Análise de Sentimento em Tweets: debate GOP e os candidatos às Eleições Presidenciais nos Estados Unidos

Miguel Sandim e Paula Fortuna

Faculdade de Engenharia da Universidade do Porto,
{miguel.sandim, paula.fortuna}@fe.up.pt

1 Introdução

O presente trabalho insere-se no âmbito da Unidade Curricular Extração de Conhecimento de Dados II e tem como objetivo geral construir um sistema de classificação capaz de atribuir um valor de sentimento a um determinado conjunto de documentos.

A solução elaborada recorre à linguagem R e segue, de forma genérica, o processo delineado no conjunto de slides fornecido na unidade curricular "Classification of documents with tm in R".

Este relatório está dividido em várias secções: revisão teórica e objectivo, método, resultados e análise, conclusões e referências.

2 Revisão Teórica e Objetivo

Neste capitulo pretende fazer-se uma breve revisão teórica sobre os conceitos explorados no trabalho, tais como *text mining* e *sentiment analysis*. Para além disso, apresenta-se ainda, de forma mais aprofundada, o objetivo do trabalho.

A área de *text mining* tem vindo a adquirir maior relevância nos últimos anos, dado a crescente utilização de redes sociais, fóruns, *e-commerce* e jornais via web. Assim, se os utilizadores procuram mais a web para expressarem as suas opiniões sobre vários assuntos, existe uma grande quantidade de informação disponível, que pode ser recolhida e analisada para diversos fins. Neste sentido, cada vez mais se torna pertinente que se desenvolvam algoritmos para análise e interpretação dessa informação de forma sistemática e eficiente. Pretende-se neste trabalho aprofundar um exemplo de utilização de *text mining* e, por isso, é importante definir-se em que consiste essa área.

Text mining refere-se ao processo de extrair informação e conhecimento interessante e não trivial de texto não estruturado. É um campo recente e interdisciplicar que recebe influência de áreas tais como *information retrieval*, *data mining*, *machine learning*, estatística e *computational linguistics* [1]. A maior dificuldade ao aplicar *text mining* reside em extrair conceitos explicitos e implícitos e relações semânticas entre os conceitos [1]. As principais áreas de análise de *text*

mining são: extração de informação, identificação de tópico, sumarização, categorização, *clustering*, conexão de conceitos, visualização de informação e resposta de questões [1].

Ainda na área de *Text Mining*, existem algoritmos de *Sentiment Analysis*, que têm também adquirido maior relevância, dado o grande potencial para gerar valor, deste tipo de técnicas [2]. *Sentiment analysis* consiste em identificar opiniões em textos e classificar excertos de texto com um sentimento (e.g. positivo, negativo ou neutro) [3].

Neste sentido, o objetivo deste trabalho é explorar a área de *Sentiment Analysis* e construir um sistema de classificação capaz de atribuir um valor de sentimento a um determinado conjunto de documentos. Nos próximos capítulos, apresenta-se a metodologia seguida para cumprir este objetivo.

3 Metodologia

3.1 Dataset

O conjunto de dados utilizado estava disponibilizado na plataforma "Kaggle" e provém originalmente da biblioteca "Crowdflower's Data for Everyone" [4]. Para classificação de sentimento foram usados 16654 "tweets" escritos em inglês sobre o debate GOP de agosto de 2015, no âmbito das eleições para as presidenciais nos Estados Unidos da América, recolhidos no Twitter. Estes "tweets" estavam classificados por *workers* do "Crowdflower". Desta forma, é possível avaliar e validar o método utilizado na classificação e o resultado obtido para cada "tweet".

3.2 Features

O *dataset* utilizado apresentava diversas *features*. Para o estudo de sentimento foram utilizadas as seguintes:

- *Candidate* - Candidato a que o "tweet" se refere.
- *Candidate_confidence* - Confiança na informação contida em *Candidate*.
- *Subject_matter* - Assunto a que o "tweet" se refere.
- *Subject_matter_confidence* - Confiança na informação contida em *Subject_matter*.
- *Text* - Texto do "tweet".
- *Sentiment* - Sentimento atribuído ao "tweet".
- *Sentiment_confidence* - Confiança na informação contida em *Sentiment*.

