Univerità degli studi di Padova

THE SIMPSONS

Giovanni Corradini, Pietro De Vecchi, Silvia Moro, Chiara Micheletti

29 giugno 2019

Indice

Introduzione

In questo eleborato sono stati analizzati i dialoghi, in lingua inglese, di circa 600 episodi della celebre serie animata "The Simpsons", creata da Matt Groening nel 1987.

I principali obiettivi della ricerca sono:

- evidenziare le parole più frequenti dei personaggi principali e capire eventuali termini caraterizzanti;
- analizzare il sentiment di alcuni personaggi caratteristici;
- individuare degli eventuali topic specifici per determinati luoghi;
- proporre due modelli alternativi in grado di predire, con un certo grado di accuratezza, quale sia il personaggio che recita una certa battuta, in base alle parole che vengono utilizzate nella battuta stessa.

Descrizione dei dati

I dati sono stati reperiti su Kaggle. Il dataset, "The Simpsons by the Data", caricato da William Cukierski, si compone di diversi file. Per gli obiettivi della ricerca si è concentrata l'attenzione su *simpsons_script_lines.csv*.

Il dataset è composto da 157462 righe (una per ciascuna battuta) e 13 colonne, rappresentanti delle variabili di interesse, che riguardano:

- **id**: numero identificativo della battuta;
- **episode_id**: numero identificativo dell'episodio;
- **number**: numero della battuta in ciascun episodio (es. number=0 –> prima battuta dell'episodio);
- **raw_text**: testo della battuta non normalizzato e con all'interno precisato chi parla (es. *Homer Simpson: I want a beer.*);
- **timestamp_in_ms**: marca temporale di quando viene pronunciata la battuta (espressa in millisecondi);
- **speaking_line**: 'TRUE' se vengono pronunciate parole, 'FALSE' altri-

menti (versi, rumori…);

- **character_id**: numero identificativo del personaggio parlante;
- **location_id**: numero identificativo della location dove avviene il dialogo;
- **raw_character_text**: nome del personaggio parlante;
- **raw_location_text**: nome della location dove avviene il dialogo;
- **spoken_words**: testo della battuta non normalizzato;
- **normalized_text**: testo della battuta normalizzato;
- **word_count**: conteggio delle parole pronunciate nella battuta.

Per evitare problematiche durante le varie analisi è stato deciso di rimuovere preventivamente le battute prive di vocaboli sfruttando la variabile *speaking_line* == 'TRUE'.

Pulizia del dataset

Nell'iniziale fase di pulizia dei dati sono state eseguite alcune operazioni: normalizzazione dei testi, stemming, gestione dei dati mancanti e creazione e aggiunta di due variabili che permettono di utilizzare istantaneamente la frequenza delle parole:

-**word_count** : conta il numero delle parole nella battuta normalizzata;
-**tot_words** : conta il numero totali delle parole pronunciate da un personaggio in tutti gli episodi in cui compare.

```
data("stop_words")
# Normalizzazione dataset:
df <- db %>%
  select(id, spoken_words) %>%
  unnest_tokens(output = word, input = spoken_words) %>%
  filter(!str_detect(word, "^[0-9]*$")) %>% # rimozione numeri
  anti_join(get_stopwords()) %>%    # rimuove stop words di snowball
  anti_join(stop_words) %>%    # rimuove altre stop words
  mutate(word = SnowballC::wordStem(word)) %>%   # stemming
  group_by(id) %>%
  summarise(testo = paste(word, collapse = ' ')) %>%
  left_join(db, by = 'id') %>%
  filter(!is.na(character_id)) %>%    # 1 obs
  filter(!is.na(location_id)) %>%     # 359 obs, per lo più in TV
  group_by(id) %>%
  summarise(testo, episode_id, character_id, raw_character_text,
            word_count = wordcount(testo))
```

Per quanto riguarda la normalizzazione, lavorando sulla variabile *spoken_words*, sono state utilizzate funzioni contenute nelle librerie `ngram` e `tidytext` per rimuovere numeri, caratteri speciali e stopwords. Lo stemming invece è basato sul pacchetto `Snowball` che permette il collasso delle parole in un'unica radice comune per facilitare il confronto di vocabolario. Infine, i dati mancanti sono stati filtrati grazie a funzioni di base, andando ad eliminare dal dataset una osservazione che presentava la variabile **character_id** senza attributo e 359 battute che sono state pronunciate in televisione che non presentano alcun valore in **location_id** .

Si è infine calcolato il rapporto tra l'ampiezza del vocabolario e la dimensione del corpus, per verificare che venissero rispettati i requisiti imposti in letteratura per poter effettuare analisi statistiche.
Tale rapporto è risultato idoneo, in quanto pari a 6.2% e quindi inferiore al valore soglia del 20% .
Lo stesso vale per la percentuale di hapax (43%), inferiore al valore soglia (50%), indicante un linguaggio trattabile statisticamente perché non troppo originale.

```
corpus<-Corpus(VectorSource(df$testo))
dtm<-DocumentTermMatrix(corpus)
tdm<-TermDocumentMatrix(corpus)

n<-colSums(df[,6])
voc <- table( unlist(strsplit(df$testo," ")) )
v_n<-dim(voc)
v_n/n
```

```
## word_count
## 0.06222026
```

```
hapax<- findFreqTerms(tdm, lowfreq = 0, highfreq = 1)
num_hap<-length(hapax)/v_n
num_hap
```

```
## [1] 0.4314838
```

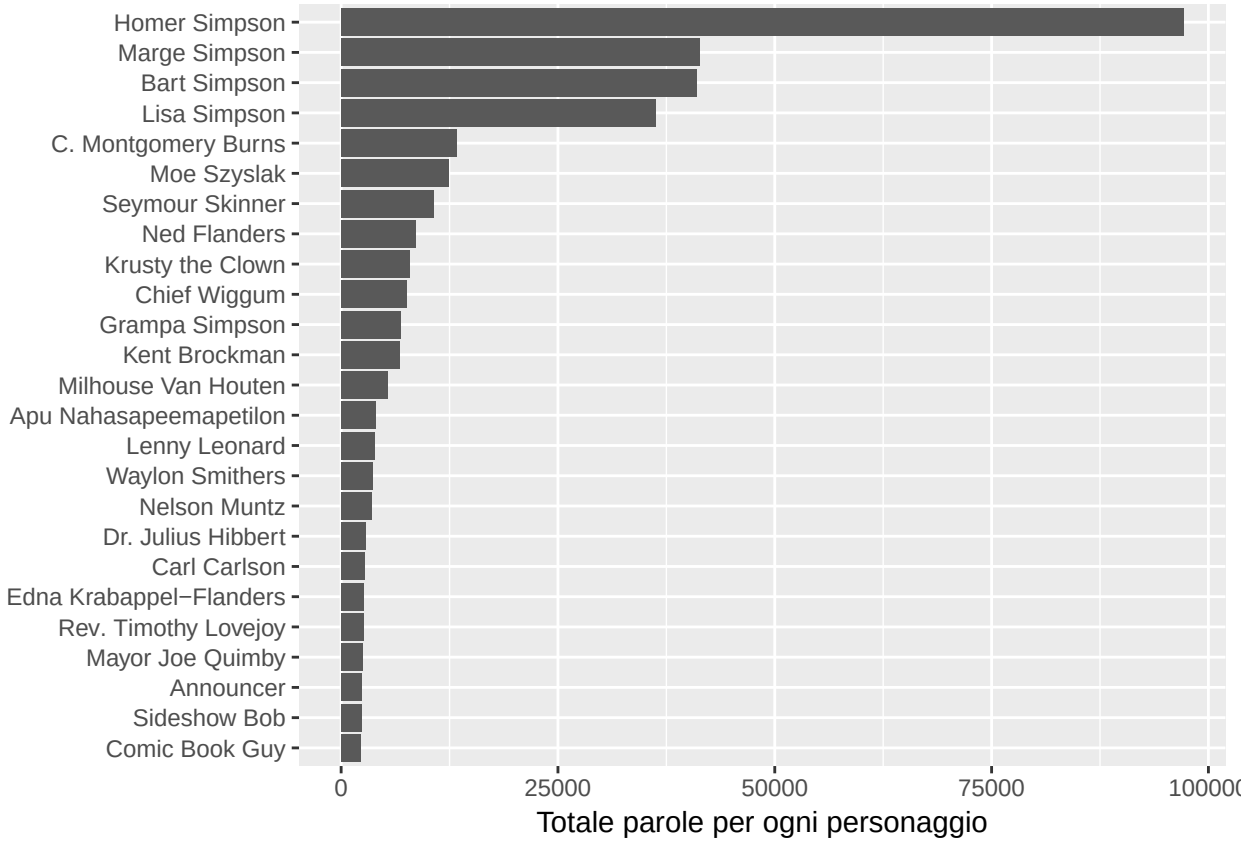
Frequenza parole

Con i dati puliti, sono stati dapprima individuati i personaggi della serie che parlano di più e poi le parole in generale più pronuciate in tutti i dialoghi. Per "parole", dal momento che dai testi sono state rimosse le stopwords, si intenderà d'ora in avanti parole **piene**. Per questa parte del lavoro sono state utilizzate solo le variabili *character_id*, *raw_character_text* e *tot_words*.

```
# spring_vip contiene testi normalizzati e senza stop_words
spring_vip <- df %>%
  group_by(character_id,raw_character_text) %>%
  summarise(tot_words = sum(word_count)) %>%
  arrange(desc(tot_words))
```

Grafico 1: Personaggi che pronunciano più parole

```
spring_vip %>% ungroup() %>% top_n(25) %>%
  mutate(raw_character_text = reorder(raw_character_text, tot_words)) %>%
  ggplot(aes(raw_character_text, tot_words)) +
  geom_col() +
  ylab('Totale parole per ogni personaggio') +
  xlab(NULL) +
  coord_flip()
```



Com'era prevedibile, i componenti della famiglia (esclusa Maggie) sono i personaggi che pronunciano più parole. Essi sono seguiti da: Montgomery Burns, Moe Szyslak, Ned Flanders, Waylon Smithers, Seymour Skinner e Krusty il Clown. Si è pertanto deciso di condurre la quasi totalità delle analisi su questi.

