209041841_GY7708_CW2

209041841

08/05/2021

GY7708 Coursework-2

Geospatial information processing project report

install webshot to convert the map to image

```
webshot::install_phantomjs()
```

Load the necessary libraries into the R environment

```
library(stopwords)
library(tokenizers)
library(SnowballC)
library(WikipediR)
library(wordcloud)
library(tidyverse)
library(readr)
library(dplyr)
library(tidytext)
library(rvest)
library(farver)
library(magrittr)
library(jsonlite)
library(httr)
library(textdata)
library(ggraph)
library(igraph)
library(leaflet.providers)
library(mapview)
library(sf)
```

Part - 1 : Read the data and retrive the summary of the wikipedia article associated with Newham, London Borough.

Read the CSV data into R.

Data : wikipedia geotags in greater london borough's

```
# Read CSV file
Wiki_lon <- readr::read_csv("wikipedia_geotags_in_GreaterLondon.csv")
```

Extract the London borough which is assigned.

```
# Extract Newham wikipedia geotags
Newham_LAD <-
  Wiki_lon %>%
  # filter newham and primary geo tag
  filter(lad_name == "Newham" & gt_primary == 1) %>%
  # mutate line number, to join the text words
  mutate(line = 1:305)
head(Newham_LAD, n=5)
```

```
## # A tibble: 5 x 26
##   lad_name  gt_id gt_page_id gt_globe gt_primary gt_lat gt_lon gt_dim gt_type
##   <chr>     <dbl>      <dbl> <chr>         <dbl>  <dbl>  <dbl>  <dbl> <chr>
## 1 Newham   4.29e8   59349030 earth             1   51.5 0.0221   1000 NULL
## 2 Newham   4.87e8   56750465 earth             1   51.5 0.0378    625 NULL
## 3 Newham   5.45e8   53080812 earth             1   51.5 0.0239   1000 landma~
## 4 Newham   5.45e8   47022399 earth             1   51.5 0.0324   1000 NULL
## 5 Newham   5.71e8   52754913 earth             1   51.5 0.0001    500 NULL
## # ... with 17 more variables: gt_name <chr>, gt_country <chr>, gt_region <chr>,
## #   page_id <dbl>, page_namespace <dbl>, page_title <chr>,
## #   page_restrictions <lgl>, page_is_redirect <dbl>, page_is_new <dbl>,
## #   page_random <dbl>, page_touched <dbl>, page_links_updated <dbl>,
## #   page_latest <dbl>, page_len <dbl>, page_content_model <chr>,
## #   page_lang <chr>, line <int>
```

Retrive the wikipedia summary for each page title in Newham geotags.

Retrive the summary of the wikipedia geotags by the page_title

```
# Create a list
newham_tag <- list()
for (i in Newham_LAD$page_title){

# Retrieve the summary
a_page_summary <-
  httr::GET(
    # Base API URL
    url = "https://en.wikipedia.org/w/api.php",
    # API query definition
    query = list(
      # Use JSON data format
      format = "json",
      action = "query",
      # Only retrieve the intro
      prop = "extracts",
      exintro = 1,
      explaintext = 1,
      redirects = 1,
      # Set the title
      titles = i
      )
    )%>%
```

```
  # Get the content
  httr::content(
    as = "text",
    encoding = "UTF-8"
    ) %>%
  # Trasnform JSON content to R list
  jsonlite::fromJSON() %>%
  # Extract the summary from the list
  magrittr::extract2("query") %>%
  magrittr::extract2("pages") %>%
  magrittr::extract2(1) %>%
  magrittr::extract2("extract")

newham_tag <- append(newham_tag, list(a_page_summary))
}
```

We have retrived the 305 page summary for the geotags, tagged in Newham

```
# Display the data
head(newham_tag, 2)
```

```
## [[1]]
## [1] "The Forest Gate School of Music (later the Metropolitan Academy of Music) was established in 188
##
## [[2]]
## [1] "Green Street House, usually known as Boleyn Castle, was a stately home in East Ham in the moder
```

Create the dataframe for the newham tag

Creating the document matrix dataframe, will store the summary of the title

```
df <- data.frame(matrix(unlist(newham_tag), nrow=length(newham_tag),
                        byrow=TRUE), stringsAsFactors = FALSE)
```

```
# Rename the column name to summary
newham_df <- rename(
  df, "summary" = "matrix.unlist.newham_tag...nrow...length.newham_tag...byrow...TRUE.") %>%
  # mutate line number to join the two table
    mutate(line = 1:305)
```

The line number will help us to join the data with the stem words.

Join the tables

```
# Join Newham_LAD and summary dataframe tables
summary_tag <-
  Newham_LAD %>%
  # Join by Line
  inner_join(newham_df)
# Display head
head(summary_tag, 5)
```

```
## # A tibble: 5 x 27
##   lad_name  gt_id gt_page_id gt_globe gt_primary gt_lat gt_lon gt_dim gt_type
```

```
##   <chr>     <dbl>      <dbl> <chr>         <dbl>  <dbl>  <dbl>  <dbl> <chr>
## 1 Newham   4.29e8   59349030 earth             1   51.5 0.0221   1000 NULL
## 2 Newham   4.87e8   56750465 earth             1   51.5 0.0378    625 NULL
## 3 Newham   5.45e8   53080812 earth             1   51.5 0.0239   1000 landma~
## 4 Newham   5.45e8   47022399 earth             1   51.5 0.0324   1000 NULL
## 5 Newham   5.71e8   52754913 earth             1   51.5 0.0001    500 NULL
## # ... with 18 more variables: gt_name <chr>, gt_country <chr>, gt_region <chr>,
## #   page_id <dbl>, page_namespace <dbl>, page_title <chr>,
## #   page_restrictions <lgl>, page_is_redirect <dbl>, page_is_new <dbl>,
## #   page_random <dbl>, page_touched <dbl>, page_links_updated <dbl>,
## #   page_latest <dbl>, page_len <dbl>, page_content_model <chr>,
## #   page_lang <chr>, line <int>, summary <chr>
```

Here, we have impported the summary of the page_title and incorporated with newham dataframe variable.

Part - 2: Spatial Frequency Analysis of retrived wikipedia summaries

Introduction

The London Borough of Newham was founded in 1965. The name Newham is a combination of the compass points of the old borough names, and it represents its origins. With a population of 353,134 inhabitants, Newham is situated on the border of inner and outer East London, making it 3rd most populous borough in London and the 20th highest crowded area in England. It is situated on the north bank of the Thames River, 5 miles east of the City of London. Newham, one of the six host boroughs for the 2012 Summer Olympics, is home to the most of all Olympic Park, such as London Stadium.

The counting of each letter's occurrence in a text is known as frequency analysis. Frequency analysis is based on the fact that different letters and combinations of letters appear with different frequencies in different parts of a text. Implementing tidy data concepts to make data handling easier and more efficient when working with text. As an outcome, tidy text is described as a table containing one token per row. Tokenization is the method of breaking down text into individual tokens, which we'd like to use for text analysis. The token stored in each row for tidy text mining is usually a separate word, However, this might be an n-gram, a phrase, or a paragraph. In the tidy text libraries, we provide functionality to tokenize by commonly used input text such as these and transform to a one-term-per-row format.

We will create the tibble dataframe using dpylr libraries with line number, which contains the text format datatype are not compatible with filter or count frequent words, we need to convert as the character as well as the one-token-per-document-per-row. We need to use the unnest token() functions to split the text into single tokens within our tidy text method. Unnest tokens() transforms every word to lower case by default. Stop words are words which aren't useful in an analysis and are typically very common in English. We can delete the stop words in the dataframe using the anti join() function. Stemming is a typical last step in text pre-processing. Stemming a word means changing it to its simplest conjugate form. Stemming is popular because we don't want the terms "established" and "establish" to have different definitions for the algorithms we'll be using to extract latent themes from unstructured texts. Finally, we will join the stemming words with the coordinates using inner_join() function. The next step in text mining is to measure the frequency of each word in Newham borough's Wikipedia tags and map the 20 most frequently tagged words in the borough. We can visualise 500 words using the wordcloud() function to visualise the most frequently tagged words in Newham. Very last, we'll use SF libraries to transform the words data frame into a spatial point format, and we'll use the mapview() feature to plot the words that are tagged in Newham spatio - temporal.

Now, start the text pre-processing like tokenization, removing stopwords, word stemming.