Para além destas *features*, é ainda de salientar que foram utilizados dicionários de sentimento, em diversas experiências, tendo sido obtidas outras *features* inerentes à sua utilização. As diversas experiências e métricas obtidas serão detalhadas em capítulos posteriores.

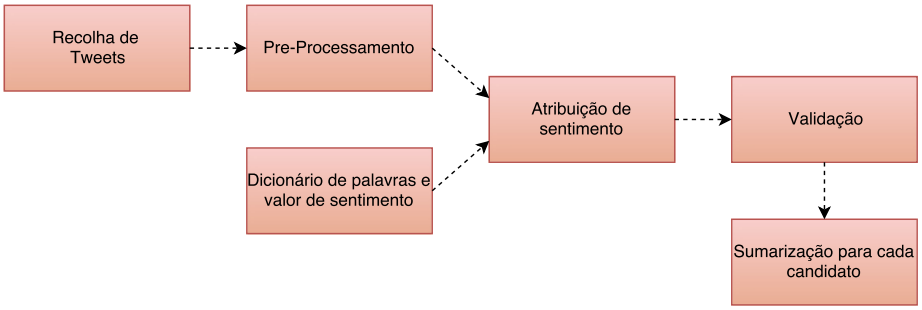

Figura 1. Pipeline do processo de análise do sentimento de "tweets" seguido durante o curso deste projeto.

3.3 *Pipeline* para Análise de Sentimento

Para realizar o processo de análise de sentimento foi seguida a *pipeline* esquematizada na figura 1.

Neste capítulo será apresentado mais em detalhe cada um dos passos seguidos. Salienta-se que a análise foi levada a cabo em R e serão também apresentados alguns extratos dos *scripts* executados.

3.3.1 Recolha de "Tweets"

Nesta experiência o processo de recolha de "tweets" foi simplificado, dado que foi utilizado um *dataset* com "tweets" classificados. Desta forma, é possível validar o modelo de atribuição de sentimento desenvolvido.

3.3.2 Pré-Processamento

A fase de pré processamento é fundamental em *text mining*, uma vez que nesta se trabalha com dados não estruturados. Esta fase vai procurar identificar unigramas de forma sistemática, bem como eliminar algumas palavras e elementos no texto, para que em fases posteriores da *pipeline* seja possível aplicar algumas análises. Neste caso procedeu-se a:

- Remoção de caracteres não ASCII;
- Remoção de entidades de "retweet";
- Remoção de identificação de perfis do "Twitter";
- Remoção de pontuação;
- Remoção de números;
- Remoção de links HTML;
- Conversão do texto para minúsculas;
- *Stemming*;
- Remoção dos "tweets" classificados com sentimento neutro.

O código em R utilizado é apresentado de seguida:

```r
library(tm)

preProcessing <- function(stem)
{
  ######################################
  #            Prepare data
  ######################################
  #read the csv
  original_data <- read.csv("csv_data/GOF_data.csv")

  #delete not needed columns

  original_data <- data.frame("id" = original_data$id,
                              "text" = original_data$text,
                              "candidate" = original_data$candidate,
                              "candidate_confidence" =
 ↪   original_data$candidate_confidence,
                              "subject_matter" = original_data$subject_matter,
                              "subject_matter_confidence" =
 ↪   original_data$subject_matter_confidence,
                              "sentiment" = original_data$sentiment,
                              "sentiment_confidence" =
 ↪   original_data$sentiment_confidence)

  original_data$text <- as.character(original_data$text)

  ################################
  ##### Text cleaning ############
  ################################
  # Remove non-ASCII characters:
  Encoding(original_data$text) <- "latin1"
  original_data$text <- iconv(original_data$text, "latin1", "ASCII", sub="")

  ## Twitter related:
  # Remove retweet entities
  original_data$text <- gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", original_data$text)
  # Remove @ people
  original_data$text <- gsub("@\\w+", "", original_data$text)
  # Remove html links
  original_data$text <- gsub("http[^[:space:]]*", "", original_data$text)

  ## Other:
  tm_text <- Corpus(VectorSource(original_data$text))

  # Remove numbers:
  tm_text <- tm_map(tm_text, removeNumbers)
  # To lowercase:
  tm_text <- tm_map(tm_text, content_transformer(tolower))
  # Remove ponctuation:
  tm_text <- tm_map(tm_text, removePunctuation)
  # Remove whitespace:
  tm_text <- tm_map(tm_text, stripWhitespace)
  # Text stemming:
  if (stem)
    tm_text <- tm_map(tm_text, stemDocument)

  original_data$text <- unlist(sapply(tm_text, `[`, "content"))

  ######################################
  ######## Filter data rows ############
  ######################################
  # Delete neutrals
  original_data <- original_data[original_data$sentiment != "Neutral",]
  original_data$sentiment <- factor(original_data$sentiment)
  # Process no candidate mentioned
  original_data$candidate[original_data$candidate == ""] <- "No candidate mentioned"
```

```
  original_data$candidate <- factor(original_data$candidate)

  return(original_data)
}

original_data <- preProcessing(stem = FALSE)
original_data_stemming <- preProcessing(stem = TRUE)
```