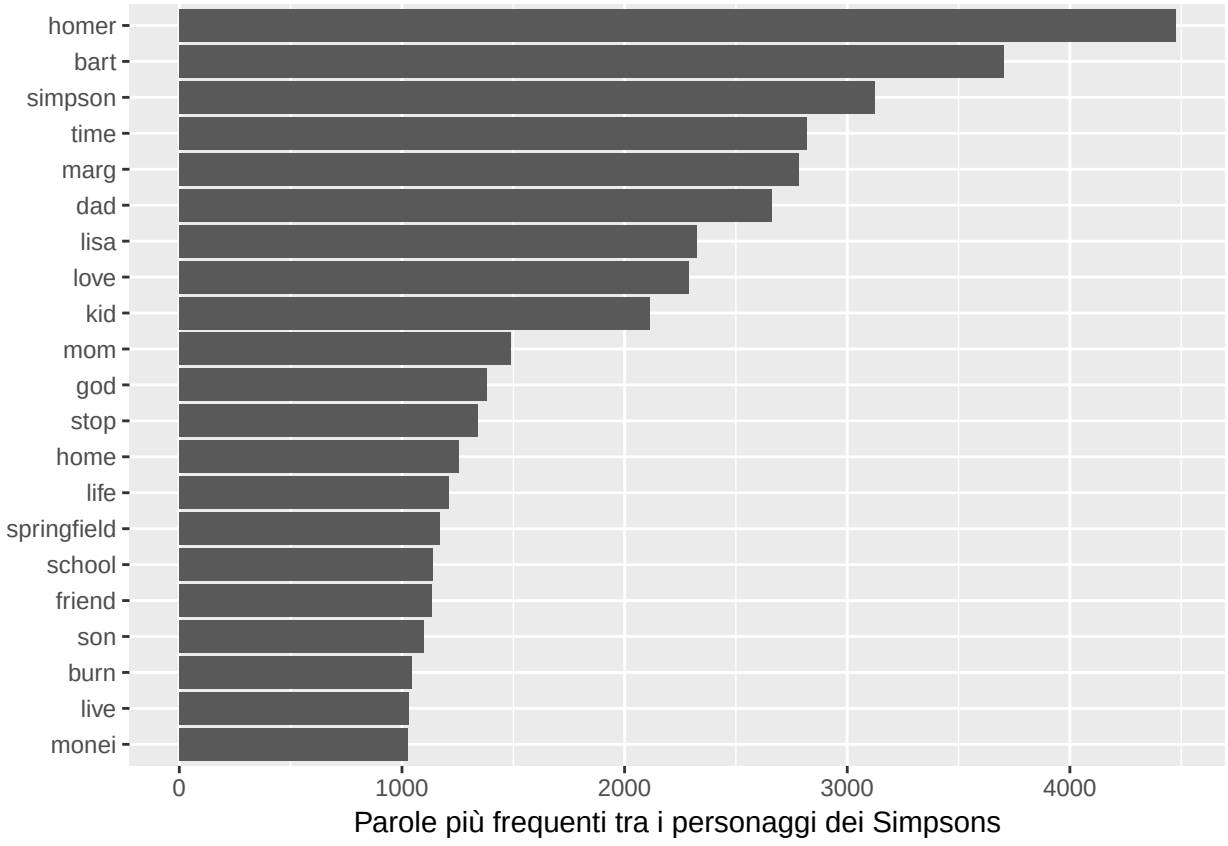
Inoltre, da un'analisi preliminare sono emersi numerosi termini (intercalari, espressioni tipiche, slang) che, anche se potrebbero essere in qualche modo indicativi del personaggio che le pronuncia, in alcuni casi sono risultati fuorvianti per le specifiche procedure utilizzate. Per questo motivo è stato scelto di considerare alcune espressioni come stopwords, eliminandole dal dataset.

```
s_w <- bind_rows(tibble(word = c('hei','uh','gonna','boi','dai',
                                 'gui','call','peopl','wait',
                                 'feel','sir','talk','huh',
                                 'yeah','ya','ah'),
                        lexicon = rep('simpsons',16)),
                 stop_words)
```

Prima di concentrare l'attenzione sui personaggi sopra citati, per avere un'idea del lessico utilizzato, sono state individuate le parole più pronunciate in generale all'interno della serie.

Grafico 2: Parole più frequenti tra tutti i personaggi dei Simpsons

```
select(df, testo) %>%
  unnest_tokens(word, testo) %>%
  anti_join(s_w) %>%
  count(word, sort = TRUE) %>%
  filter(n > 1000) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word,n)) +
  geom_col() +
  labs(y = "Parole più frequenti tra i personaggi dei Simpsons",
       x = NULL) +
  coord_flip()
```



Come preannunciato, d'ora in poi verranno presi in considerazione solo Homer, Marge, Bart, Lisa, Burns, Moe, Ned e Krusty. (Skinner pronuncia più parole di Krusty, ma si è preferito quest'ultimo per la particolarità del suo linguaggio.)

```
dati_spring <- select(df, c(testo, character_id)) %>%
# 'testo' è il testo normalizzato
  filter(character_id %in% c(1,2,8,9,15,17,11,139)) %>%
  mutate(character_id = as.factor(character_id))

# dati_spring è il dataset con i testi normalizzati e il 'character_id'

# Gestione livelli fattore

levels(dati_spring$character_id)[levels(dati_spring$character_id)=="1"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="2"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="8"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="9"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="15"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="17"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="11"]
levels(dati_spring$character_id)[levels(dati_spring$character_id)=="139"]

spring_words <- dati_spring %>%
  unnest_tokens(word, testo) %>%
  anti_join(s_w) %>% # stopwords aggiunte manualmente
  count(character_id, word, sort = TRUE)

tot_parole <- spring_words %>%
  group_by(character_id) %>%
  summarize(total = sum(n))

spring_words <- left_join(spring_words, tot_parole)

mod_stargazer(spring_words[15,], summary=FALSE)
```

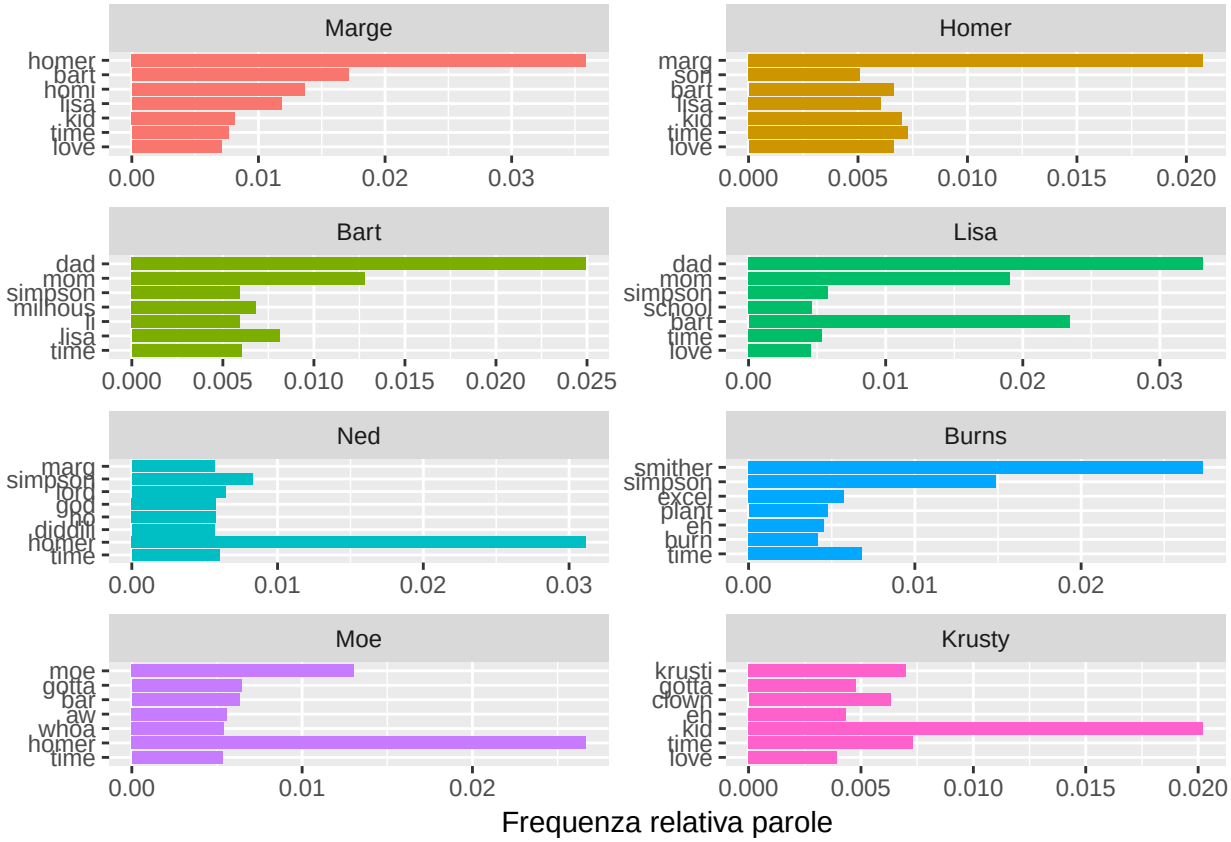
Table 1:

	character_id	word	n	total
1	1	lisa	462	39082

Per ciascuno degli otto personaggi considerati è stata analizzata la frequenza relativa delle parole pronunciate, considerando testi normalizzati e senza stop-words, al fine di evidenziare alcune peculiarità.

Grafico 3: Frequenze relative parole pronunciate dagli otto personaggi selezionati

```
spring_words %>%
  mutate(score = n/total) %>%
  arrange(desc(score)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  select(-total) %>%
  group_by(character_id) %>%
  top_n(7) %>%
  mutate(word = reorder(word, score)) %>%
  ungroup() %>%
  ggplot(aes(word, score, fill = character_id)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "Frequenza relativa parole") +
  facet_wrap(~character_id, ncol = 2, scales = "free") +
  coord_flip()
```



I grafici rispecchiano le relazioni che legano i vari personaggi:

- la parola più usata da Homer è “Marge” (e viceversa);
- la parola più usata da Burns è “Smithers”.
- la parola più usata sia da Lisa che da Bart è “dad”;

Come nel **Grafico 2**, si nota che “Homer” è tra le parole più usate ed è proprio il protagonista della serie. Infine, può risultare sorprendente che tra le parole più frequenti di Homer non compaia la sua tipica esclamazione “D’oh”: questo può essere dovuto alle operazioni preliminari di normalizzazione del testo.
Si troverà poi riscontro di questi risultati nella prossima sezione, dedicata all’indice tf-idf.