Text pre-processing the retrieved Wikipedia summaries

```
# Create a tibble
# mutate line number and convert the tagged text as character
tibble_t <-tibble(text = newham_tag,line =1:305)
# change text as character
tibble_t$text <- as.character(tibble_t$text)

tibble_t
```

```
## # A tibble: 305 x 2
##    text                                                                  line
##    <chr>                                                                <int>
##  1 "The Forest Gate School of Music (later the Metropolitan Academy of Mu~     1
##  2 "Green Street House, usually known as Boleyn Castle, was a stately hom~     2
##  3 "The Memorial Community Church is a Baptist church in Plaistow, Newham~     3
##  4 "Barclay Hall, E7 8JQ, is a 1900 building located in London's Green St~     4
##  5 "The Abbey Mill was a medieval tidal watermill in West Ham, London, da~     5
##  6 "Upton House was a building in West Ham, Essex (now the London Borough~     6
##  7 "St Mary's Church is a Church of England church in Plaistow, east Lond~     7
##  8 "St Philip and St James' Church is a Church of England church in Plais~     8
##  9 "Holy Trinity Church was a Church of England parish church in Canning ~     9
## 10 "St John the Baptist's Church, East Ham, was a Church of England churc~    10
## # ... with 295 more rows
```

Tokenization of words

```
data(stop_words)
# Seggregate the token words from text
token_words <- tibble_t %>%
  # unnest tokens separates the words
  unnest_tokens(word, text) %>%
  # Remove stopwords from text
  anti_join(stop_words)

token_words
```

```
## # A tibble: 20,032 x 2
##     line word
##    <int> <chr>
##  1     1 forest
##  2     1 gate
##  3     1 school
##  4     1 music
##  5     1 metropolitan
##  6     1 academy
##  7     1 music
##  8     1 established
##  9     1 1885
## 10     1 harding
## # ... with 20,022 more rows
```

Stemming words

```
# Stem the token words
stem_words <- token_words %>%
  # Mutate stem words
  mutate(stem = wordStem(word))

stem_words
```

```
## # A tibble: 20,032 x 3
##     line word         stem
##    <int> <chr>        <chr>
##  1     1 forest       forest
##  2     1 gate         gate
##  3     1 school       school
##  4     1 music        music
##  5     1 metropolitan metropolitan
##  6     1 academy      academi
##  7     1 music        music
##  8     1 established  establish
##  9     1 1885         1885
## 10     1 harding      hard
## # ... with 20,022 more rows
```

create the new table with page_title, type, longitute, latitude and line and join the stem words using inner_join()

```
# Create a new table with title, coordinates and words.
words_newham <-
summary_tag %>%
  dplyr::select(page_title,gt_type, gt_lon, gt_lat, line) %>%
  inner_join(stem_words)

words_newham
```

```
## # A tibble: 20,032 x 7
##    page_title                gt_type gt_lon gt_lat  line word        stem
##    <chr>                     <chr>    <dbl>  <dbl> <int> <chr>       <chr>
##  1 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 forest      forest
##  2 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 gate        gate
##  3 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 school      school
##  4 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 music       music
##  5 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 metropolit~ metropolit~
##  6 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 academy     academi
##  7 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 music       music
##  8 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 established establish
##  9 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 1885        1885
## 10 Forest_Gate_School_of_Mu~ NULL    0.0221   51.5     1 harding     hard
## # ... with 20,022 more rows
```

Let's, count the word frequency which tagged in Newham.

```
# Count stem words.
count_stem <- words_newham %>%
  count(stem, sort=TRUE)
```

```
count_stem
```

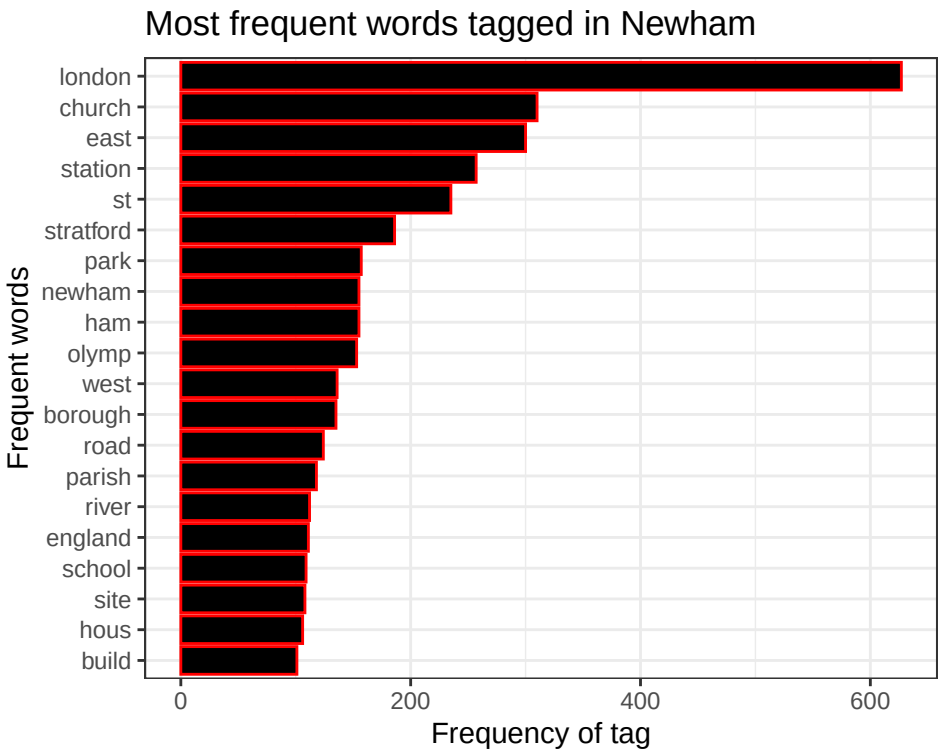
```
## # A tibble: 3,943 x 2
##    stem          n
##    <chr>     <int>
##  1 london      627
##  2 church      310
##  3 east        300
##  4 station     257
##  5 st          235
##  6 stratford   186
##  7 park        157
##  8 ham         155
##  9 newham      155
## 10 olymp       153
## # ... with 3,933 more rows
```

Frequency of words

```
# Calculate the frequency of top 20 words
freq <- count_stem %>%
  slice_max(n, n=20) %>%
  mutate(stem = reorder(stem,n))
```

Plot the frequency of words tagged

```
freq %>%
ggplot2::ggplot (
aes(
x = n, y = stem
)
) +
ggplot2::geom_bar(stat = "identity", fill="black", colour="red") +
ggplot2::ggtitle("Most frequent words tagged in Newham") +
ggplot2::xlab("Frequency of tag") +
ggplot2::ylab("Frequent words") +
ggplot2::theme_bw()
```



The most frequent tagged words are London, Church and east and so on.

Let's plot the word cloud for the frequent words tagged.

```
wordcloud(words = count_stem$stem, freq = count_stem$n, min.freq=20, max.words = 500,
          random.order = FALSE, scale = c(5, 0.5), colors=brewer.pal(6, "Dark2"))
```

Plotting the frequency of the words spatially

In this section we will plot the frequent words that are tagged in Newham spatio-temporal

Filter the top 3 frequent words that are tagged such as london, church and east.

```
# Filter the top 3 frequent words that are tagged in Newham
london_words <- words_newham %>%
  filter(stem == "london")
church_words <- words_newham %>%
  filter(stem == "church")
east_words <- words_newham %>%
  filter(stem == "east")
```

Convert the words into the spatial points inorder to plot the data in the map

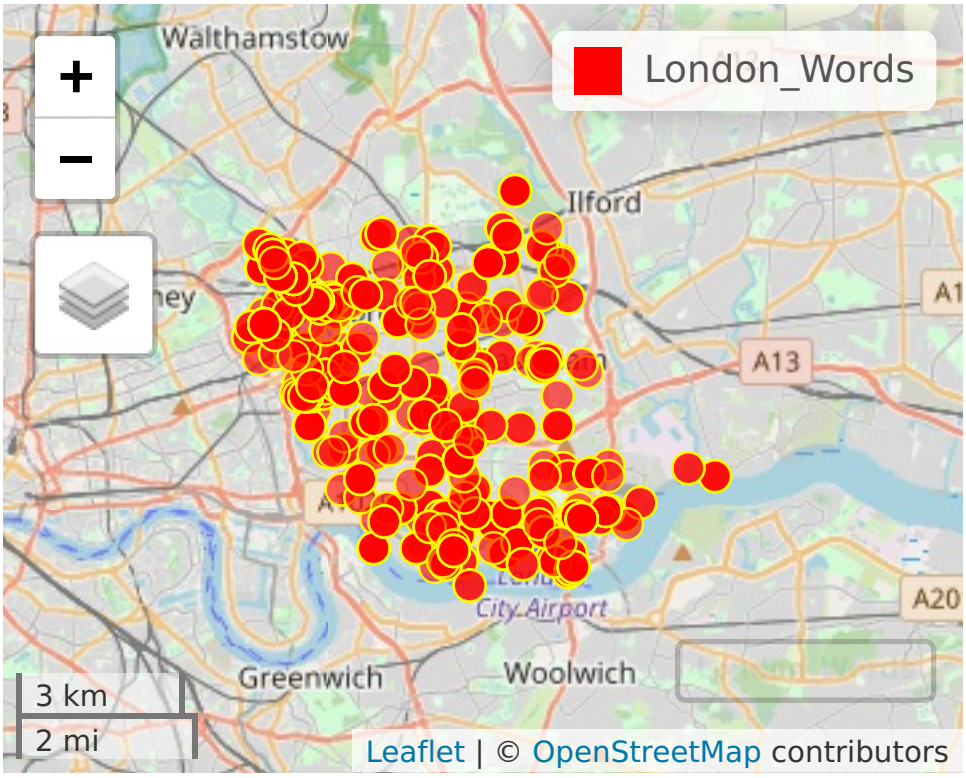
```
# create the spatial points for the words tagged with the coords
London_Words <- st_as_sf(london_words, coords = c("gt_lon","gt_lat")) %>%
  # project CRS as EPSG:4326
  st_set_crs("EPSG:4326") %>%
  # cast as point layer
  st_cast("POINT")

# create the spatial points for chruch words
Church_Words <- st_as_sf(church_words, coords = c("gt_lon","gt_lat")) %>%
  # project CRS as EPSG:4326
  st_set_crs("EPSG:4326") %>%
```

```
  # cast as point layer
  st_cast("POINT")

# create the spatial points for east words
East_Words <- st_as_sf(east_words, coords = c("gt_lon","gt_lat")) %>%
  # project CRS as EPSG:4326
  st_set_crs("EPSG:4326") %>%
  # cast as point layer
  st_cast("POINT")
```