É de salientar que foram removidos os "tweets" classificados como neutros, uma vez que está demonstrado na literatura que este sentimento é mais difícil de detetar [5].

3.3.3 Atribuição de sentimento

Para atribuição de sentimento foram usados diferentes algoritmos, baseados num mesmo princípio que se passa a descrever. A estratégia mais comum para análise de sentimento consiste em confrontar as palavras contidas no textos com dicionários de sentimento. Nestes dicionários existem listas de palavras para as quais é definido um sentimento positivo ou negativo e com as quais se confronta cada uma das palavras do texto que se quer classificar. Depois de saber o sentimento de cada palavra do texto, calcula-se uma métrica global para o conjunto de palavras. Passamos a apresentar as várias estratégias desenvolvidas, recorrendo a diferentes dicionários.

3.3.3.1 Sentimento com base no dicionário de Liu e Hu [6]

Numa primeira experiência foi implementado um script de análise de sentimento manual, seguiu-se o seguinte algoritmo:

Algoritmo 1 Algoritmo de análise de sentimento.

for all $t \in tweets$ **do**
  $numPositives \leftarrow 0$
  $numNegatives \leftarrow 0$
  for all $word \in t$ **do**
    if isPositive($word$) **then**
      $numPositives \leftarrow NumPositives + 1$
    else if isNegative($word$) **then**
      $numNegatives \leftarrow numNegatives + 1$
    end if
  end for
  $sentimentTweet \leftarrow numPositives - numNegatives$
end for

Apresenta-se o código de R que permitiu implementar este algoritmo:

```
sentimentManual <- function(data, stem)
{
  pos.words <- read.table("csv_data/positive-words.txt")
  neg.words <- read.table("csv_data/negative-words.txt")
```

```
  #evaluation tweets function
  score.sentiment <- function(sentences, pos.words, neg.words, .progress='none')
  {
    scores <- laply(sentences, function(sentence, pos.words, neg.words){
      word.list <- strsplit(sentence, " ")
      words <- unlist(word.list)
      pos.matches <- match(words, pos.words$V1)
      neg.matches <- match(words, neg.words$V1)
      pos.matches <- !is.na(pos.matches)
      neg.matches <- !is.na(neg.matches)
      score <- sum(pos.matches) - sum(neg.matches)
      return(score)
    }, pos.words, neg.words, .progress=.progress)

    scores.df <- data.frame("score" = scores, "text" = sentences)
    return(scores.df)
  }

  sentiment_scores_manual <- score.sentiment(data$text, pos.words, neg.words,
↪   .progress='text')

  # Build filename:
  if (stem)
    fileName <- "csv_data/sentiment_manual_stemming.csv"
  else
    fileName <- "csv_data/sentiment_manual.csv"

  write.csv(sentiment_scores_manual, file = fileName, row.names=FALSE)
}

sentimentManual(original_data, stem = FALSE)
sentimentManual(original_data_stemming, stem = TRUE)
```

