Assignment 10 by Team 3

Ashutosh Agarwal, Shun-Lung Chang, Pooja Natu

This study was conducted in R. The source code can be found here.

1. Convert the html to text files and separate the individual news items. The individual press release items serve as documents.

The data was retrieved from the Jacobs University press release archives, and then compiled into a dataframe with three variables:

- id: serial number for each press release
- text: title and content of each press release
- year: year of issue

```
colnames(text_df)
```

```
[1] "id"   "text" "year"
```

```
text_df[1:5, 'text']
```

```
# A tibble: 5 x 1
                                                                        text
                                                                       <chr>
1 Wissenschaft jenseits von Science Fiction: Jacobs University beteiligt sich
2 Sozialer Mehrwert durch Musik? Begleitstudie der Jacobs University zur Symb
3 Leibniz-Preis für Jacobs-Professorin Antje Boetius Dec , Antje Boetius, sei
4 Neuer Förderpreis der Stiftung Mercator für Studierende der Jacobs Universi
5 Management mit Zukunft: TiasNimbas Business School und Jacobs University st
```

2. Remove stop words and perform stemming.

Stop words are words used for grammer, but with little meaning. Common stop words are articles, such as 'the' and 'a', and prepositions, such as 'in', 'at', etc. Stemming refers to the process in which a word is reduced to its word stem. For example, 'goes', 'went', and 'gone' are reduced to 'go'. By doing so, a word with various forms will not be considered different words. In R, stop words can be discarded by `tokenizers::stopwords()`. Stemming can be carried out by `SnowballC::wordStem()`. The results are shown below.

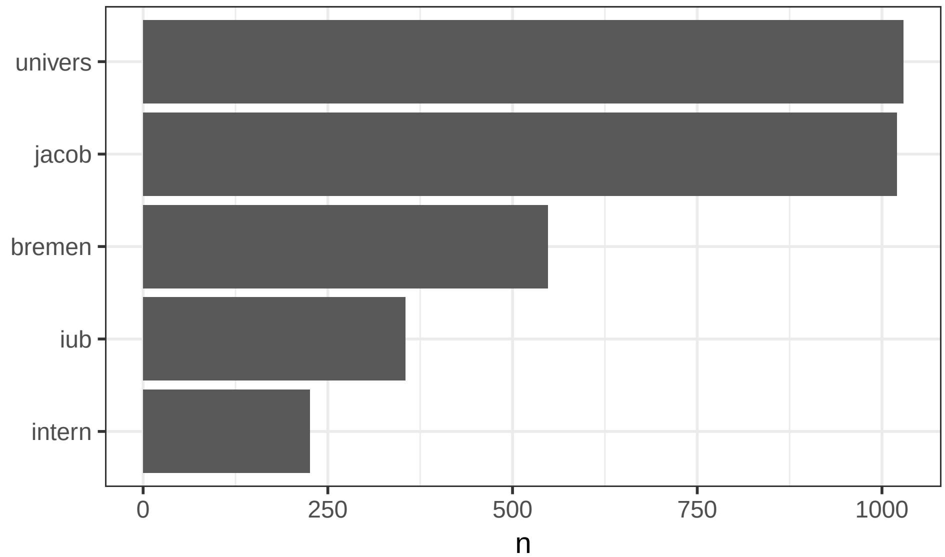
```
t <- text_df %>%
    unnest_tokens(word, text) %>%
    anti_join(tibble(word = c(stopwords("de"), stopwords("en"))), by = "word") %>%
    mutate(stemmed_word = wordStem(word))
```

id	year	word	stemmed_word
1	2008	wissenschaft	wissenschaft
1	2008	jenseits	jenseit
1	2008	science	scienc
1	2008	fiction	fiction
1	2008	jacobs	jacob
1	2008	university	univers

3. Perform a frequency analysis to compute the term-document (TD) matrix. What are the most common terms?

The frequency analysis can be performed by grouping each word (`dplyr::group_by()`) and by counting its occurrences (`dplyr::count()`). The following plots indicates that the most frequent three words are 'univers', 'jacob', and 'bremen'.

```
top_5_words <- t %>%
    group_by(stemmed_word) %>%
    count(sort = TRUE) %>%
    ungroup() %>%
    slice(1:5)
```