Indice Tf-idf

E'stato dunque calcolato e rappresentato graficamente l'indice *tf-idf*, prima delle parole dei componenti della famiglia e poi dei restanti quattro personaggi presi in considerazione, per individuare le parole più caratterizzanti di ciascuno. Il primo calcolo dell'indice ha però evidenziato espressioni e modi di dire non significativi, perché troppo generici. Per ottenere risultati migliori si è deciso di aggiornare il precedente elenco di stopwords.

```
s_w_tf <- bind_rows(tibble(word = c('hei','uh','gonna','boi','dai',
                                    'gui','call','peopl','wait',
                                    'feel','sir','talk','huh',
                                    'yeah','ya','ah'),
                           lexicon = rep('simpsons',16)),
                    stop_words)

spring_words_tf <- dati_spring %>%
  unnest_tokens(word, testo) %>%
  anti_join(s_w_tf) %>% # stopwords aggiunte manualmente
  count(character_id, word, sort = TRUE)

tot_parole <- spring_words_tf %>%
  group_by(character_id) %>%
  summarize(total = sum(n))

spring_words_tf <- left_join(spring_words_tf, tot_parole)
```

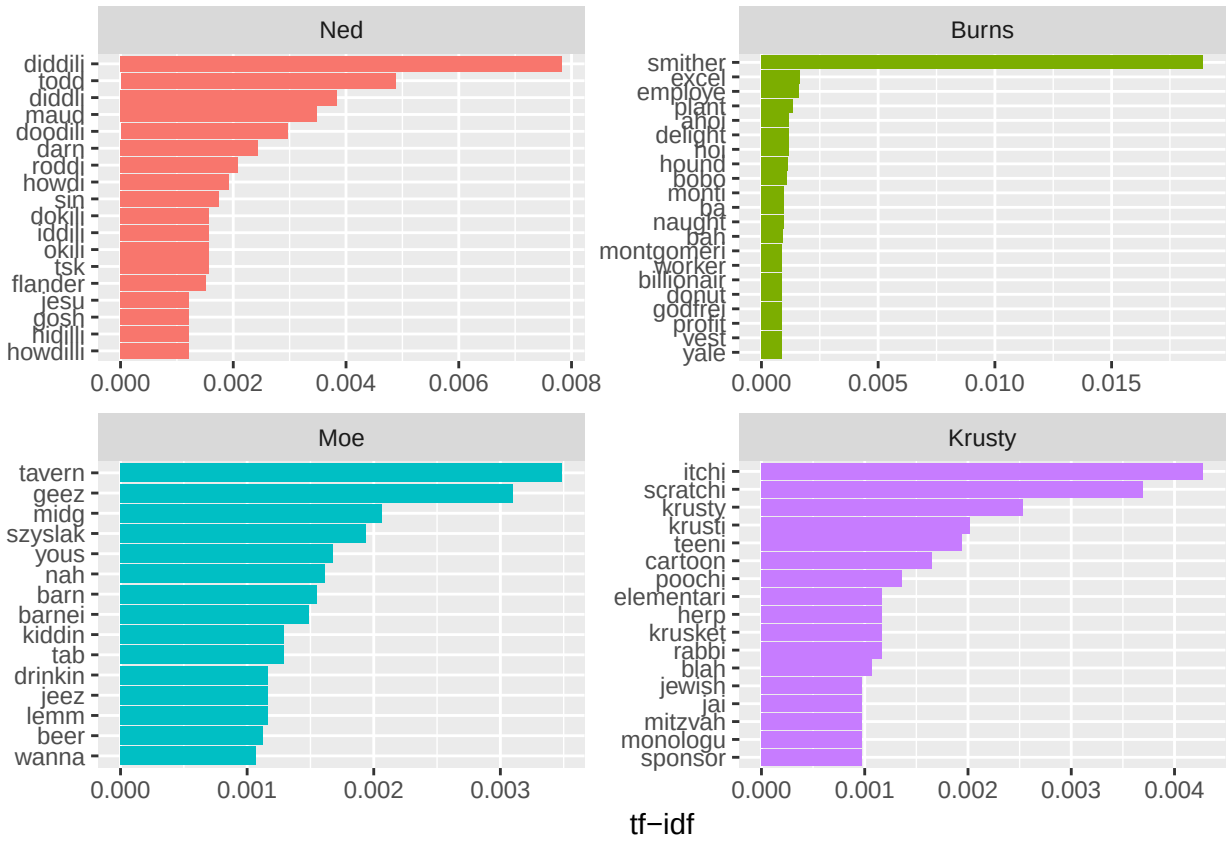
Grafico 4: Valori tf-idf per i quattro componenti della famiglia Simpson

```
dati_fam_tfidf <- spring_words_tf %>%
  filter(character_id %in% c('Homer','Marge','Lisa','Bart')) %>%
  bind_tf_idf(word, character_id, n)

dati_fam_tfidf %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(character_id) %>%
  top_n(13) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = character_id)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~character_id, ncol = 2, scales = "free") +
  coord_flip()
```

La parola in assoluto più significativa per Marge è "homie", nomignolo da lei affibiato al marito Homer. Allo stesso tempo una delle parole più caratterizzanti per quest'ultimo è "sweetie", ovvero "dolcezza", con cui è solito riferirsi a Marge. E' interessante notare come il termine con valore dell'indice più alto per Homer sia "hee", la sua tipica risata. Il corrispondente italiano del termine "Carumba", proprio di Bart, è "Oh cacchio", prima parola che impara da bambino e che poi continua ad usare anche quando cresce. Infine Lisa stupisce con il suo linguaggio forbito con vocaboli come "anagram", "feminist" e "buddhist", anomali per una bambina di 8 anni.

Grafico 5: Valori tf-idf per Ned, Burns, Moe, Krusty

```
dati_spring_tfidf <- spring_words_tf %>%
  filter(character_id  %in% c('Burns','Moe','Krusty','Ned')) %>%
  bind_tf_idf(word, character_id, n)

dati_spring_tfidf %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(character_id) %>%
  top_n(15) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = character_id)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~character_id, ncol = 2, scales = "free") +
  coord_flip()
```



Nel barplot di Ned si ritrova il suo linguaggio ricco di diminutivi e vezzeggiativi, come “diddily” e “doodily”, e, ancora una volta, termini religiosi quali “jesus”, “sin” e “gosh”. In Burns tornano, come già visto nell’analisi delle frequenze, “excellent” e “Smithers”, insieme a vocaboli caratteristici di un businessman come “profit”, “billionaire” e “employee”. “Tavern” e “beer” sono forse risultati prevedibili per Moe, essendo il proprietario di un pub, mentre sorprende l’espressione “Midge”, soprannome che riserva a Marge, per cui ha avuto per anni un’infatuazione. A caratterizzare Krusty sono in particolare le parole “Itchy” e “Scratchy”, Grattachecca e Fichetto nella versione italiana, protagonisti del cartone animato più amato dai bambini di Springfield e che viene trasmesso durante gli spettacoli del clown. Si notano anche termini legati alla tradizione ebraica come “mitzvah” e “jewish”, dovuti alle sue origini.

In conclusione, l’indice riesce a ritrarre in maniera veritiera i personaggi, individuando bene le parole più caratteristiche di ciascuno. Anche per questo motivo si è deciso di inserirlo come variabile esplicativa nei modelli finali.

Sentiment Analysis

Per la Sentiment Analysis sono stati riportati i risultati solo di Homer, Burns e Lisa perché più sensati e significativi.
Per questa procedura sono stati utilizzati i dizionari *bing* e *nrc*, evidenziando con il primo solo le parole positive e negative prounciate da ciascuno, con il secondo dieci sentimenti prestabiliti dal dizionario (positivo, negativo, rabbia, aspettativa, disgusto, paura, gioia, tristezza, sorpresa e fiducia). La scelta di non concentrare l'attenzione su un unico dizionario è volta a verificare la correttezza e la coerenza di classificazione di alcuni termini.

Grafico 6: I sentimenti di Homer (bing)

```
Sentimenti_di_Homer <- dati_spring %>%
  unnest_tokens(word, testo) %>%
  filter(character_id == 'Homer') %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
Sentimenti_di_Homer %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Sentimenti di Homer",
       x = NULL) +
  coord_flip()
```

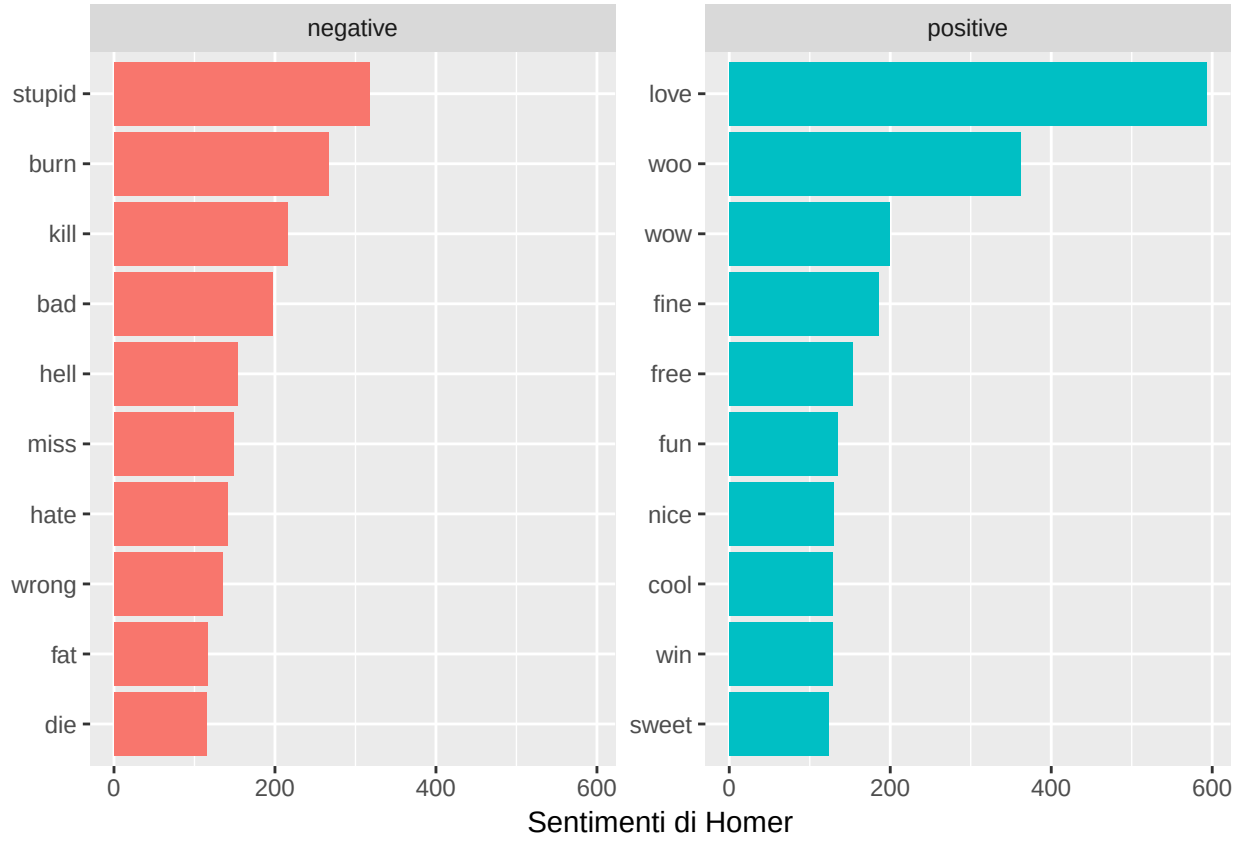


Grafico 7: I sentimenti di Homer (nrc)

```
Sentimenti_di_Homer2 <- dati_spring %>%
  unnest_tokens(word, testo) %>%
  filter(character_id == 'Homer') %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
Sentimenti_di_Homer2 %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
```

```
  labs(y = "Sentimenti di Homer con nrc",
       x = NULL) +
  coord_flip()
```