3.3.3.2 Sentimento com base na função polarity do package "QDAP", com os parâmetro default [7]

Numa segunda experiência, utilizou-se o *package* "QDAP" de R e a função *polarity* para análise de sentimento. Esta função engloba uma abordagem mais completa que a anteriormente apresentada. Utilizando também o dicionário de Hu e Liu [6], é feito uma aproximação a um *cluster* de contexto através das quatro palavras anteriores e as duas seguintes. Nesta vizinhança é avaliada a presença de palavras neutras, negadores, amplificadores e de-amplificadores. As palavras neutras não têm valor na equação, mas afetam a contagem de palavras e por isso também são contabilizadas. É de notar que este algoritmo considera que, se aparecer um sinal de pontuação como uma vírgula ou um ponto, as palavras seguintes não serão consideradas vizinhança.

Apresenta-se, no final da secção 3.3.3, o código de R que permitiu implementar este algoritmo, conjuntamente com o algoritmo da próxima subsecção.

3.3.3.3 Sentimento com base na função polarity do package "QDAP", utilizando o dicionário AFINN [8]

Para além da experiência citada no capítulo anterior, procurou-se utilizar outro dicionário mais atual e mais adequado ao contexto das redes sociais e "tweets". O dicionário escolhido foi o AFINN-111, versão mais recente, com 2477 palavras e expressões em inglês (tais como "wow", "wowow", "lol"), avaliadas

com um inteiro compreendido entre menos cinco (valência negativa) e cinco (valência positiva).

Apresenta-se de seguida o código de R que permitiu utilizar a função *polarity*, para implementar os dois últimos algoritmos apresentados, variando os seus parâmetros:

```
sentimentQdap <- function(data, stem)
{
  dictionary_AFINN <- read.csv("csv_data/AFINN-111.txt", stringsAsFactors = FALSE,
 ↪   header=FALSE, sep="\t")

  pol.df_Default <- polarity(original_data$text)$all
  pol.df_AFINN <- polarity(original_data$text, polarity.frame = dictionary_AFINN)$all

  #add this result to the data
  sentiment_scores_qdap_default <- data.frame("score" = pol.df_Default$polarity,
                                              "text" = pol.df_Default$text)
  #add this result to the data
  sentiment_scores_qdap_afinn <- data.frame("score" = pol.df_AFINN$polarity,
                                            "text" = pol.df_AFINN$text)

  # Build filenames:
  if (stem)
  {
    defaultFileName <- "csv_data/sentiment_qdap_default_stemming.csv"
    afinnFileName <- "csv_data/sentiment_qdap_afinn_stemming.csv"
  }
  else
  {
    defaultFileName <- "csv_data/sentiment_qdap_default.csv"
    afinnFileName <- "csv_data/sentiment_qdap_afinn.csv"
  }

  write.csv(sentiment_scores_qdap_default, file = defaultFileName, row.names=FALSE)
  write.csv(sentiment_scores_qdap_afinn, file = afinnFileName, row.names=FALSE)
}

sentimentQdap(original_data, stem = FALSE)
sentimentQdap(original_data_stemming, stem = TRUE)
```

3.3.4 Validação

Para avaliar e validar a utilização de cada um dos algoritmos apresentados, foram utilizadas as medidas de precision, recall e F1, assim como análise da confusion matrix.