Furthermore, the term-document matrix below shows that in the first ten press releases, the three most common terms did appear several times.

```
word_counts <- t %>%
    group_by(id, stemmed_word) %>%
    count() %>%
    arrange(id, -n) %>%
    ungroup()

td <- word_counts %>% spread(stemmed_word, n, fill = 0) %>%
    select(-id) %>%
    as.matrix()
```

	univers	jacob	bremen	iub	intern
1	3	3	0	0	0
2	2	2	5	0	0
3	1	2	2	0	0
4	1	2	0	0	1
5	2	2	2	0	0
6	1	1	1	0	0
7	2	3	1	0	0
8	2	1	1	0	0
9	4	3	1	0	0

	univers	jacob	bremen	iub	intern
10	3	3	1	0	0

4. Compute inverse-document frequency (IDF) and term importance (TI). What are now the most common terms?

In text mining, the term importance (or term frequency-inverse document frequency, TF-IDF hereafter) measures a word's saliency. Firstly, the term frequency (TF hereafter) measures a term's occurrence in a document. The definition is:

$TFij = \frac{Nij}{Ni}$, where Nij is the number of occurrences of word j in document i, and Ni is the number of words in document i.

Also, inverse document frequency (IDF hereafter) measures whether a word is common across all documents. It is defined as:

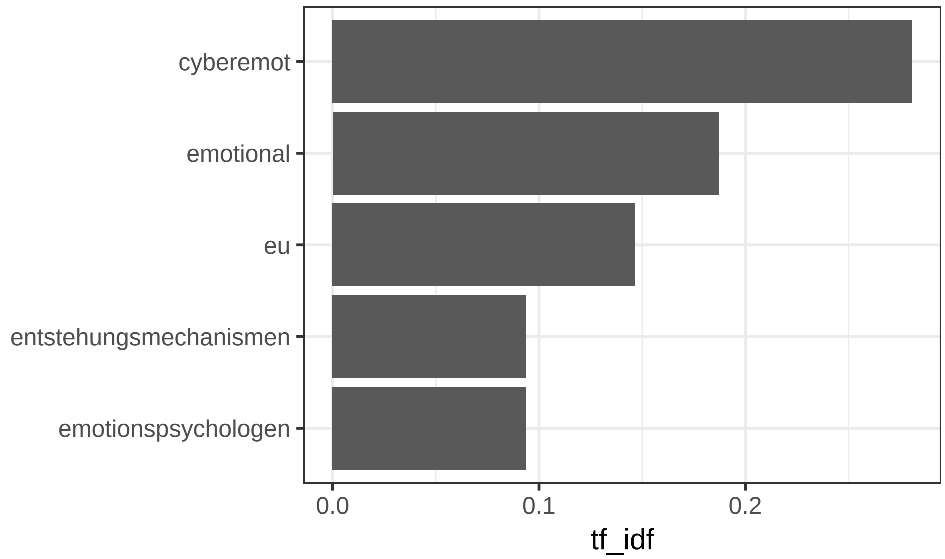
$IDFj = ln(\frac{N}{Nj})$, where N is the number of all documents, and Nj is the number of word j's occurrences in all documents.

TF-IDF is the product of term frequency and inverse document frequency. Of great importance to note is that, a common word across the document will obtained a lower IDF, thus gaining a lower term importance. Therefore, the rationale for calculating TF-IDF is that if a word is rare in the corpus, it implies a greater importance in a document. Inverse-document frequency and term importance in each press release can be acquired by `tidytext::bind_tf_idf()`. The following table shows that 'univers' gained low importance, which resulted from its ubiquitousness in the whole corpus.

```r
tf_idf <- word_counts %>%
    bind_tf_idf(term = stemmed_word, document = id, n = n)
```

id	stemmed_word	n	tf	idf	tf_idf
1	cyberemot	3	0.0434783	6.4583383	0.2807973
1	eu	3	0.0434783	3.3672958	0.1464042
1	jacob	3	0.0434783	0.3490907	0.0151779
1	univers	3	0.0434783	0.1022306	0.0044448
1	emotional	2	0.0289855	6.4583383	0.1871982
1	projekt	2	0.0289855	2.4693542	0.0715755

A closer look at the term importance of the first document reveals that important terms are surrounding the topics of cyber and emotion, and the press title: "Wissenschaft jenseits von Science Fiction: Jacobs University beteiligt sich am CyberEmotions-Project der EU", also shows it is an article regarding emotions in robotics.