Spatial plot

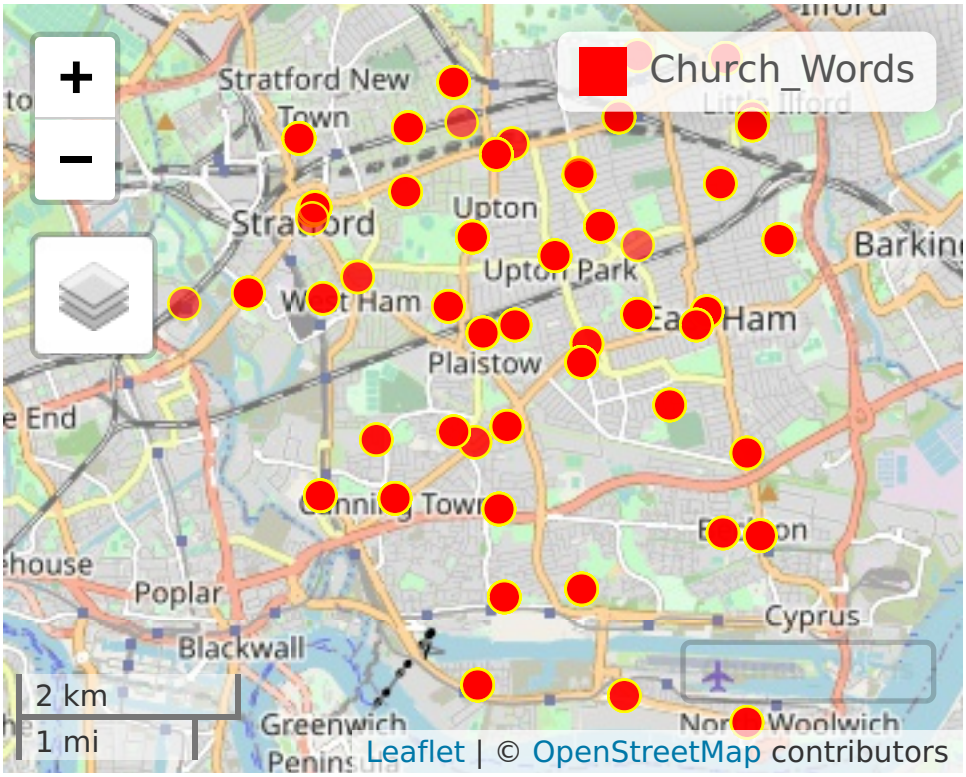
Let's visualize the words tagged location in interactive maps.

```
# plot the london words spatially in mapview using OSM
mapview(London_Words, map.type="OpenStreetMap", col.regions="red", color ="yellow")
```



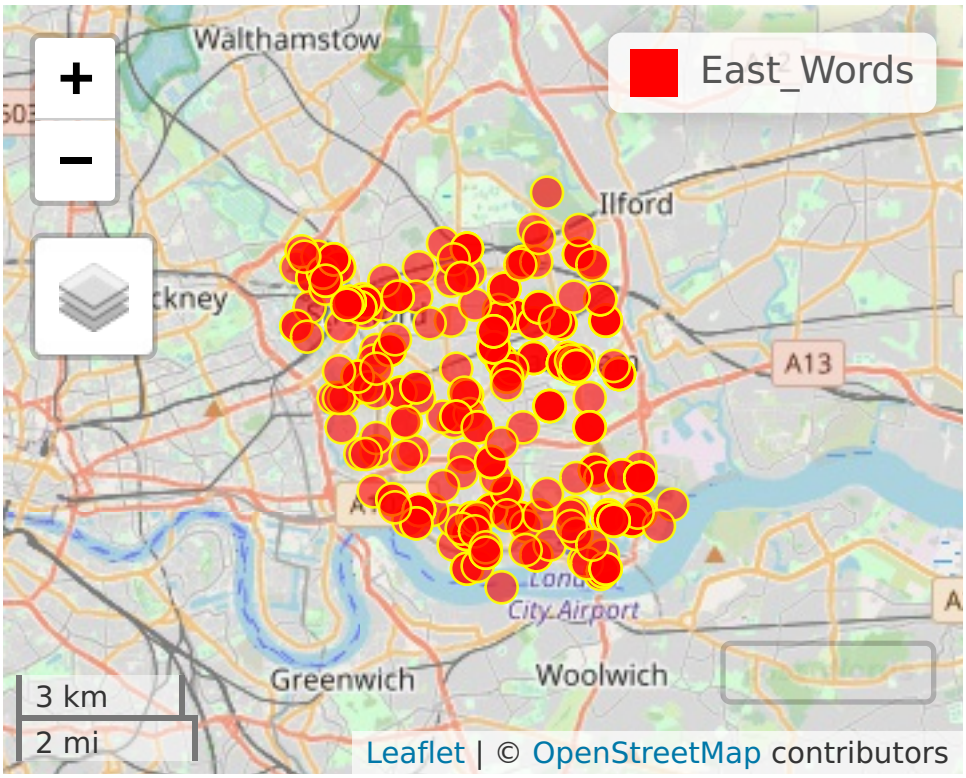
The word "London" tagged throught the Newham area.

```
# plot the church words spatially in mapview using OSM
mapview(Church_Words, map.type="OpenStreetMap", col.regions="red", color ="yellow")
```



The word "Church" were tagged in some parts of Newham

```
# plot the east words spatially in mapview using OSM
mapview(East_Words, map.type="OpenStreetMap", col.regions="red", color ="yellow")
```



The word “east” are scattered throught the regions

Discussion and Conclusion:

We were able to interpret the summary of Wikipedia tags into text with the use of the tibble dataframe and We split each row of the new data frame after using unnest tokens, so that each row contains one token word; unnest_tokens() will segregate default tokenization is single words. We were able to substitute conjugate words using the stemming function. We were able to interpret the spatial analysis of frequent words tagged in Newham by joining the stemming words table with a new table that contained page title, gt type, gt lon, gt lan, and line. We've now counted the number of words that were tagged, which indicates the frequency of all those words around Newham. To plot the frequency of the terms in a bar chart, we filtered out the top 20 words. The words London, Church, East, train, st, stratfort, and others words are tagged more in the Newham borough as a result of the bar plot. We were able to visualize the most commonly used keywords in a column of text using wordcloud methods, it's a representation of text data in a visual format. Finally, we used the mapview() function to plot the top three most frequently tagged terms spatially in the map. We categorized the top three words into individual variables and transformed them to spatial points. The words London, church, and east are tagged 627, 310, and 300, respectively, on the Open Street Map. All three words are scattered about the Newham areas, with the majority of them associated with the summer Olympics, tourist attractions, and churches.

We move on to sentiment analysis and more on n-gram analysis in next part.

Part - 3 : Sentiment Analysis of the frequant words

In data mining, sentiment evaluation is the method of studying individuals ‘s opinions emotions, and ways of thinking about a subject and categorizing their decisions into neutral, positive and negative categories. While we read messages, we employ our knowledge of the psychological content of words to determine if a portion of message is positive or negative, or maybe marked by a very complex sentiment like excitement or anger. Utilizing Natural Language Processing principles, sentiment evaluation is employed to identify renunciation inside the message. One strategy to analyse a message's sentiment analysis is to think about it as a collection of specific words, with the sentiment value of whole message equal to the amount of the specific words' sentiment value. Numerous emotion lexicons are available with the tidy text bundle. The general purpose lexicons are AFINN and Bing these two are based on unigram that is single word.

AFINN: The AFINN lexicon rates words from -5 to 5, with positive grades suggesting positive sentiment and negative grades suggesting negative sentiment.