Essas métricas foram obtidas através do seguinte *script* em R:

```
library(caret)

# ######################################
# ######## PREPARE VALIDATION ##########
# ######################################

# Read results saved before:
sentiment_scores_manual <- read.csv("csv_data/sentiment_manual.csv")
sentiment_scores_manual_stemming <- read.csv("csv_data/sentiment_manual_stemming.csv")
sentiment_scores_qdap_default <- read.csv("csv_data/sentiment_qdap_default.csv")
sentiment_scores_qdap_afinn <- read.csv("csv_data/sentiment_qdap_afinn.csv")
sentiment_scores_qdap_default_stemming <-
 ↪   read.csv("csv_data/sentiment_qdap_default_stemming.csv")
```

```
sentiment_scores_qdap_afinn_stemming <-
 ↪   read.csv("csv_data/sentiment_qdap_afinn_stemming.csv")

# Convert the numeric sentiment values to classes:
convertSentimentFactor <- function(data)
{
  data$sentiment[data$score >= 0] <- "Positive"
  data$sentiment[data$score < 0] <- "Negative"
  data$sentiment <- as.factor(data$sentiment)
  return(data)
}

sentiment_scores_manual <- convertSentimentFactor(sentiment_scores_manual)
sentiment_scores_manual_stemming <-
 ↪   convertSentimentFactor(sentiment_scores_manual_stemming)
sentiment_scores_qdap_default <- convertSentimentFactor(sentiment_scores_qdap_default)
sentiment_scores_qdap_afinn <- convertSentimentFactor(sentiment_scores_qdap_afinn)
sentiment_scores_qdap_default_stemming <-
 ↪   convertSentimentFactor(sentiment_scores_qdap_default_stemming)
sentiment_scores_qdap_afinn_stemming <-
 ↪   convertSentimentFactor(sentiment_scores_qdap_afinn_stemming)

########################################
# ######## Evaluation Metrics ############
########################################

# Produces several validation statistics:
# y: Factor vector of true values
# predictions: Factor vector of the predicted values
# positiveClass: Name, as a string, of the positive class
validationStatistics <- function (predictions, y, positiveClass)
{
  # Calculate precision and recall:
  result <- confusionMatrix(predictions, y, positiveClass)
  print(result)
  precision <- result$byClass[['Pos Pred Value']]
  recall <- result$byClass[['Sensitivity']]

  # Calculate F1:
  f1 <- 2 * precision * recall / (precision + recall)

  return(list(precision = precision, recall = recall, f1 = f1))
}

################################
# 1 - Precision, Recall, F1:
################################
# Validate manual approach:
res_man <- validationStatistics(sentiment_scores_manual$sentiment,
 ↪   original_data$sentiment, "Positive")
# Validate manual approach (stemming):
res_man_stem <- validationStatistics(sentiment_scores_manual_stemming$sentiment,
 ↪   original_data_stemming$sentiment, "Positive")
# Validate qdap default approach:
res_qdap_deafault <- validationStatistics(sentiment_scores_qdap_default$sentiment,
 ↪   original_data$sentiment, "Positive")
# Validate qdap afinn approach:
res_qdap_afinn <- validationStatistics(sentiment_scores_qdap_afinn$sentiment,
 ↪   original_data$sentiment, "Positive")
# Validate qdap default approach (stemming):
res_qdap_deafault_stem <-
 ↪   validationStatistics(sentiment_scores_qdap_default_stemming$sentiment,
 ↪   original_data_stemming$sentiment, "Positive")
# Validate qdap afinn stemming (stemming);
res_qdap_afinn_stem <-
 ↪   validationStatistics(sentiment_scores_qdap_afinn_stemming$sentiment,
 ↪   original_data_stemming$sentiment, "Positive")
```

4 Resultados

Apresentam-se neste capítulo os resultados obtidos com os vários algoritmos, nas várias experiências realizadas. Foram realizadas diversas análises, que se dividem em: medidas descritivas (frequências e *wordclouds*); resultados da aplicação de cada algoritmo de análise de sentimentos; e aplicação do algoritmo para classificação de sentimento para cada candidato.

4.1 Frequências de "tweets" de cada candidato e de cada categoria

Foram analisados 10729 "tweets" com o objetivo de obter o número de "tweets" para cada candidato, bem como o número de "tweets" para cada tópico, estando estas análises sumarizadas nas figuras 2 e 3.

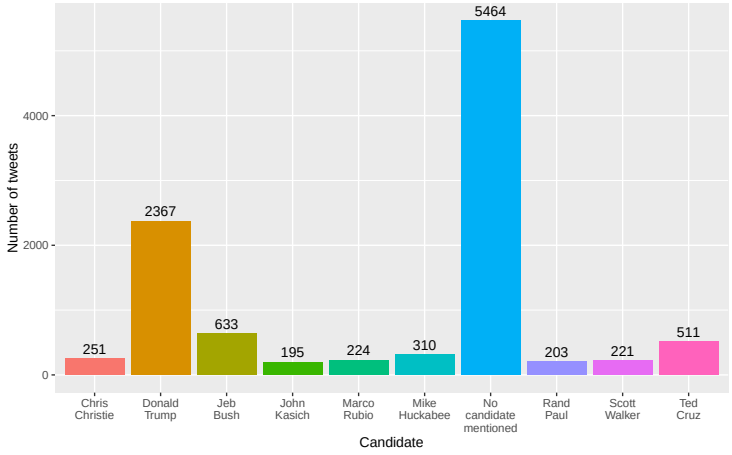

Figura 2. Frequências dos candidatos nos "tweets".