5. Compute pairwise cosine and Euclidean distance between all documents.

Applying `text2vec::dist2` on the term-document matrix can assist us in obtaining pairwise similarities (measured by cosine or euclidean distance) of documents.

```
cos_dist <- dist2(td, method = 'cosine')
euc_dist <- dist2(td, method = 'euclidean')
```

The following are cosine distance matrix and euclidean distance matrix, respectively. Both of them show the first 3 x 3 submatrix. The diagonal elements are all 0 since they are comparisons of identical documents. And lower off-diagonal values suggest more similar documents.

```
cos_dist[1:3, 1:3] %>% kable()
```

1	2	3
0.0000000	0.8578515	0.8455115
0.8578515	0.0000000	0.7659177
0.8455115	0.7659177	0.0000000

```
euc_dist[1:3, 1:3] %>% kable()
```

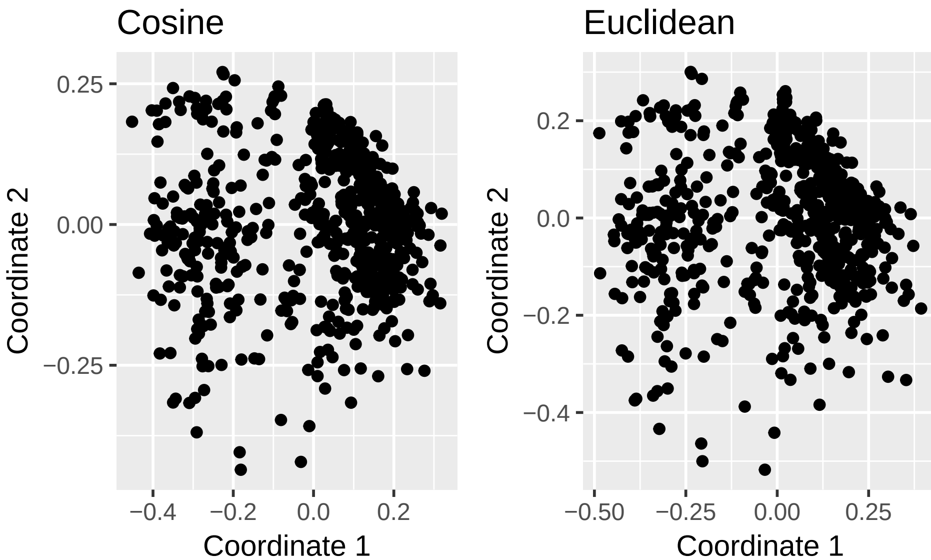
0.000000	1.309848	1.300393
1.309848	0.000000	1.237673
1.300393	1.237673	0.000000

6. Apply a multi-dimensional scaling approach to the distance matrix and render a 2D scatterplot. Compare the two distance metrics.

A multi-dimensional scaling approach transforms a high-dimentional matrix into a low-dimentional one. To approach this task, we applied the classical multi-dimensional scaling (by `cmdscale()`) on the two matrices obtained in task 5.

```r
cos_dist_fit <- cmdscale(cos_dist, k = 2)
euc_dist_fit <- cmdscale(euc_dist, k = 2)
```

As can be seen from the scatterplots below, a cluster appear in the left-upper quadrant of both plots. Given that most topics of the press were unrelated, those dissimilar pairs in turn formed the cluster in the two plots.



7. Capture the year of release during parsing and color code the scatterplot by time. Produce a Word Cloud for each year.

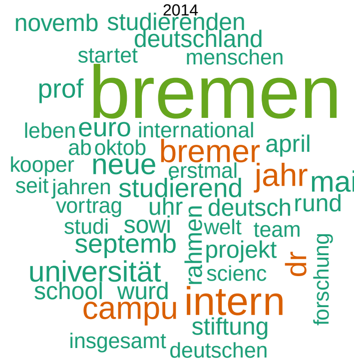
To create a word cloud for each year, we constructed a function `create_wordcloud()`. Take 2013 to 2015 for example, one can see that the most common terms in the clouds are 'univers', 'jacob' and 'Bremen', which are already pointed out in task 3.

```r
create_wordcloud <- function(year) {

    d <- t %>%
        filter(year == year) %>%
        group_by(stemmed_word) %>%
        count() %>%
        ungroup()

    wordcloud(words = d$stemmed_word, freq = d$n,
              max.words = 50, colors = brewer.pal(8, "Dark2"))
    text(x = 0.5, y = 1, cex = 0.5, as.character(year))
}
```

```r
for (i in seq(2013, 2015)) {
    create_wordcloud(i)
}
```

2013

2014



2015