Bing: The Bing lexicon divides terms between negative and positive groups in a binary manner.

n-gram:

We've begun tokenizing by sentence or word with the unnest tokens feature, that really is valuable for the emotion and intensity evaluations done so long. However, we may utilize the feature to classify into n-grams, which are collections of terms. We may create a simulation of the interactions around words X and Y by looking at how frequently they are accompanied by each other. A coherent series of n objects from a specific set of speech or message is known as an n-gram in the domains of probability and computational linguistics. Usually, n-grams were extracted through either a speech corpus or message. N-grams are indeed known as shingles only when objects are words. We achieve this by calling unnest tokens() with the token = “n-grams” variable and assigning n to the quantity of words we want to collect within every n-gram. Bigrams: Once n is matched to 2, we're looking at pairs of two terms that are commonly referred to as “bigrams”. Trigrams: Once n is fixed to 3, which are three-word sequences that are sometimes referred to as “trigrams”.

We may organise the words into a network, or graph, as one standard visual representation. We'll use the term “graph” to refer to a set of connected nodes rather than a visual representation. For controlling and

analysing networks, the igraph package has a number of useful features. The graph from data frame feature, that further accepts a data frame of edges with columns for "from," "to," and edge parameters, is one way to generate an igraph object from tidy data. We'll use the ggraph() function to plot the igraph results. Finally, we'll plot the sentiment values for the top tagged terms using AFINN values. The positive and negative values for the stem words will be separated and displayed using the mapview() function.

Bigram words Analysis

```
# extract the bigram words
bigram <- tibble_t %>%
  # unnest the tokens
  unnest_tokens(bigram, text, token="ngrams", n=2) %>%
  count(bigram, sort = TRUE)

bigram
```

```
## # A tibble: 20,111 x 2
##    bigram          n
##    <chr>       <int>
##  1 of the        523
##  2 in the        328
##  3 is a          194
##  4 the london    183
##  5 to the        161
##  6 east london   157
##  7 on the        157
##  8 it was        127
##  9 and the       116
## 10 it is         116
## # ... with 20,101 more rows
```

```
# Separate the bigram words into individual words
bigram_sep <- bigram %>%
  separate(bigram, c("word1", "word2"), sep = " ")
# Filter the bigram words
bigram_fil <- bigram_sep %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
#bigram counts
bigram_count <- bigram_fil %>%
  count(word1, word2, sort = TRUE)

bigram_count
```

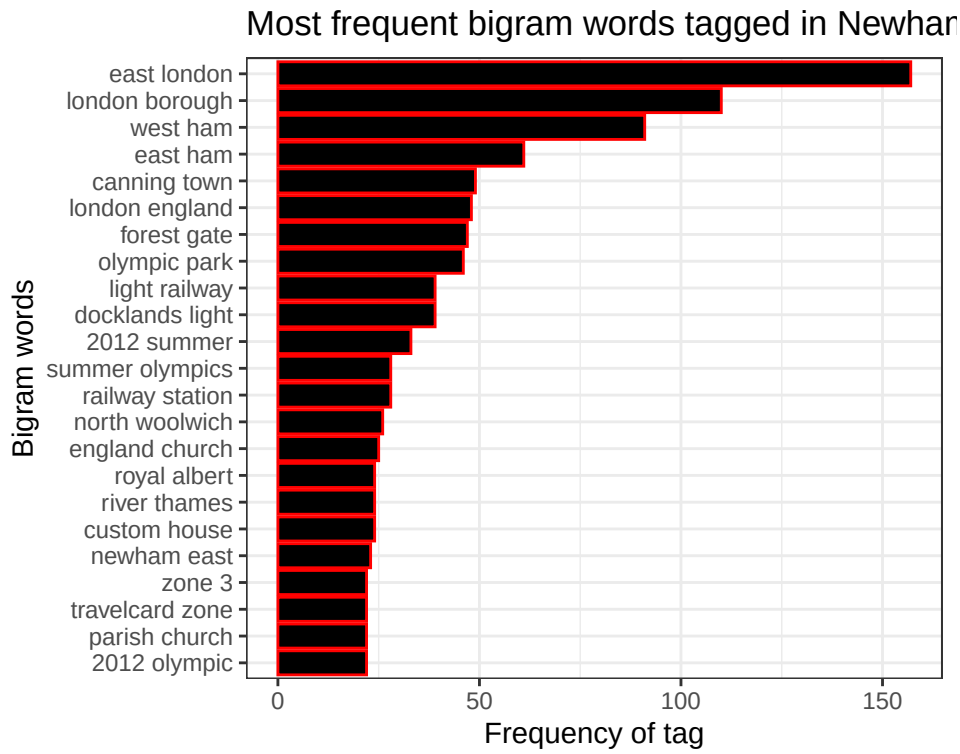
```
## # A tibble: 6,247 x 3
##    word1 word2        n
##    <chr> <chr>    <int>
##  1 0     03           1
##  2 0     victory      1
##  3 0.3   miles        1
##  4 0.40  km2          1
##  5 0.48  km           1
##  6 0.62  mile         1
##  7 0.7   mi           1
```

```
##  8 0.9   hectares     1
##  9 00    11           1
## 10 00    bst          1
## # ... with 6,237 more rows
```

```
# Join the bigram seprated words
bigram_uni <- bigram_fil %>%
  # join the bigram words
  unite(bigram, word1, word2, sep = " ")

bigram_uni
```

```
## # A tibble: 6,247 x 2
##    bigram              n
##    <chr>           <int>
##  1 east london       157
##  2 london borough    110
##  3 west ham           91
##  4 east ham           61
##  5 canning town       49
##  6 london england     48
##  7 forest gate        47
##  8 olympic park       46
##  9 docklands light    39
## 10 light railway      39
## # ... with 6,237 more rows
```

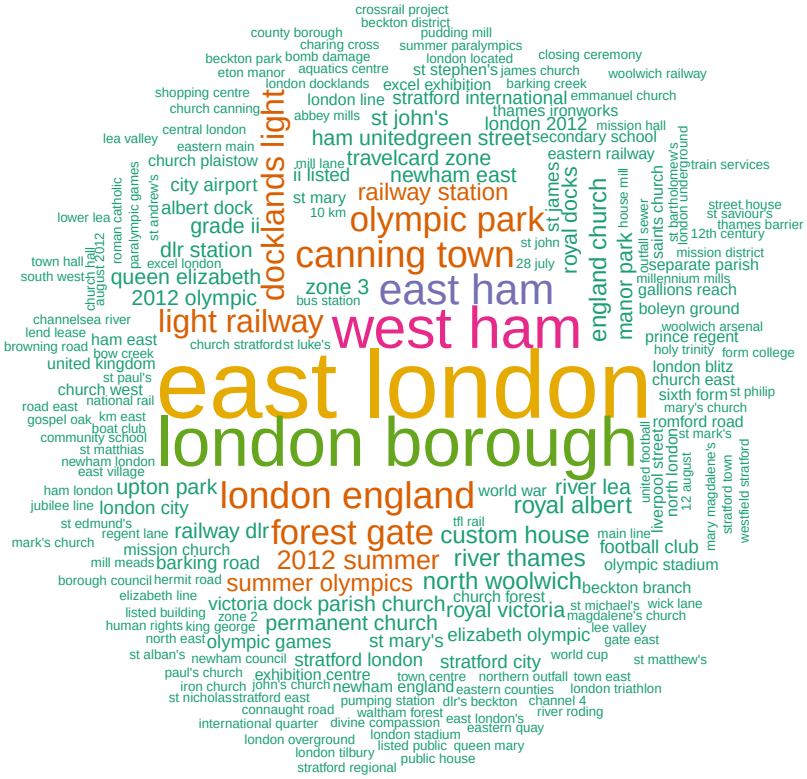
```
# plot the bigram words frequency
bigram_uni %>%
  slice_max(n, n=20) %>%
  mutate(bigram = reorder(bigram,n)) %>%
ggplot2::ggplot (
aes(
x = n, y = bigram
)
) +
ggplot2::geom_bar(stat = "identity", fill="black", colour="red") +
ggplot2::ggtitle("Most frequent bigram words tagged in Newham") +
ggplot2::xlab("Frequency of tag") +
ggplot2::ylab("Bigram words") +
ggplot2::theme_bw()
```



Most frequent bigram plotted are east london, london borough and west ham so on.