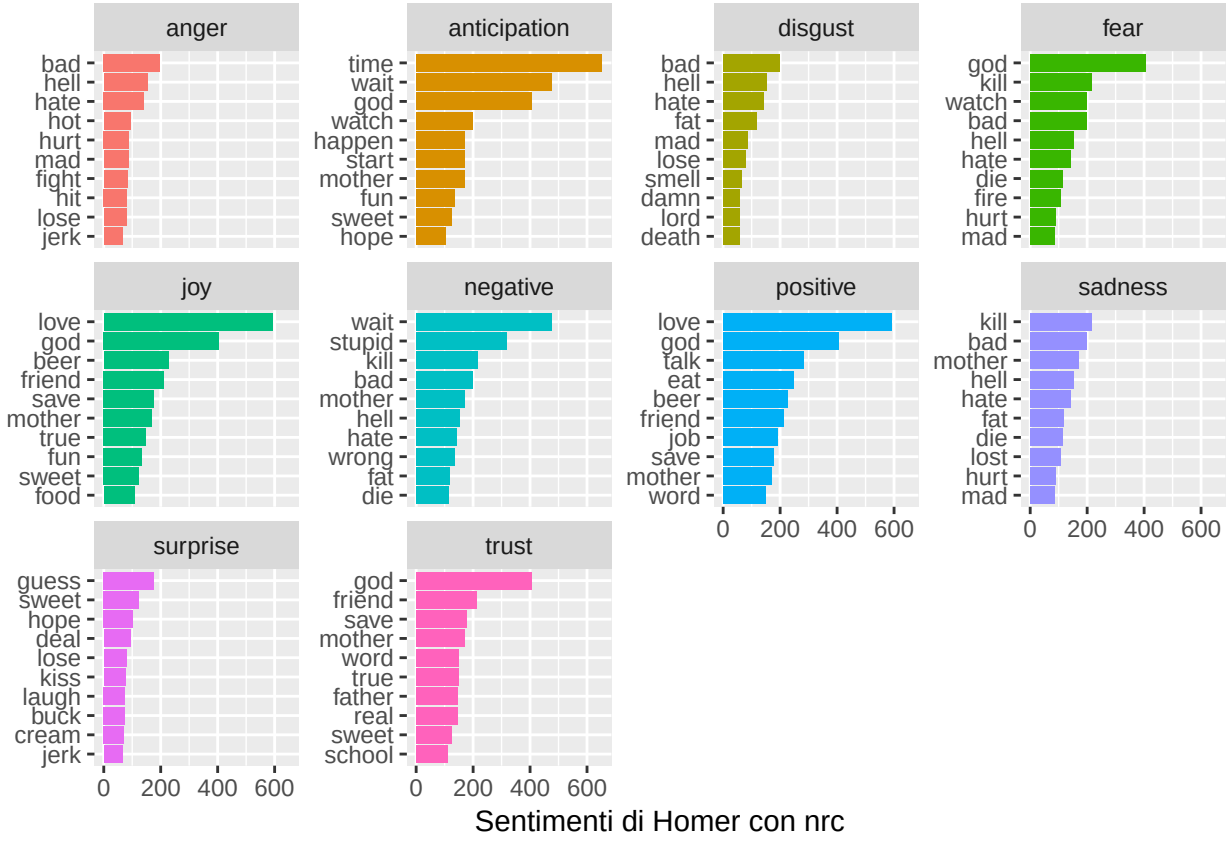


Grafico 8: I sentimenti di Burns (bing)

```
Sentimenti_di_Burns <- dati_spring %>%
  unnest_tokens(word, testo) %>%
  filter(character_id == 'Burns') %>%
  filter(word != 'burn') %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
Sentimenti_di_Burns %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
```

```
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Sentimenti di Burns",
       x = NULL) +
  coord_flip()
```

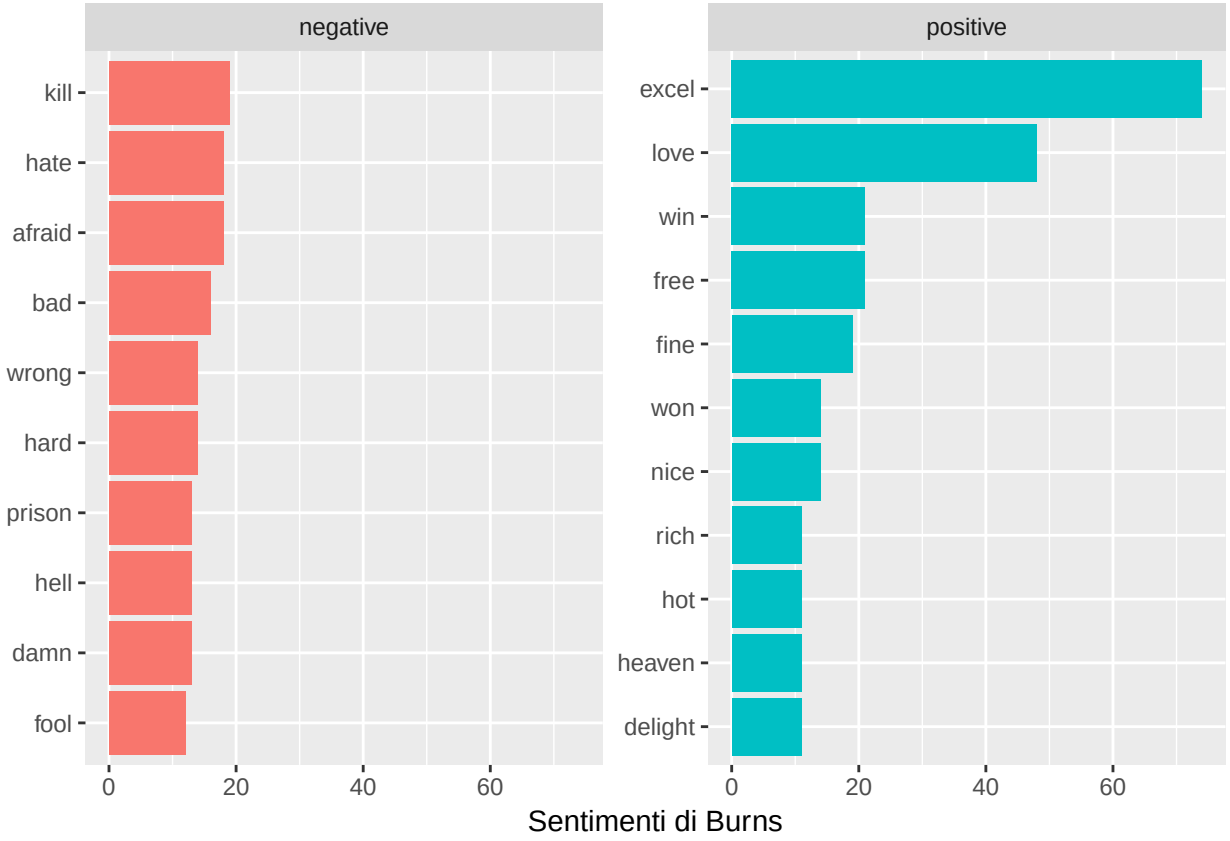
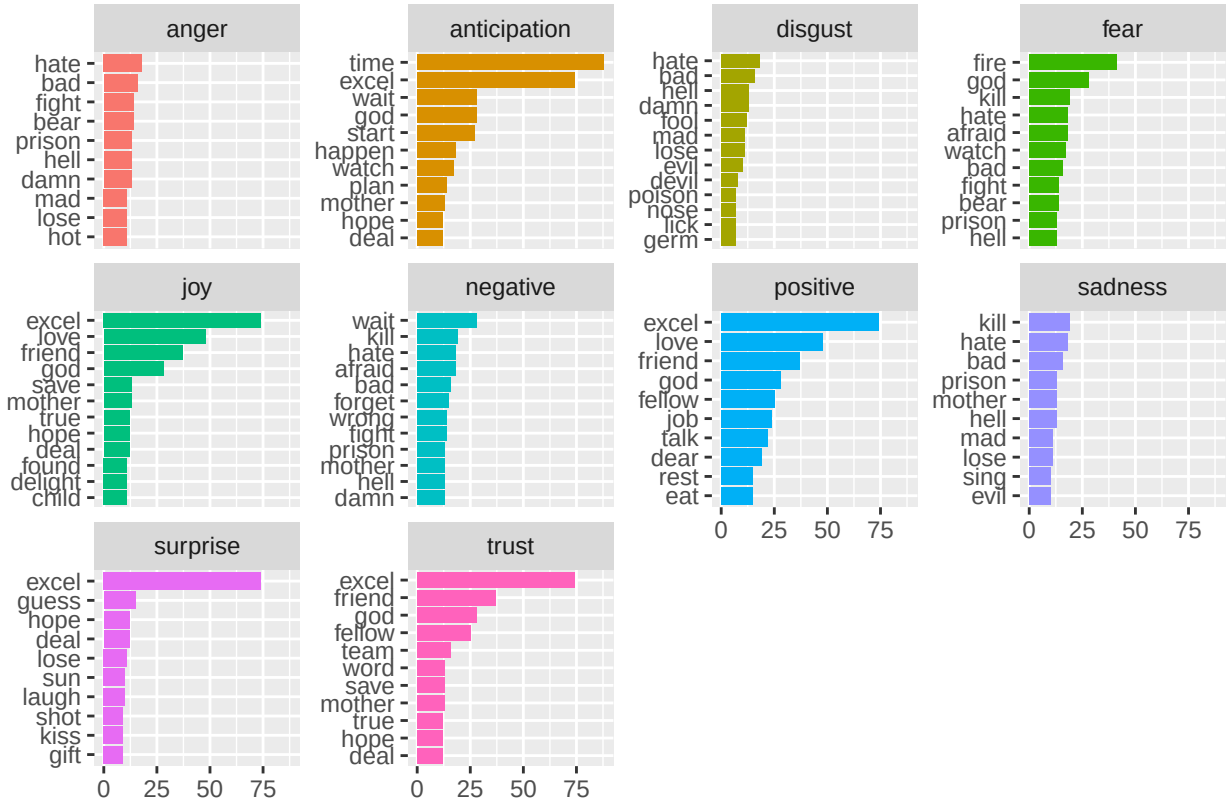


