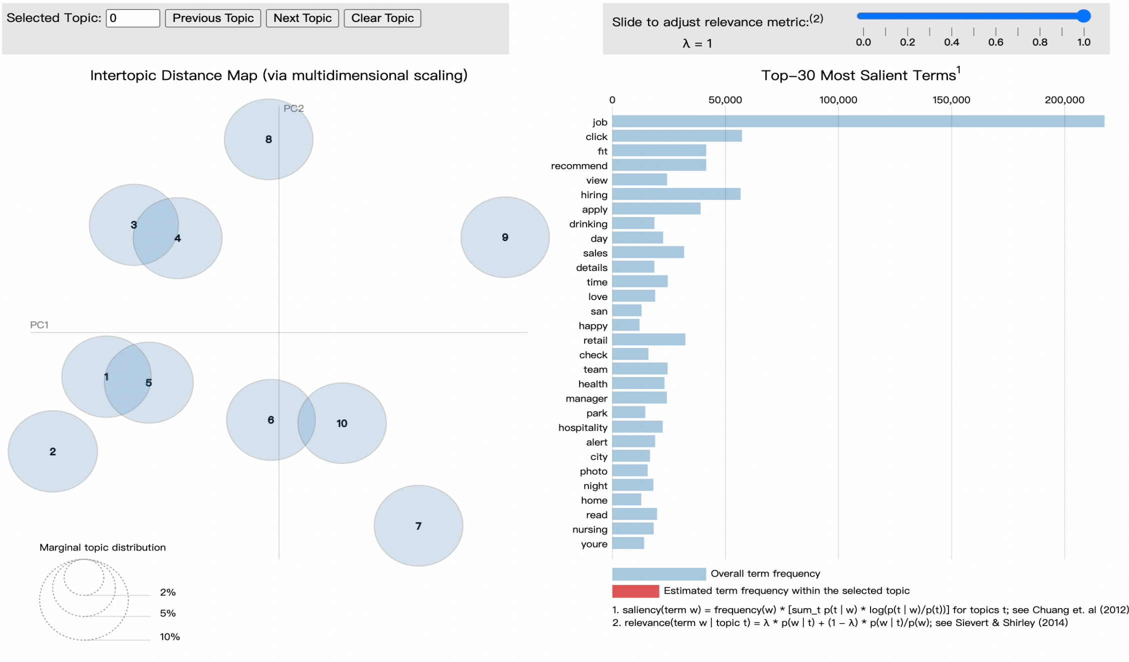

Figure 9: Topics