Let's plot the word cloud for bigram words.

```
wordcloud(words = bigram_uni$bigram, freq = bigram_uni$n, min.freq=5,
          max.words = 300, random.order = FALSE, scale = c(3, 0.3),
          colors=brewer.pal(6, "Dark2"))
```



Trigram Words Analysis

```
# seggregate the trigram words as separate word
trigram <- tibble_t %>%
  # unnest the tokens
  unnest_tokens(trigram, text, token="ngrams", n=3) %>%
  count(trigram, sort = TRUE)

trigram
```

```
## # A tibble: 29,182 x 2
##    trigram                    n
##    <chr>                  <int>
##  1 london borough of        105
##  2 borough of newham         97
##  3 the london borough        94
##  4 in the london             78
##  5 part of the               64
##  6 in east london            56
##  7 east london it            45
##  8 docklands light railway   39
##  9 church of england         31
## 10 a church of               30
## # ... with 29,172 more rows
```

```
# Seprate the trigram words
trigram_sep <- trigram %>%
  separate(trigram, c("word1", "word2", "word3"), sep = " ")
# Filter the trigram words
trigram_fil <- trigram_sep %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
         !word3 %in% stop_words$word)
# Count the words
trigram_count <- trigram_fil %>%
  count(word1, word2, word3, sort = TRUE)

trigram_count
```

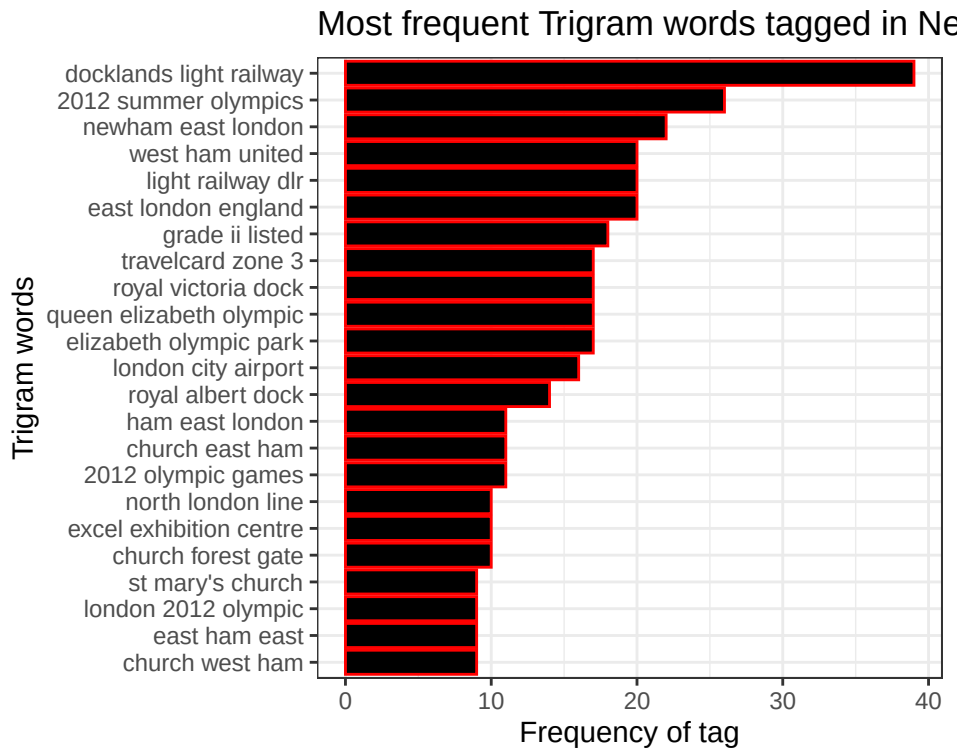
```
## # A tibble: 3,487 x 4
##    word1 word2   word3        n
##    <chr> <chr>   <chr>    <int>
##  1 0     03      42           1
##  2 0     03      46           1
##  3 0.3   miles   0.48         1
##  4 0.40  km2     site         1
##  5 0.62  mile    1.00         1
##  6 0.7   mi      concrete     1
##  7 00    11      lasting      1
##  8 00    bst     utc          1
##  9 1     chain   19.3         1
## 10 1     january 2000         1
## # ... with 3,477 more rows
```

```
# Unite the trigram
trigram_uni <- trigram_fil %>%
  #join the separated words
  unite(trigram, word1, word2, word3, sep = " ")

trigram_uni
```

```
## # A tibble: 3,487 x 2
##    trigram                      n
##    <chr>                    <int>
##  1 docklands light railway     39
##  2 2012 summer olympics        26
##  3 newham east london          22
##  4 east london england         20
##  5 light railway dlr           20
##  6 west ham united             20
##  7 grade ii listed             18
##  8 elizabeth olympic park      17
##  9 queen elizabeth olympic     17
## 10 royal victoria dock         17
## # ... with 3,477 more rows
```

```
# Plot the frequency of trigram words
trigram_uni %>%
  slice_max(n, n=20) %>%
  mutate(trigram = reorder(trigram,n)) %>%
```

```
ggplot2::ggplot (
aes(
x = n, y = trigram
)
) +
ggplot2::geom_bar(stat = "identity", fill="black", colour="red") +
ggplot2::ggtitle("Most frequent Trigram words tagged in Newham") +
ggplot2::xlab("Frequency of tag") +
ggplot2::ylab("Trigram words") +
ggplot2::theme_bw()
```



The most frequent trigram words tagged are docklands light railway, 2012 summer olympics and newham east london so on

Let's plot the wordcloud for the trigram

```
wordcloud(words = trigram_uni$trigram, freq = trigram_uni$n, min.freq=5,
          max.words = 200, random.order = FALSE, scale = c(2, 0.4),
          colors=brewer.pal(6, "Dark2"))
```

Sentiment Analysis

Bigram words sentiment analysis

```
# Filter negative words
bigram_sep %>%
  filter(word1 == "not" | word1 == "no" | word1 =="none" | word1 == "nothing" | word1 == "never") %>%
  count(word1, word2, sort = TRUE)
```

```
## # A tibble: 55 x 3
##    word1 word2          n
##    <chr> <chr>      <int>
##  1 never been           1
##  2 never came           1
##  3 never had            1
##  4 never likely         1
##  5 never reopened       1
##  6 no    00338672       1
##  7 no    allotment      1
##  8 no    charges        1
##  9 no    confirmed      1
## 10 no    connection     1
## # ... with 45 more rows
```

Import the afinn sentiment values.

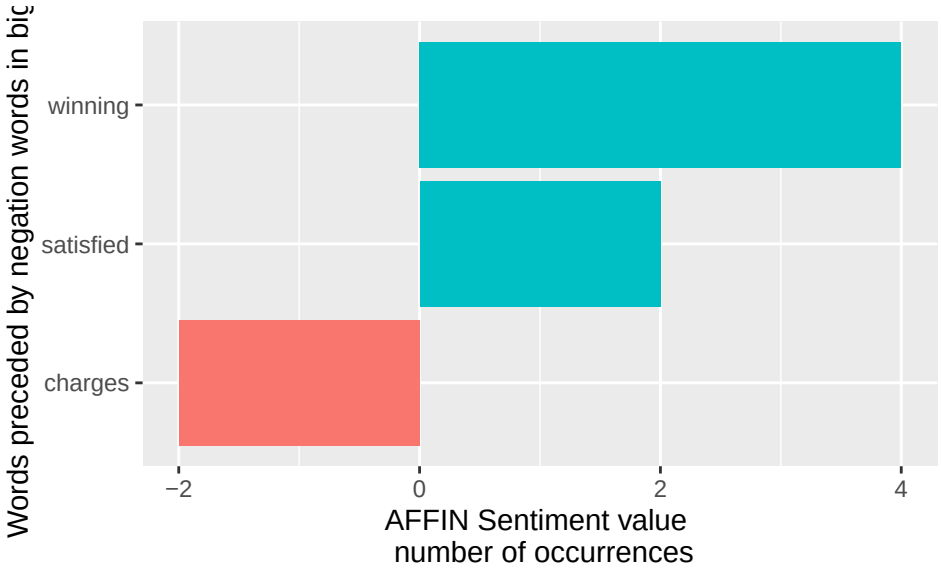
```r
# Import the afinn sentiment words value
AFINN <- get_sentiments("afinn")

AFINN
```

```
## # A tibble: 2,477 x 2
##    word       value
##    <chr>      <dbl>
##  1 abandon       -2
##  2 abandoned     -2
##  3 abandons      -2
##  4 abducted      -2
##  5 abduction     -2
##  6 abductions    -2
##  7 abhor         -3
##  8 abhorred      -3
##  9 abhorrent     -3
## 10 abhors        -3
## # ... with 2,467 more rows
```

```r
not_words <- bigram_sep %>%
  # filter not words
  filter(word1 == "not" | word1 == "no" | word1 =="none" | word1 == "nothing" | word1 == "never") %>%
  # inner join afinn words
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, value, sort = TRUE)

not_words
```

```
## # A tibble: 3 x 3
##   word2     value     n
##   <chr>     <dbl> <int>
## 1 charges      -2     1
## 2 satisfied     2     1
## 3 winning       4     1
```

```r
not_words %>%
  mutate(con = n * value) %>%
  arrange(desc(abs(con))) %>%
  mutate(word2 = reorder(word2, con)) %>%
  ggplot(aes(n * value, word2, fill = n * value > 0)) +
  geom_col(show.legend = FALSE) +
  labs(x = "AFFIN Sentiment value \n number of occurrences",
       y = "Words preceded by negation words in bigram")
```



The word winning, satisfied and charges are affilated to the AFINN values in the bigram words.