Grafico 9: I sentimenti di Burns (nrc)

```
Sentimenti_di_Burns2 <- dati_spring %>%
  unnest_tokens(word, testo) %>%
  filter(character_id == 'Burns') %>%
  filter(word != 'burn') %>%
  inner_join(get_sentiments("nrc")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
Sentimenti_di_Burns2 %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Sentimenti di Burns con nrc",
       x = NULL) +
  coord_flip()
```

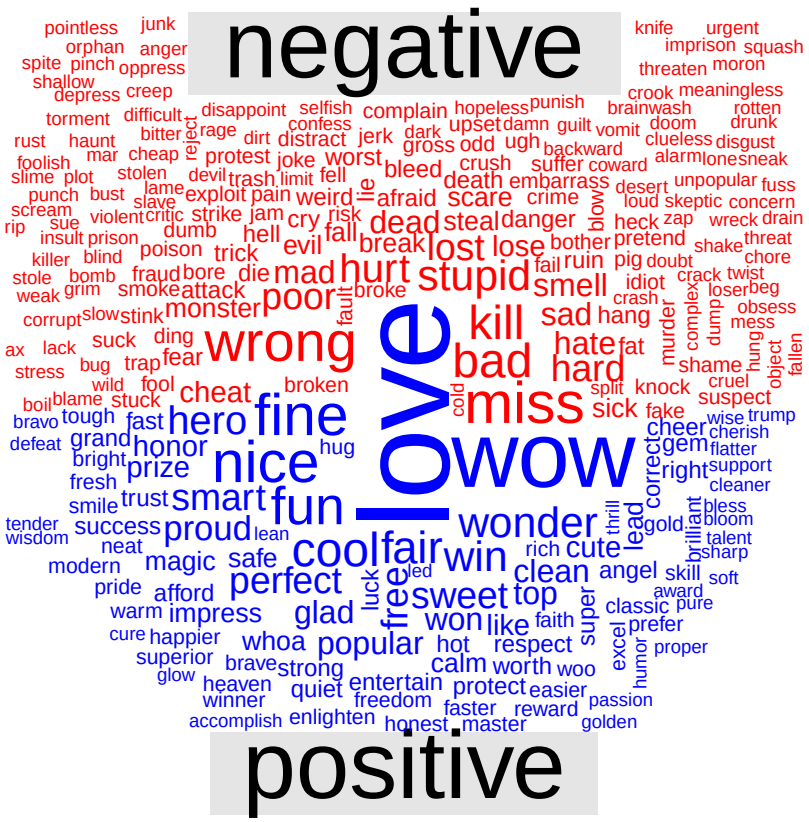


Sentimenti di Burns con nrc

Grafico 10: I sentimenti di Lisa

In questo caso è stato scelto di utilizzare un'altra tecnica, confrontando i sentimenti del personaggio attraverso una ComparisonCloud.

```
dati_spring %>%
  unnest_tokens(word, testo) %>%
  filter(character_id == 'Lisa') %>%
  filter(word != 'burn') %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("red", "blue"),
                   max.words = 300)
```



Questo studio ha fatto emergere parole in qualche modo affini alle personalità dei tre personaggi presi in considerazione, anche se sembrano venire privilegiati vocaboli positivi. Ciò è particolarmente sorprendente soprattutto per Mr. Burns, che potrebbe essere identificato come il cattivo della serie. Si nota poi che il linguaggio semplice e non forbito di Homer si ritrova tanto in situazioni negative ("stupid", "fat", "hate"), quanto in quelle positive ("love", "woo", "wow"), rispecchiando l'essenza del personaggio. L'opposto si verifica con Lisa: nella wordcloud si incontrano infatti numerosi termini inusuali per una bambina della sua età, come già si era visto nella sezione dedicata al tf-idf. In realtà, per apprezzare al meglio i risultati di questo tipo di analisi, bisognerebbe probabilmente considere più termini e capire il contesto in cui essi vengono utilizzati. E'il caso della celebre espressione "Excellent!" del signor Burns: tipicamente esclamata dal multimiliardario quando i suoi piani malvagi hanno successo, può essere classificata come positiva, se viene usata per esprimere soddisfazione per motivazioni così poco nobili?

Lda

Analisi preliminari condotte sui singoli personaggi con la *latent Dirichlet allocation* non hanno portato a risultati soddisfacenti. Questo è forse dovuto alla struttura della serie: essendo così lunga e varia, gli argomenti trattati da ciascun individuo sono molti e diversi e risulta impossibile individuarne uno predominante. Si è quindi deciso di utilizzare questa procedura per valutare se specifici luoghi di Springfield, in cui si svolgono la maggior parte dei dialoghi (casa Simpson, scuola elementare, taverna di Moe e centrale nucleare), sono caratterizzati dall'uso di certe parole e quindi dalla trattazione di certi argomenti.
Per farlo è stato creato un nuovo dataset, solo con le variabili di interesse *testo* , *episode_id*, *location_id*, *raw_location_text* e *word_count*, su cui sono state ripetute le procedure di normalizzazione e rimosse alcune stopwords individuate manualmente.

```
s_w_2 <- bind_rows(tibble(word = c('homer','marg','lisa',
                                   'bart','simpson','time'),
                          lexicon = rep('simpsons',6)),
                   s_w)
```

```
df_2 <- db %>%
  select(id, spoken_words) %>%
  unnest_tokens(output = word, input = spoken_words) %>%
  filter(!str_detect(word, "^[0-9]*$")) %>% # rimuove i numeri
  anti_join(get_stopwords()) %>%    # rimuove stopwords di snowball
  anti_join(s_w_2) %>%    # rimuove altre stopwords
  mutate(word = SnowballC::wordStem(word)) %>%   # stemming
  group_by(id) %>%
  summarise(testo = paste(word, collapse = ' ')) %>%
  left_join(db, by = 'id') %>%
  filter(!is.na(character_id)) %>%    # 1 obs
  filter(!is.na(location_id)) %>%     # 359 obs, per lo più in TV
  group_by(id) %>%
  summarise(testo, episode_id, location_id, raw_location_text,
            word_count = wordcount(testo))
```

Per cercare di ottenere una matrice meno sparsa e topic più caratterizzanti,

le battute sono state inizialmente raggruppate in episodi e poi per location.

```
dati_lda <- select(df_2, c(testo, location_id, episode_id)) %>%
  filter(location_id %in% c(5,3,15,10)) %>%
  mutate(location_id = as.factor(location_id))

# Gestione livelli fattore
levels(dati_lda$location_id)[levels(dati_lda$location_id)==
                               "5"] <- "Casa"
levels(dati_lda$location_id)[levels(dati_lda$location_id)==
                               "3"] <- "Scuola"
levels(dati_lda$location_id)[levels(dati_lda$location_id)==
                               "15"] <- "Taverna"
levels(dati_lda$location_id)[levels(dati_lda$location_id)==
                               "10"] <- "Centrale"
```

```
episodi_dtm <- parole_episodio %>%
  cast_dtm(doc, word, n)

episodi_dtm
```

```
## <<DocumentTermMatrix (documents: 1208, terms: 16119)>>
## Non-/sparse entries: 111522/19360230
## Sparsity           : 99%
## Maximal term length: 37
## Weighting          : term frequency (tf)
```

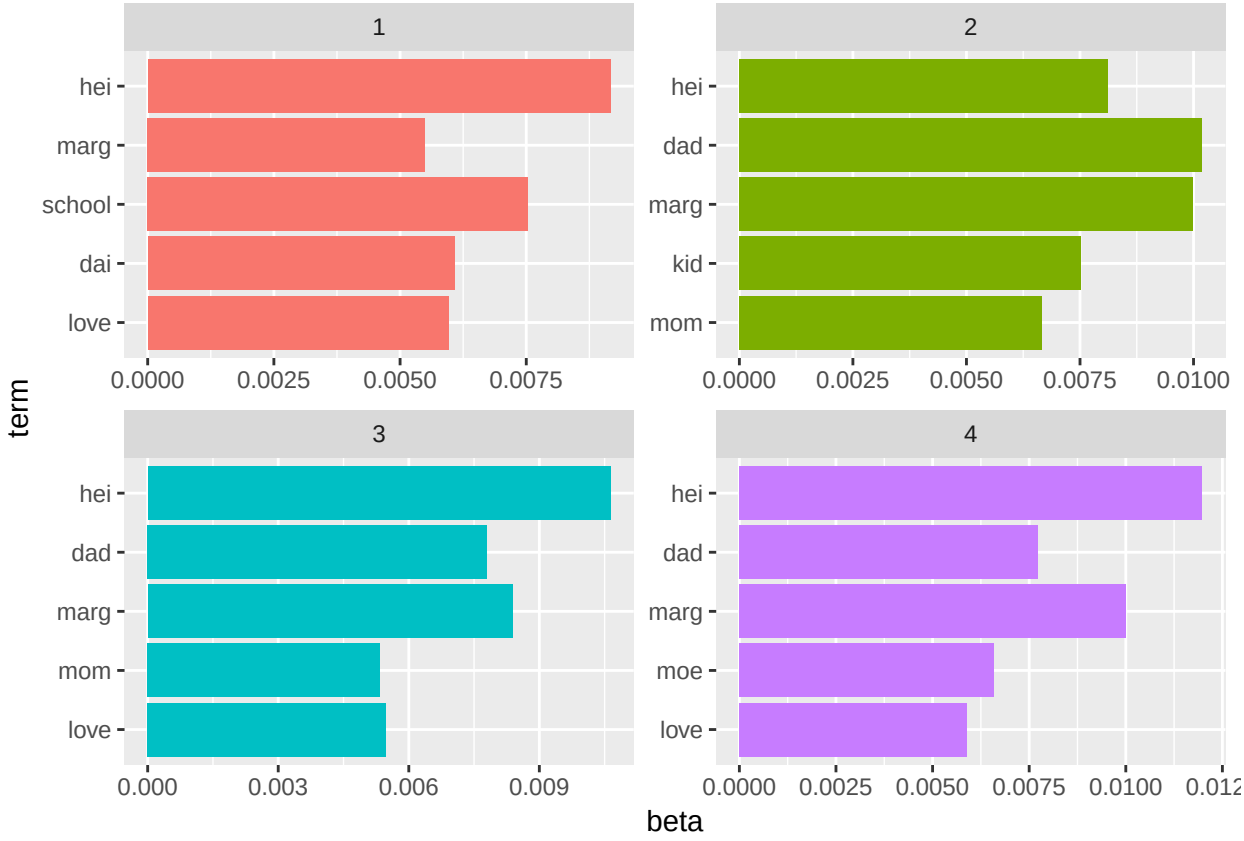
La matrice presenta un problema di elevata sparsità: questo è probabilmente dovuto al fatto che il numero di unità utilizzate è ancora troppo grande.

Di seguito viene comunque data una rappresentazione preliminare dei topic.

Grafico 11: Cinque parole più frequenti per topic

```
top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
```

```
  coord_flip()
```



I barplots ottenuti non sono indicativi: le parole più ricorrenti non permettono di distinguere i quattro topic.

Per provare ad ottenere risultati migliori, si è deciso di considerare come documenti le stagioni per intero.

```
dim(lda_db[lda_db$id=='Casa',]) # 542
dim(lda_db[lda_db$id=='Scuola',]) # 256
dim(lda_db[lda_db$id=='Taverva',]) # 240
dim(lda_db[lda_db$id=='Centrale',]) # 170

stagione <- bind_rows(arrange(as_tibble(apply(tibble(stag = 1:26),2,
                      function(x) rep(x,20))),stag),
                      tibble(stag = rep(27,22)),
                      arrange(as_tibble(apply(tibble(stag = 1:12),2,
                      function(x) rep(x,20))),stag),
                      tibble(stag = rep(13,16)),
```

```
                        arrange(as_tibble(apply(tibble(stag = 1:12),2,
                        function(x) rep(x,20))),stag),
                        arrange(as_tibble(apply(tibble(stag = 1:8),2,
                        function(x) rep(x,20))),stag),
                        tibble(stag = rep(9,10)))


db_spring <- bind_cols(lda_db, stagione) %>%
  select(-episode_id)

spring <- rbind(summarize(group_by(filter(db_spring,id ==
                                          'Casa'),stag),
          testo = paste(testo,collapse = ' '),id = 'Casa'),
                summarize(group_by(filter(db_spring,id ==
                                          'Scuola'),stag),
          testo = paste(testo,collapse = ' '),id = 'Scuola'),
                summarize(group_by(filter(db_spring,id ==
                                          'Taverna'),stag),
          testo = paste(testo,collapse = ' '),id = 'Taverna'),
                summarize(group_by(filter(db_spring,id ==
                                          'Centrale'),stag),
          testo = paste(testo,collapse = ' '),id = 'Centrale'))

docum <- paste(spring$stag, spring$id, sep = '_')
stag_lda <- as_tibble(cbind(spring, docum)) %>% select(-c(stag,id))
```

Ora è disponibile una tidy matrix con testi divisi per stagioni per i vari personaggi. Dopo la rimozione delle le stopwords essa viene trasformata in una DTM.

```
parole_stag <- stag_lda %>%
  unnest_tokens(word, testo) %>%
  anti_join(s_w) %>%
  count(docum, word, sort = TRUE) %>%
  ungroup()
```

```
mod_stargazer(parole_stag[1:10,], summary=FALSE)
```

Table 2:

	docum	word	n
1	22	marg	74
2	34	marg	74
3	34	dad	73
4	8	dad	59
5	38	dad	56
6	1	marg	54
7	19	duck	53
8	46	marg	53
9	42	dad	51
10	17	dad	49

```
# Trasformiamo la tibble in una DTM:
stag_dtm <- parole_stag %>%
  cast_dtm(docum, word, n)

stag_dtm
```

```
## <<DocumentTermMatrix (documents: 61, terms: 16110)>>
## Non-/sparse entries: 74592/908118
## Sparsity           : 92%
## Maximal term length: 37
## Weighting          : term frequency (tf)
```

La matrice presenta ora una sparsità del 92%.

Proseguendo quindi in questo senso è di seguito presentata l'LDA finale:

```
stag_LDA <- LDA(stag_dtm, k = 4, control = list(seed = 1234))

stag_topics <- tidy(stag_LDA, matrix = "beta")
mod_stargazer(stag_topics[1:10,], summary = FALSE,
              title = ' Per-topic-per-word probabilities ')
```

```
top_terms_stag <- stag_topics %>%
  filter(term != 'marg') %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
```

Table 3: Per-topic-per-word probabilities

	topic	term	beta
1	1	marg	0.0130084132414055
2	2	marg	0.00755350802159592
3	3	marg	0.0103355516797728
4	4	marg	0.000994437525730249
5	1	dad	0.0107062736538062
6	2	dad	0.00691453918821361
7	3	dad	0.0090570595011547
8	4	dad	0.00326424499024269
9	1	duck	0.00135833707291813
10	2	duck	0.000149261971903128

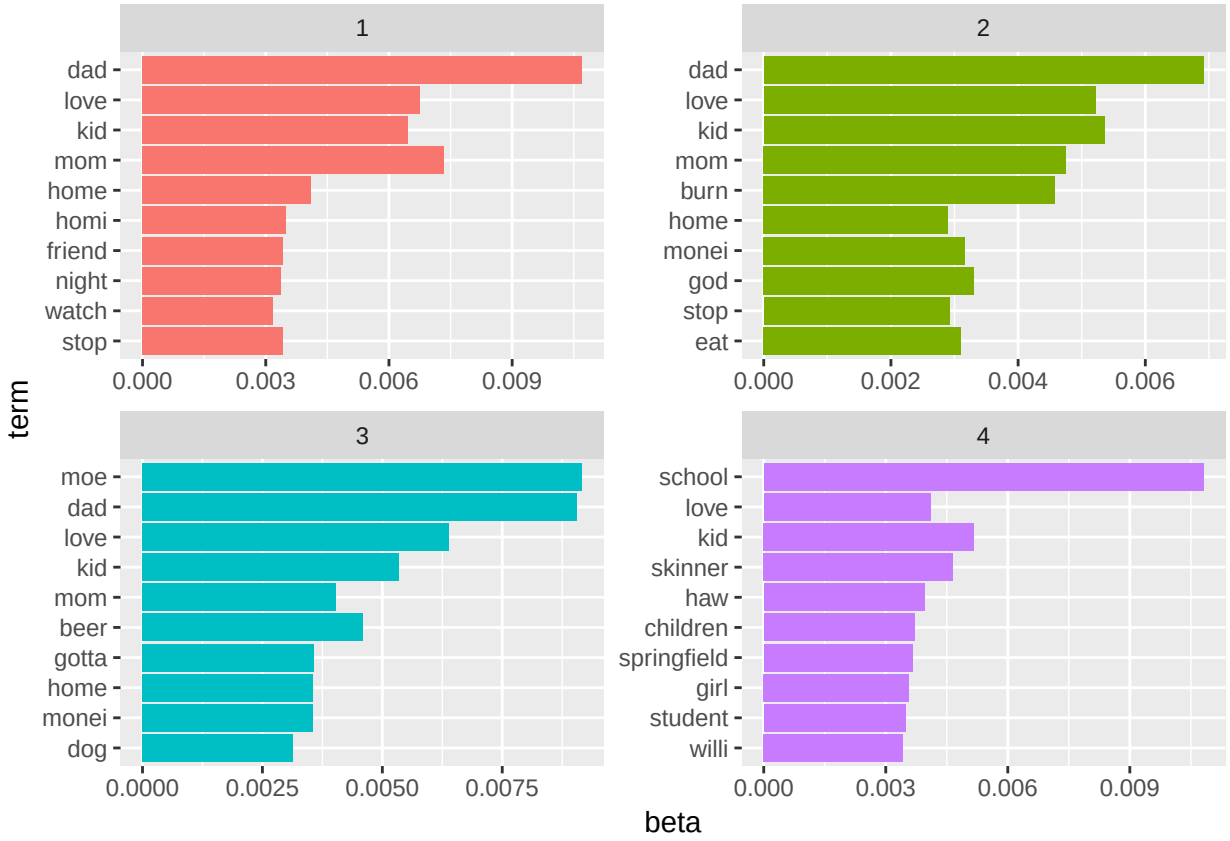
```
  ungroup() %>%
  arrange(topic, -beta)

mod_stargazer(top_terms_stag[1:10,], summary = FALSE,
              title=' 10 termini con beta più elevato ')
```

Table 4: 10 termini con beta più elevato

	topic	term	beta
1	1	dad	0.0107062736538062
2	1	mom	0.00733409800628356
3	1	love	0.00675970693731723
4	1	kid	0.00647350869373032
5	1	home	0.0041060700775576
6	1	homi	0.00350215127221079
7	1	stop	0.00342847865035281
8	1	friend	0.00341618540623051
9	1	night	0.0033678621335464
10	1	watch	0.00318880636418999