Let's analyse the sentiment by bing values

```
# Import the bing sentiment values
bing <- get_sentiments("bing")

bing
```

```
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # ... with 6,776 more rows
```

```
# Calculate the sentiment value for the bigram words
not_words1 <- bigram_sep %>%
  filter(word1 == "not" | word1 == "no" | word1 =="none" | word1 == "nothing" | word1 == "never") %>%
  inner_join(bing, by = c(word2 = "word")) %>%
  count(word2, sentiment, sort = TRUE)

not_words1
```

```
## # A tibble: 6 x 3
##   word2     sentiment     n
##   <chr>     <chr>     <int>
## 1 explosive negative      1
## 2 ideal     positive      1
```

```
## 3 objection negative      1
## 4 overlook  negative      1
## 5 satisfied positive      1
## 6 winning   positive      1
```

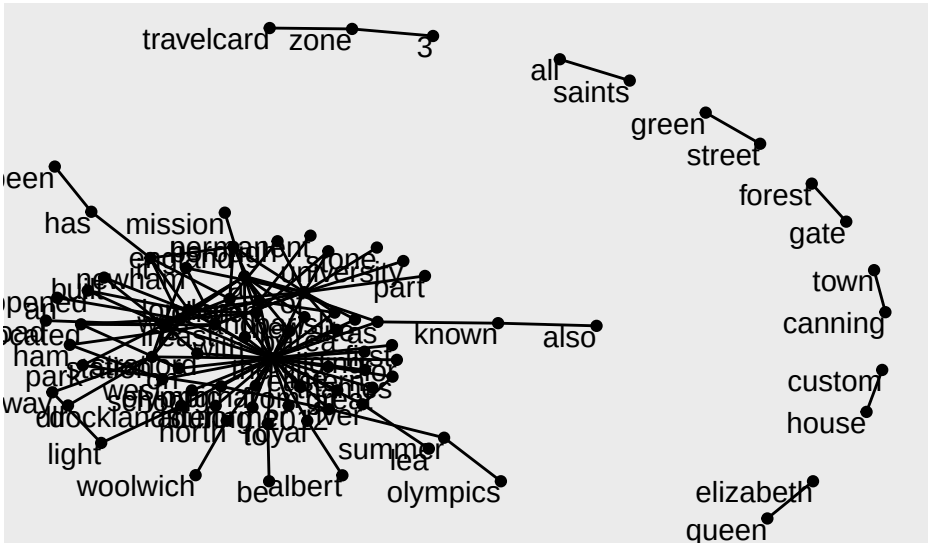
These are the bigram words are affilated to the bing lexicon values.

Let's plot the igraph and ggraph

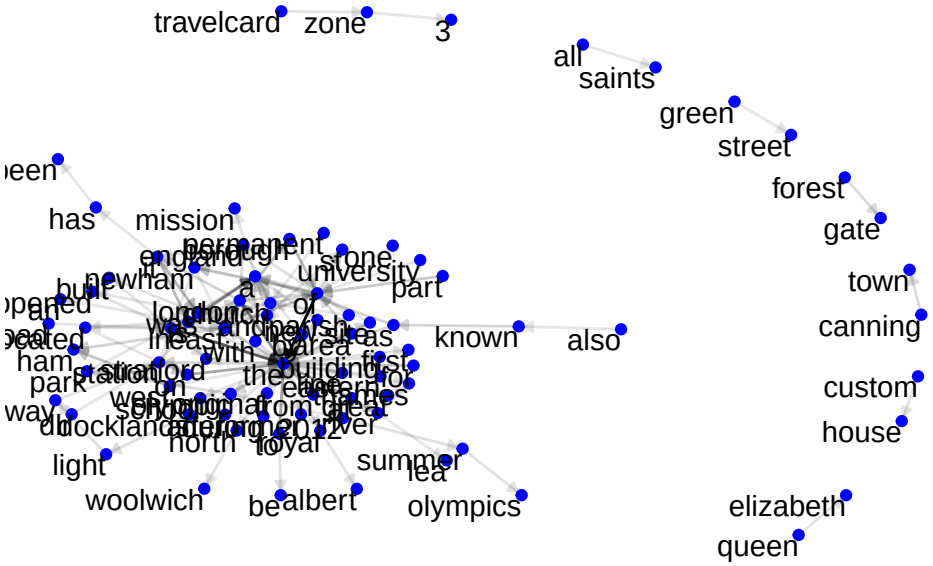
```
# convert the bigram to graph data frame
bigram_g <- bigram_sep %>%
  filter(n>20) %>%
  graph_from_data_frame()

bigram_g
```

```
## IGRAPH 0bbad68 DN-- 87 128 --
## + attr: name (v/c), n (e/n)
## + edges from 0bbad68 (vertex names):
##  [1] of     ->the     in     ->the     is     ->a       the    ->london
##  [5] to     ->the     east   ->london  on     ->the     it     ->was
##  [9] and    ->the     it     ->is      borough->of      london ->borough
## [13] of     ->newham  by     ->the     at     ->the     for    ->the
## [17] west   ->ham     part   ->of      in     ->east    the    ->river
## [21] as     ->a       from   ->the     the    ->station east   ->ham
## [25] london ->it      in     ->london  was    ->a       as     ->the
## [29] known  ->as      the    ->site    church ->of      of     ->st
## + ... omitted several edges
```

```
# Plot the ggraph
# To find the link between the words
set.seed(1234)

ggraph(bigram_g, layout = "kk") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```



```
# Plot the graph with node
set.seed(1234)

a <- grid::arrow(type = "closed", length = unit(.05, "inches"))

ggraph(bigram_g, layout = "kk") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.03, 'inches')) +
  geom_node_point(color = "blue", size = 1.5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```

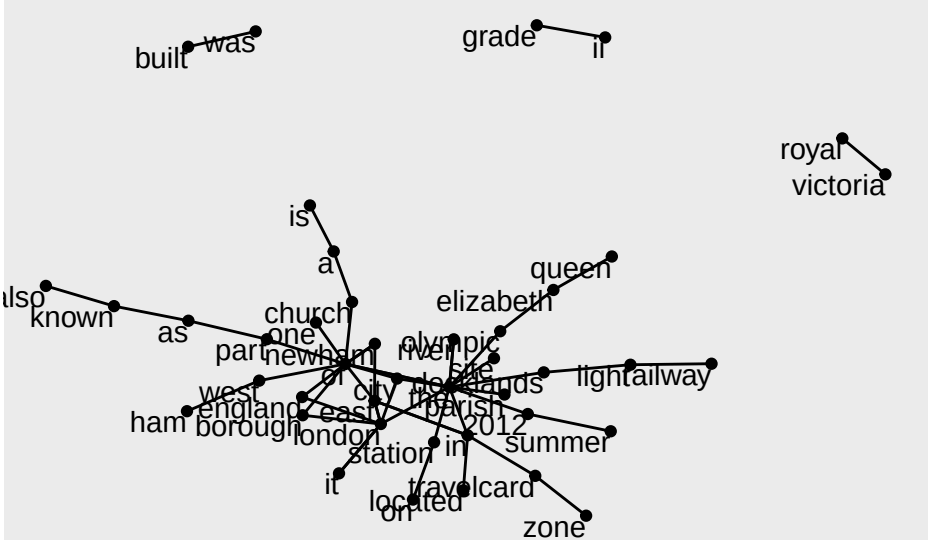


The most of the nodes are the most common words in the English. Hence, clearly not visible of the bigram word nodes

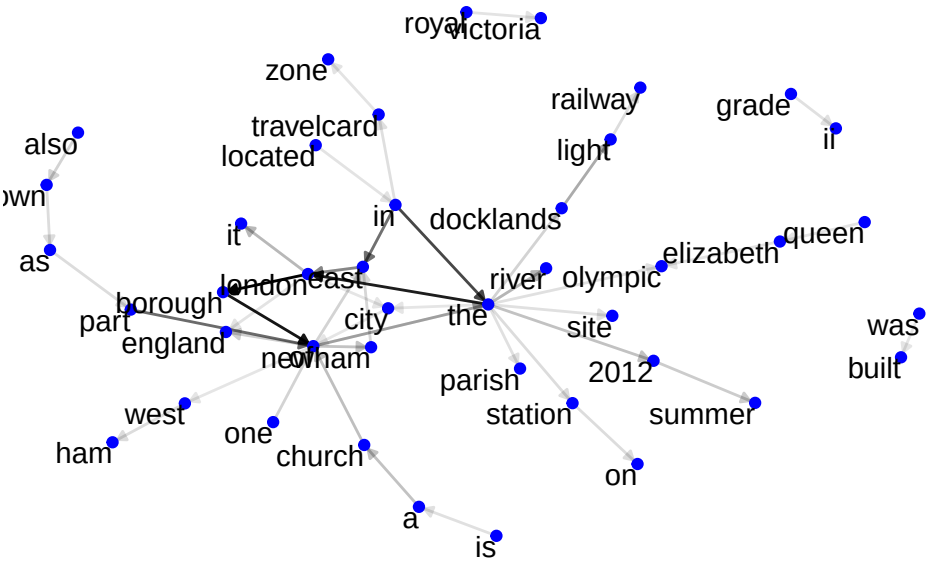
```
# Convert the trigram words to graph data frame
trigram_g <- trigram_sep %>%
  filter(n>15) %>%
  graph_from_data_frame()

trigram_g
```

```
## IGRAPH 18305c7 DN-- 42 51 --
## + attr: name (v/c), word3 (e/c), n (e/n)
## + edges from 18305c7 (vertex names):
##  [1] london   ->borough   borough  ->of        the      ->london
##  [4] in       ->the       part     ->of        in       ->east
##  [7] east     ->london    docklands->light     church   ->of
## [10] a        ->church    the      ->2012      of       ->the
## [13] 2012     ->summer    of       ->the       of       ->england
## [16] the      ->river     also     ->known     of       ->east
## [19] one      ->of        the      ->docklands london   ->it
## [22] newham   ->east      of       ->newham    as       ->part
## + ... omitted several edges
```

```
# Plot the graph
# To find the link between the words
set.seed(1234)

ggraph(trigram_g, layout = "kk") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```




```
# Plot the words with node points

set.seed(1234)

ggraph(trigram_g, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.03, 'inches')) +
  geom_node_point(color = "blue", size = 1.5) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
  theme_void()
```

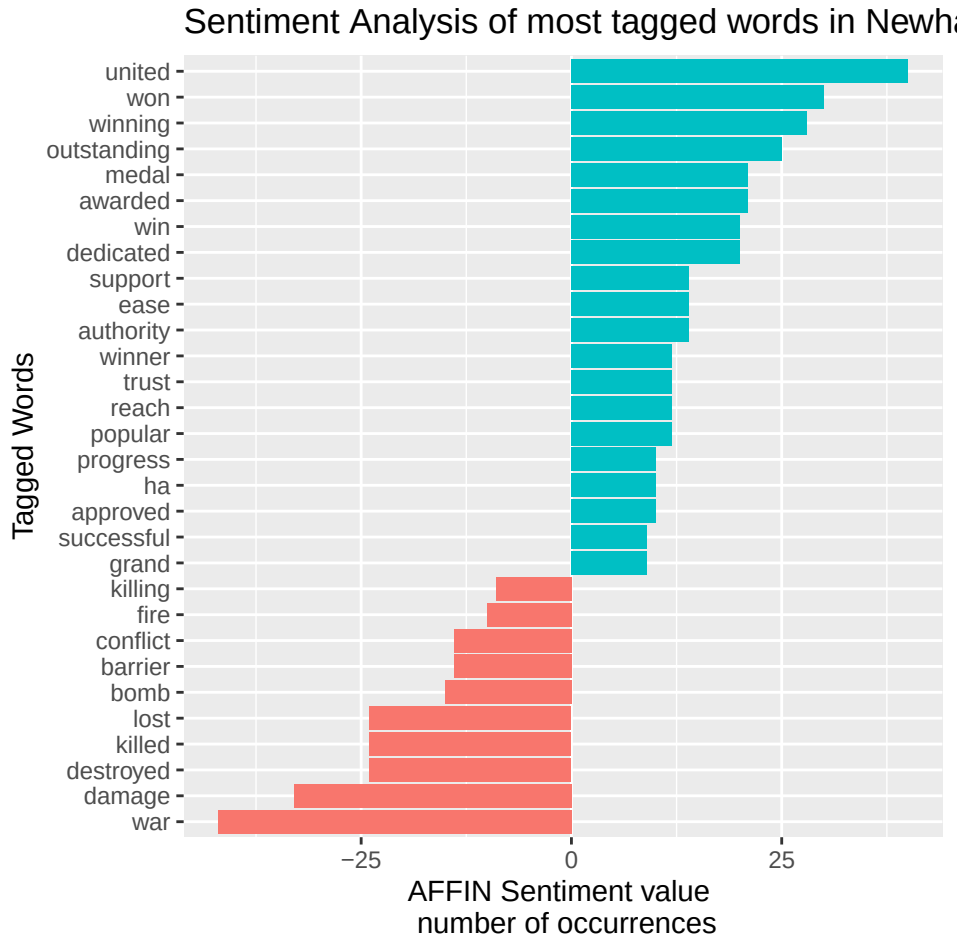


Newham, city, east are some of the centre node for the trigram words.

```
# Calculate the sentiment values for affin words
# Import the stem words
afinn <- stem_words %>%
  inner_join(AFINN, by = c(word = "word")) %>%
  count(word, value, sort = TRUE)

afinn
```

```
## # A tibble: 257 x 3
##    word      value     n
##    <chr>     <dbl> <int>
##  1 united        1    40
##  2 war          -2    21
##  3 bomb         -1    15
##  4 authority     1    14
##  5 reach         1    12
##  6 trust         1    12
##  7 damage       -3    11
##  8 dedicated     2    10
##  9 won           3    10
## 10 destroyed    -3     8
## # ... with 247 more rows
```

```
# Plot the sentiment values for the uni-gram words
afinn %>%
mutate(con = n * value) %>%
  arrange(desc(abs(con))) %>%
  head(30) %>%
  mutate(word = reorder(word, con)) %>%
  ggplot(aes(n * value, word, fill = n * value > 0)) +
  geom_col(show.legend = FALSE) +
  labs(x = "AFFIN Sentiment value \n number of occurrences",
       y = "Tagged Words") +
  ggtitle("Sentiment Analysis of most tagged words in Newham")
```



These are the words which are affilated to the AFINN values for the unigram words.

Spatial sentiment analysis usig bing Library

Let's join the bing libraries words with stemmed words.

```
# Import the stem words
bing_words <- stem_words %>%
  # Join the bing words
  inner_join(bing, by = c(word = "word")) %>%
  count(word, sentiment, sort = TRUE)

bing_words
```

```
## # A tibble: 300 x 3
##    word      sentiment     n
##    <chr>     <chr>     <int>
##  1 excel     positive     24
##  2 rail      negative     22
##  3 bomb      negative     15
##  4 damaged   negative     12
##  5 trust     positive     12
##  6 damage    negative     11
```

```
##  7 dedicated positive     10
##  8 won       positive     10
##  9 gold      positive      9
## 10 leading   positive      9
## # ... with 290 more rows
```

```
# Import the stem words with coordinates.
sentiment_words <- words_newham %>%
  # Join the bing sentiment values
  inner_join(bing, by = c(word = "word"))

sentiment_words
```

```
## # A tibble: 668 x 8
##    page_title             gt_type  gt_lon gt_lat  line word     stem   sentiment
##    <chr>                  <chr>     <dbl>  <dbl> <int> <chr>    <chr>  <chr>
##  1 Green_Street_House     NULL     0.0378   51.5     2 stately  state  positive
##  2 Green_Street_House     NULL     0.0378   51.5     2 modern   modern positive
##  3 Green_Street_House     NULL     0.0378   51.5     2 imposing impos  negative
##  4 Green_Street_House     NULL     0.0378   51.5     2 notably  notabl positive
##  5 Memorial_Community_Ch~ landmark 0.0239   51.5     3 byzanti~ byzan~ negative
##  6 St_Mary's_Church,_Pla~ NULL     0.0247   51.5     7 divine   divin  positive
##  7 St_Mary's_Church,_Pla~ NULL     0.0247   51.5     7 compass~ compa~ positive
##  8 St_Mary's_Church,_Pla~ NULL     0.0247   51.5     7 ease     eas    positive
##  9 St_Mary's_Church,_Pla~ NULL     0.0247   51.5     7 compreh~ compr~ positive
## 10 St_Philip_and_St_Jame~ NULL     0.0211   51.5     8 divine   divin  positive
## # ... with 658 more rows
```

Seggregate the postive words of bing values in separate variables.

```
# Seggregate the positive values
positive_words <- sentiment_words %>%
  filter(sentiment == "positive")

positive_words
```

```
## # A tibble: 378 x 8
##    page_title             gt_type gt_lon gt_lat  line word      stem   sentiment
##    <chr>                  <chr>    <dbl>  <dbl> <int> <chr>     <chr>  <chr>
##  1 Green_Street_House     NULL    0.0378   51.5     2 stately   state  positive
##  2 Green_Street_House     NULL    0.0378   51.5     2 modern    modern positive
##  3 Green_Street_House     NULL    0.0378   51.5     2 notably   notabl positive
##  4 St_Mary's_Church,_Pla~ NULL    0.0247   51.5     7 divine    divin  positive
##  5 St_Mary's_Church,_Pla~ NULL    0.0247   51.5     7 compassi~ compa~ positive
##  6 St_Mary's_Church,_Pla~ NULL    0.0247   51.5     7 ease      eas    positive
##  7 St_Mary's_Church,_Pla~ NULL    0.0247   51.5     7 comprehe~ compr~ positive
##  8 St_Philip_and_St_Jame~ NULL    0.0211   51.5     8 divine    divin  positive
##  9 St_Philip_and_St_Jame~ NULL    0.0211   51.5     8 compassi~ compa~ positive
## 10 Holy_Trinity_Church,_~ NULL    0.0135   51.5     9 holy      holi   positive
## # ... with 368 more rows
```

Seggregate the negative words of bing values in separate variables.