Grafico 12: Dieci parole più frequenti per topic

```
top_terms_stag %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



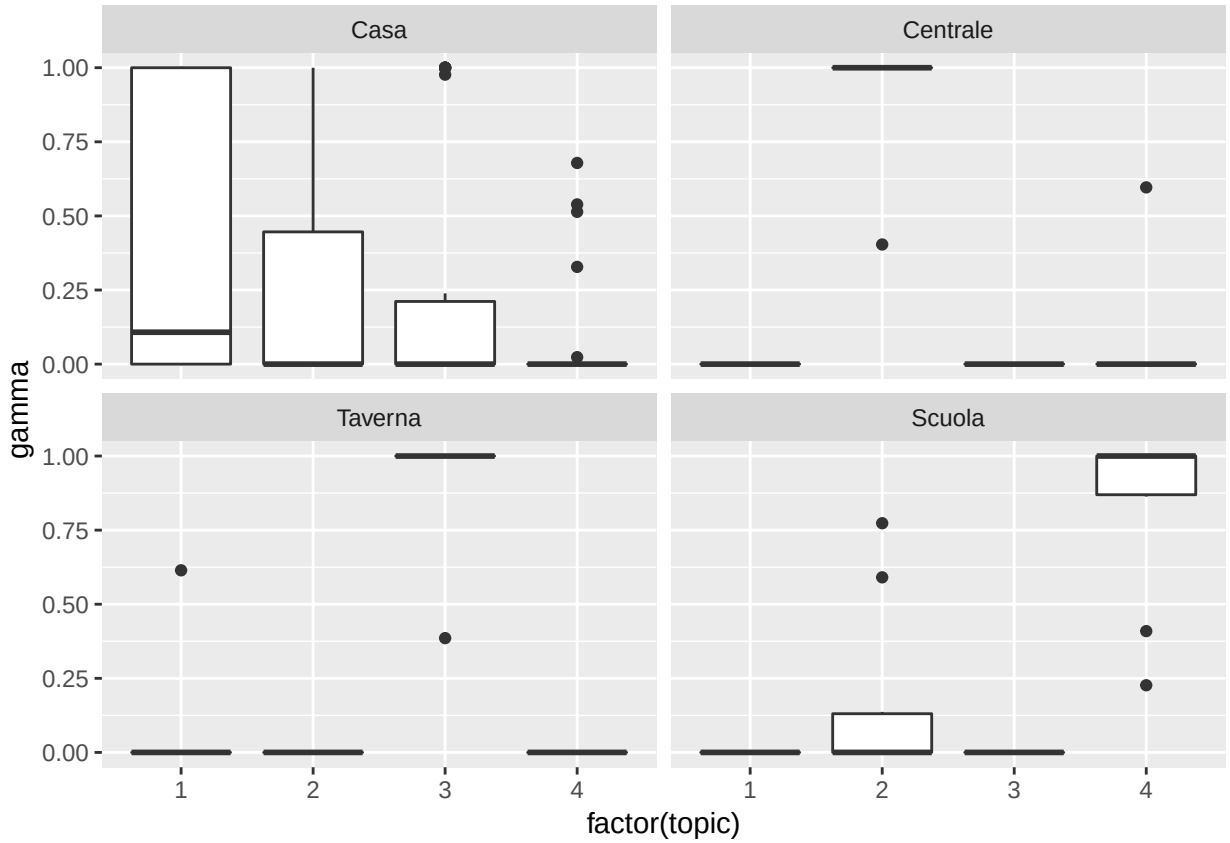

```
stag_gamma <- tidy(stag_LDA, matrix = "gamma")


stag_gamma <- stag_gamma %>%
  separate(document, c("stagione", "personaggio"), sep = "_",
           convert = TRUE)

mod_stargazer(stag_gamma[1:10,], summary = FALSE,
              title=' per-document-per-topic probabilities ')
```

Grafico 13: Distribuzione Topics

```
stag_gamma %>%
  mutate(personaggio = reorder(personaggio, gamma * topic)) %>%
  ggplot(aes(factor(topic), gamma)) +
  geom_boxplot() +
  facet_wrap(~ personaggio)
```

Table 5: per-document-per-topic probabilities

	stagione	personaggio	topic	gamma
1	1	Casa	1	0.999979137253918
2	1	Centrale	1	1.52712671381292e-05
3	1	Scuola	1	1.29835556717022e-05
4	1	Taverna	1	3.06306154628557e-05
5	10	Casa	1	0.0823873028806085
6	10	Scuola	1	1.12613965751853e-05
7	10	Taverna	1	1.35238044446458e-05
8	11	Casa	1	0.81612716252946
9	11	Scuola	1	1.450455344171e-05
10	11	Taverna	1	1.9584846458804e-05



Quest'ultimo grafico evidenzia come il secondo topic sia tipico della centrale elettrica, il terzo della taverna di Moe, il quarto della scuola. In casa Simpson, com'è normale che sia, si parla un po' di tutto.

Modellistica

L'obiettivo di quest'ultima fase di analisi è quello di riuscire a classificare le unità statistiche (battute di ciascun personaggio, raccolte per episodio) nei rispettivi quattro personaggi (Burns, Moe, Krusty e Ned), ovvero i quattro possibili livelli della variabile risposta, indicata con *character_id*, sfruttando principalmente i testi delle battute di ciascuno. Nello specifico, le variabili esplicative utilizzate sono:

- **location_id** (moda per personaggio per episodio della variabile **location_id** iniziale) che rappresenta il luogo in cui ciascuno personaggio parla più spesso nell'i-esimo episodio;

- 96 variabili (96/4 = 24) rappresentanti il conteggio delle parole più caratterizzanti per ciascun personaggio, ottenute confrontando i valori **tf-idf** più elevati;

- **tot_words** (numero di parole presenti nei testi di ogni unità statistica).

La scelta di considerare Mr. Burns, Moe, Krusty e Ned è dovuta al ruolo centrale che hanno nella serie, ma soprattutto alla loro particolarità.

```
dati_mod <- read_csv('simpsons_script_lines.csv',T) %>%
  filter(speaking_line == 'TRUE') %>%
  select(c(normalized_text, character_id, episode_id, location_id)) %>%
  drop_na(normalized_text) %>%
  filter(character_id %in% c(15,17,139,11)) %>%
  mutate(character_id = as.factor(character_id))
# risposta = 'character_id'
```

```
# Gestione livelli fattore
levels(dati_mod$character_id)[levels(dati_mod$character_id)==
                                "15"]<- "Burns"
levels(dati_mod$character_id)[levels(dati_mod$character_id)==
                                "17"]<- "Moe"
levels(dati_mod$character_id)[levels(dati_mod$character_id)==
                                "139"]<- "Krusty"
levels(dati_mod$character_id)[levels(dati_mod$character_id)==
```

```
                                    "11"]<- "Ned"
```

```
# Accorpamento dei testi delle battute in episodi
# e conseguente gestione del predittore 'location_id'

spring_db <- rbind(summarize(group_by(filter(dati_mod,character_id==
                                                 'Burns'),episode_id),
                   testo = paste(normalized_text,collapse = ' '),
                   location_id = max(table(location_id)), id = 'Burns'),
             summarize(group_by(filter(dati_mod,character_id==
                                         'Moe'),episode_id),
                   testo = paste(normalized_text,collapse = ' '),
                   location_id = max(table(location_id)), id = 'Moe'),
             summarize(group_by(filter(dati_mod,character_id==
                                         'Krusty'),episode_id),
                    testo = paste(normalized_text,collapse = ' '),
                    location_id = max(table(location_id)), id = 'Krusty'),
             summarize(group_by(filter(dati_mod,character_id==
                                         'Ned'),episode_id),
                     testo = paste(normalized_text,collapse = ' '),
                     location_id = max(table(location_id)), id = 'Ned'))

documento <- paste(spring_db$episode_id, spring_db$id, sep = '_')
spring_db <- as_tibble(cbind(spring_db, documento)) %>%
  filter(location_id != -Inf)

#Creazione lista stopwords
data("stop_words")
s_w <- bind_rows(tibble(word = c('hey','im','homer','dont','youre',
                                 'ill','gonna','ive','uh','yeah','simpson'),
                                 lexicon = rep('simpsons',11)),
                 stop_words)
```

```
dim(mod_db)
```

```
## [1] 1128   98
```

Modello Multinomiale

Inizialmente si è scelto di utilizzare il modello multinomiale come classificatore, sia per la sua fama e semplicità, sia perché è sembrato adeguato, visto il rapporto tra unità statistiche e predittori. Per avere una "valida" stima dell'accuratezza, come da prassi, il dataset è stato suddiviso in un insieme di stima (60%) e in uno di verifica (40%).

```
trainObs <- sample(nrow(mod_db), .6 * nrow(mod_db), replace = FALSE)
testObs <- sample(nrow(mod_db), .4 * nrow(mod_db), replace = FALSE)
train_dat <- mod_db[trainObs,]
test_dat <- mod_db[testObs,]
```

Si è poi stimato il modello:

```
model <- multinom(id ~., data = train_dat)
```

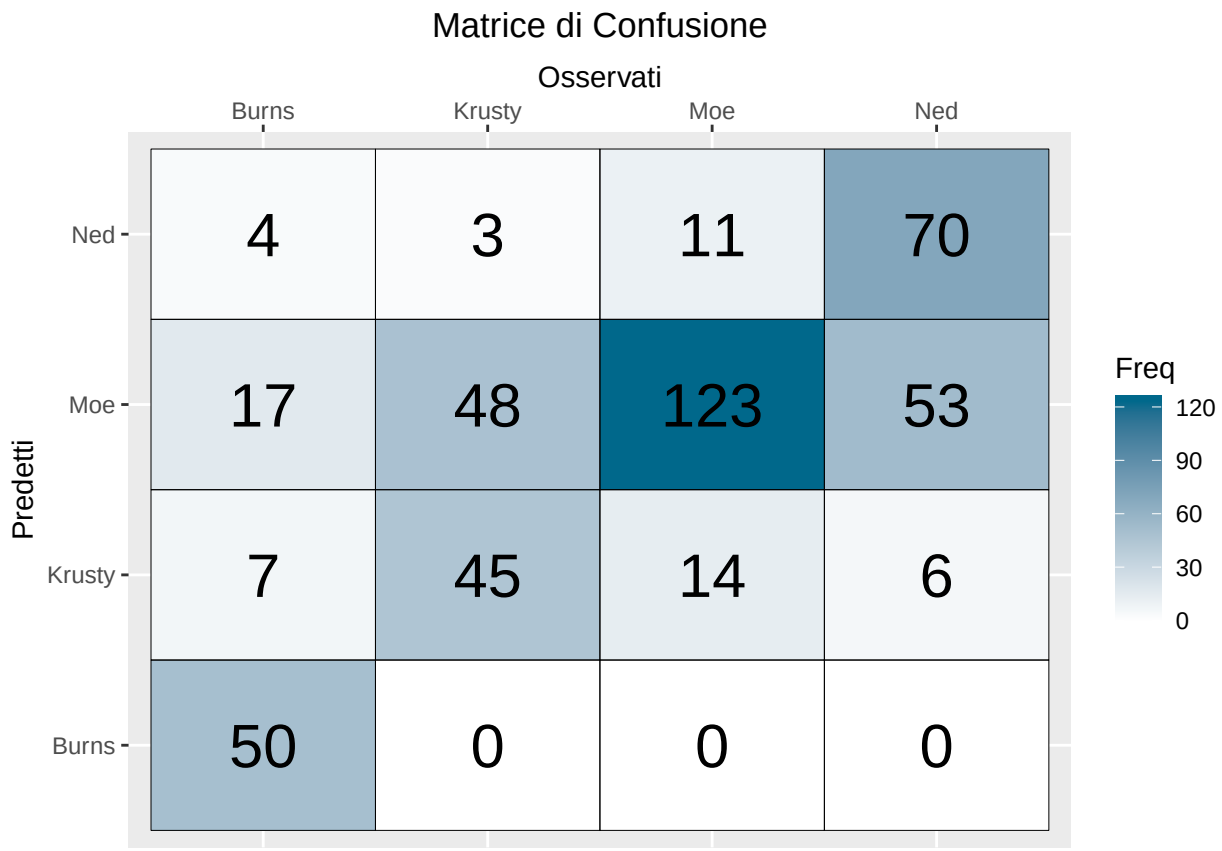
```
## # weights:  396 (294 variable)
## initial  value 937.134988
## iter  10 value 647.833466
## iter  20 value 514.730806
## iter  30 value 462.854571
## iter  40 value 457.347732
## iter  50 value 456.616597
## iter  60 value 456.592234
## iter  70 value 456.582671
## iter  80 value 456.580514
## final  value 456.580501
## converged
```

```
predicted.classes <- model %>% predict(test_dat)

mean(predicted.classes == test_dat$id)
```

```
## [1] 0.6385809
```

L'accuratezza stimata sull'insieme di verifica si aggira intorno al 66%, indicando un discreto potere predittivo del modello.

```
cm_stg0 <- confusionMatrix(predicted.classes, test_dat$id)

cm_stg0$table %>%
  data.frame() %>%
  rename(Osservati = Reference) %>%
  rename(Predetti = Prediction) %>%
  group_by(Osservati) %>%
  mutate(total = sum(Freq)) %>%
  ungroup() %>%
  ggplot(aes(Osservati, Predetti, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), size = 8) +
  scale_fill_gradient(low = "white", high = "deepskyblue4") +
  scale_x_discrete(position = "top") +
  geom_tile(color = "black", fill = "black", alpha = 0) +
  ggtitle('Matrice di Confusione') +
  theme(plot.title = element_text(hjust = 0.5))
```



Modello Random Forest

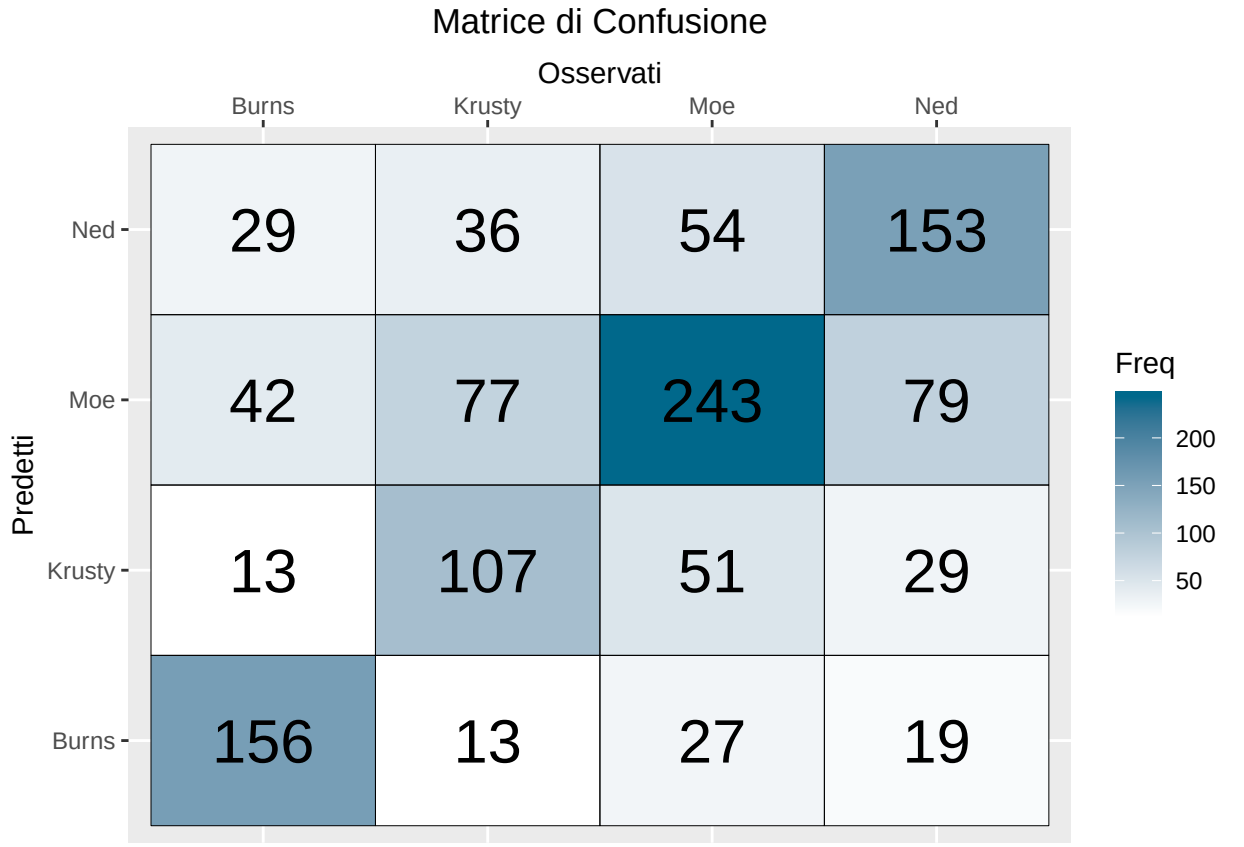
Con un modello random forest si è provato a vedere se un approccio black-box potesse dare risultati migliori del modello multinomiale, data l'elevata complessità del contesto in cui si sta lavorando.

```
set.seed(7)
rf_clf_stag <- rfsrc(id~., data = mod_db,
                     ntree = 1000, # 1000 alberi di classificazione
                     mtry =  10, # 10 predittori per ogni split
                     nodesize = 1,
                     nsplit = 0, # fa tutti gli split possibili
                     importance = TRUE,
                     block.size = NULL,
                     ensemble = "oob",
                     bootstrap = "by.root",
                     samptype = "swr",
                     na.action = "na.impute", #sostituisce i NA
                                              #efficacemente
                     nimpute = 1)

cm_stg <- confusionMatrix(rf_clf_stag$class.oob, mod_db$id)

# In questo caso l'accuracy è del 64%, mentre l'AUC è pari a 0.8.
```

Di seguito la matrice di confusione.

```
cm_stg$table %>%
  data.frame() %>%
  rename(Osservati = Reference) %>%
  rename(Predetti = Prediction) %>%
  group_by(Osservati) %>%
  mutate(total = sum(Freq)) %>%
  ungroup() %>%
  ggplot(aes(Osservati, Predetti, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), size = 8) +
  scale_fill_gradient(low = "white", high = "deepskyblue4") +
```

```
  scale_x_discrete(position = "top") +
  geom_tile(color = "black", fill = "black", alpha = 0) +
  ggtitle('Matrice di Confusione') +
  theme(plot.title = element_text(hjust = 0.5))
```



L'accuracy stimata è ora del 64%: questa diminuzione potrebbe indicare un effettivo migliore adattamento del modello multinomiale ai dati, ma non si esclude che possa essere causata da una stima più realistica dell'accuratezza da parte del modello random forest, rispetto a quella data dal multinomiale (in quanto stimata su 1000 campioni out-of-bag, invece che su un singolo insieme di verifica).

Dalla matrice di confusione si vede come il personaggio con il minor errore di classificazione sia Moe, seguito da Burns, Ned e per ultimo Krusty, come accadeva per il modello multinomiale.

Commenti ai risultati e possibili sviluppi

L'analisi delle frequenze delle parole ha evidenziato come i termini più utilizzati nella serie comprendano i nomi di tutti i componenti della famiglia Simpson, o siano strettamente legati ad essi, mostrandosi quindi precisa ed efficace. Questo è anche dovuto al fatto che Homer, Marge, Bart e Lisa sono i personaggi che parlano in assoluto di più, influenzando quindi maggiormente il contenuto dei discorsi.

L'indice *tf-idf* ha permesso di individuare i vocaboli più caratteristici di tutti i personaggi considerati. L'elevata capacità discriminatoria di questa procedura può essere attribuita al fatto che la serie animata oggetto di studio presenta individui con personalità e caratteristiche del linguaggio molto specifiche, se non, in molti casi, uniche.

La sentiment analysis, sia attraverso bing che con nrc, ha evidenziato tanto in Homer quanto nel signor Burns la prevalenza di un sentiment positivo, che si distribuisce tra gioia, aspettativa, ottimismo e fiducia. Se questo non è sorprendente per Homer, lo è per Burns: egli è infatti il suo dispotico capo, uomo malvagio e senza scrupoli da cui ci si aspetterebbe l'opposto.

Per ottenere migliori risultati con la LDA si è deciso di considerare come singolo documento una stagione, dal momento che le battute contenevano troppi pochi termini. La scelta di considerare i luoghi si è rivelata ottima: il boxplot (**Grafico 11**) fa emergere argomenti specifici per ciascun ambiente, a meno di casa Simpson dove si riscontra una maggiore eterogeneità dei dialoghi. Questo può essere attribuito al fatto che nelle serie animate come I Simpson, luoghi come scuola, posto di lavoro e bar vengono spesso strereotipati.

I modelli adattati ai dati hanno prodotto risultati abbastanza soddisfacenti: se i personaggi fossero stati classificati casualmente, l'accuratezza sarebbe stata circa pari al 25%, di gran lunga inferiore a quella ottenuta classificandoli attraverso le due procedure. Anche il valore dell'area sotto la curva ROC, circa 0.8 per entrambi i modelli, lo conferma, indicando una buona capacità predittiva. L'accuracy del modello multinomiale è risultata superiore a quella della random forest ma, come detto in precedenza, questa differenza può essere dovuta ai diversi procedimenti per la stima dell'accuratezza dei due modelli. Di conseguenza non è decretabile in modo netto quale sia la miglior regola predittiva.

Come possibili sviluppi, si potrebbero approfondire altri aspetti di questa serie animata. Per esempio, si potrebbe realizzare una rete tra i personaggi, per ottenere una panoramica dei cittadini di Springfield e della loro centralità nella storia. Si potrebbe poi analizzare il sentiment legato a certi luoghi della città, o ancora coinvolgere nello studio personaggi minori.
Le possibilità sono molteplici, data la ricchezza del dataset e la notorietà del cartone.