```
# Seggregate the negative values
negative_words <- sentiment_words %>%
  filter(sentiment == "negative")
```

```
negative_words
```

```
## # A tibble: 290 x 8
##    page_title               gt_type  gt_lon gt_lat  line word   stem   sentiment
##    <chr>                    <chr>     <dbl>  <dbl> <int> <chr>  <chr>  <chr>
##  1 Green_Street_House       NULL     0.0378   51.5     2 impos~ impos  negative
##  2 Memorial_Community_Chur~ landmark 0.0239   51.5     3 byzan~ byzan~ negative
##  3 Holy_Trinity_Church,_Ca~ NULL     0.0135   51.5     9 badly  badli  negative
##  4 Holy_Trinity_Church,_Ca~ NULL     0.0135   51.5     9 damag~ damag  negative
##  5 Holy_Trinity_Church,_Ca~ NULL     0.0135   51.5     9 damage damag  negative
##  6 St_Paul's_Church,_East_~ NULL     0.0630   51.5    11 died   di     negative
##  7 St_Michael_and_All_Ange~ NULL     0.0606   51.5    12 burni~ burn   negative
##  8 St_Alban's_Church,_Upto~ NULL     0.0446   51.5    16 damag~ damag  negative
##  9 St_Michael's_Church,_Ru~ NULL     0.0397   51.5    17 damag~ damag  negative
## 10 Church_of_the_Ascension~ NULL     0.0373   51.5    18 split  split  negative
## # ... with 280 more rows
```

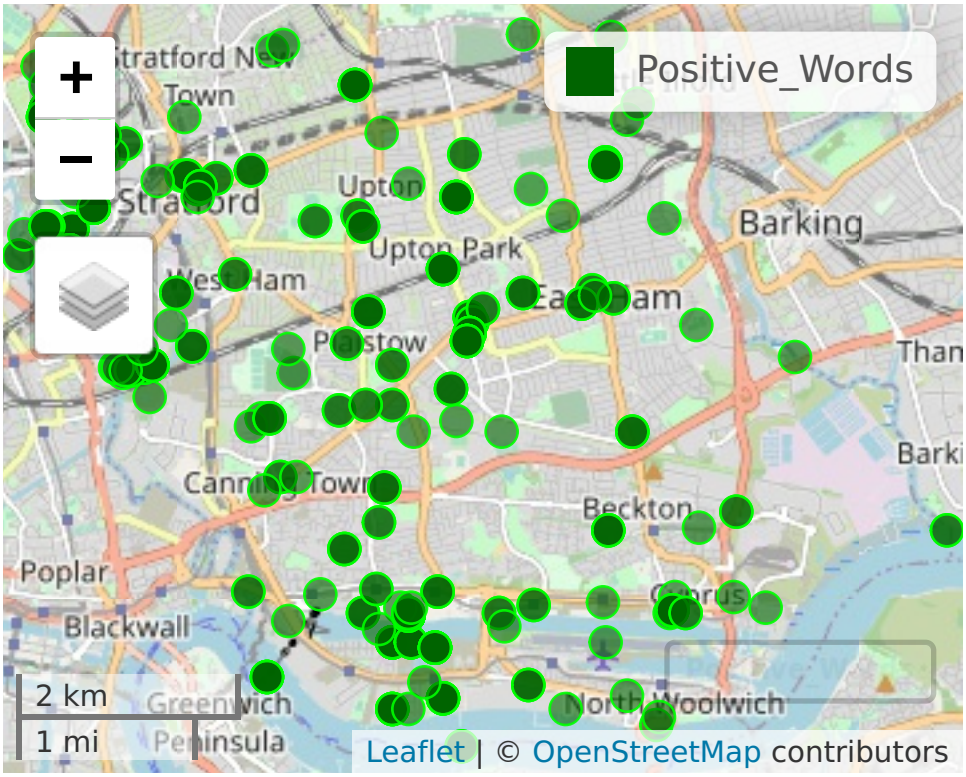
Plot the positive and negative words spatially

Plot the positive words

```
# Convert the data into spatial points
Positive_Words <- st_as_sf(positive_words, coords = c("gt_lon","gt_lat")) %>%
  # Set CRS as EPSG:4326
  st_set_crs("EPSG:4326") %>%
  # Cast as point
  st_cast("POINT")
```

Plot the positive words in the mapview

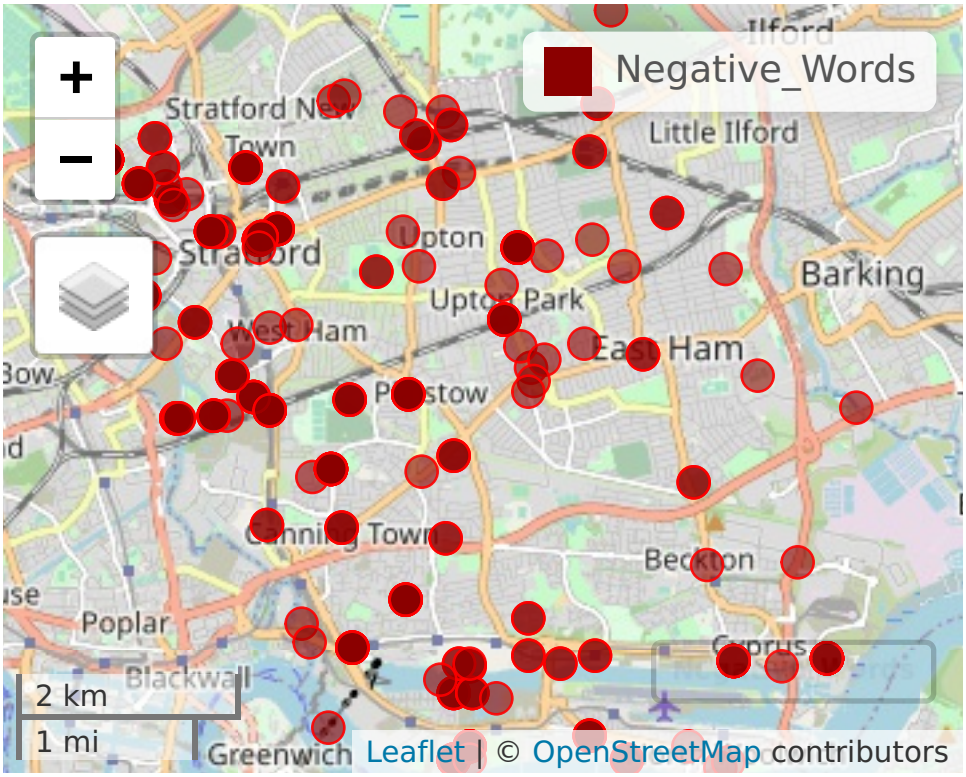
```
# Plot the map
mapview(Positive_Words, map.type="OpenStreetMap", col.regions="darkgreen", color ="green")
```



The postive words which are tagged throughout the Newham regions.

```
# Convert the data into spatial points
Negative_Words <- st_as_sf(negative_words, coords = c("gt_lon","gt_lat")) %>%
  # Set CRS as EPSG:4326
  st_set_crs("EPSG:4326") %>%
  # Cast as point
  st_cast("POINT")
```

```
# Plot the map
mapview(Negative_Words, map.type="OpenStreetMap", col.regions="darkred", color ="red")
```



The negative words which are tagged scattered some areas in the Newham regions.

Discussion and Conclusion:

We plotted the most common words that tagged bi-words and tri-words by analysing bigram and trigram words. East London, London borough, east ham, west ham, and so on are the most frequently bigram tagged words, while docklands light railway, 2012 summer Olympics, newham east London, west ham dlr, and so on are the most frequently tri-gram tagged words. The representation of text data in visual format has been plotted as a wordcloud for both bigram and trigram that are tagged in Newham borough. We visualised the nodes of the text structure using ggraph; in bigram, we can see nods such as was, of, the, and it, which are common words in English but are not clearly visualised. The nods newham, east, and city are common node centres in trigram. The sentiment values were measured using negated words such as not, no, no one, and so on for the bigram. Winning, satisfied and charges are the words associated with the negated words for AFINN values and for the bing lexicon explosive, objection and overlook are associated with negative values and ideal, satified and winning are the positive words which are plotted. The AFINN and bing values for the unigram stem terms were then determined, and the top 30 words with a mix of positive and negative values were plotted which has total 257 words. The bing value for the unigram stem words were calculated with the number of occurances. The 668 stem words with coordinates are combined with the measured bing values, and the 378 positive words and 290 negative words are separated into different variables. Both variables are translated to two separate spatial points, allowing the words to be plotted in the mapview.