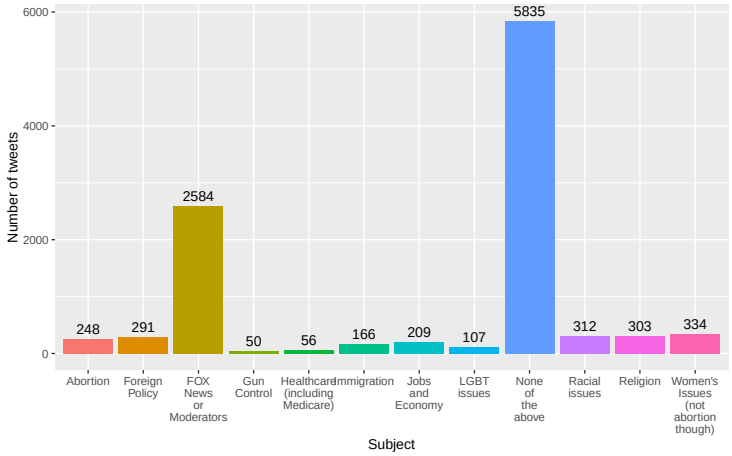

Figura 3. Frequências dos tópicos nos "tweets".

4.2 *Wordclouds* para frequências de palavras

Para perceber que palavras apareceram mais vezes associadas a cada um dos candidatos foram construídas *wordclouds* com base nas frequências (ver fig. 4)

Estas foram elaboradas em R, utilizando o seguinte excerto de código:

```
library(wordcloud)
library(tm)

generate.cloud <- function(candidate_str)
{
  #subset data
  subset_data <- subset(preprocessed_data, candidate == candidate_str)

  # Create corpus
  corpus=Corpus(VectorSource(subset_data$text))

  # Convert to lower-case
  corpus=tm_map(corpus,tolower)

  # Remove stopwords
  corpus=tm_map(corpus,function(x) removeWords(x,stopwords()))

  # convert corpus to a Plain Text Document
  corpus=tm_map(corpus,PlainTextDocument)

  col=brewer.pal(6,"Dark2")
  png(paste(candidate_str,"wordcloud.png"), width=1280,height=800)
  wordcloud(corpus, min.freq=25, scale=c(5,2),rot.per = 0.25,
            random.color=T, max.word=45, random.order=F,colors=col)
}

candidates_names <- unique(preprocessed_data$candidate)
lapply(candidates_names, generate.cloud)
```

4.3 Validação e comparação dos diferentes algoritmos

Para validar as várias abordagens apresentadas no capítulo 3.3.3 foram obtidas diversas *confusion matrixes*, bem como as medidas de *precision*, *recall* e *F1*.

```
library(caret)

# ######################################
# ######## PREPARE VALIDATION ##########
# ######################################

# Read results saved before:
sentiment_scores_manual <- read.csv("csv_data/sentiment_manual.csv")
sentiment_scores_manual_stemming <- read.csv("csv_data/sentiment_manual_stemming.csv")
sentiment_scores_qdap_default <- read.csv("csv_data/sentiment_qdap_default.csv")
sentiment_scores_qdap_afinn <- read.csv("csv_data/sentiment_qdap_afinn.csv")
sentiment_scores_qdap_default_stemming <-
 ↪   read.csv("csv_data/sentiment_qdap_default_stemming.csv")
sentiment_scores_qdap_afinn_stemming <-
 ↪   read.csv("csv_data/sentiment_qdap_afinn_stemming.csv")

# Convert the numeric sentiment values to classes:
convertSentimentFactor <- function(data)
{
  data$sentiment[data$score >= 0] <- "Positive"
  data$sentiment[data$score < 0] <- "Negative"
  data$sentiment <- as.factor(data$sentiment)
```

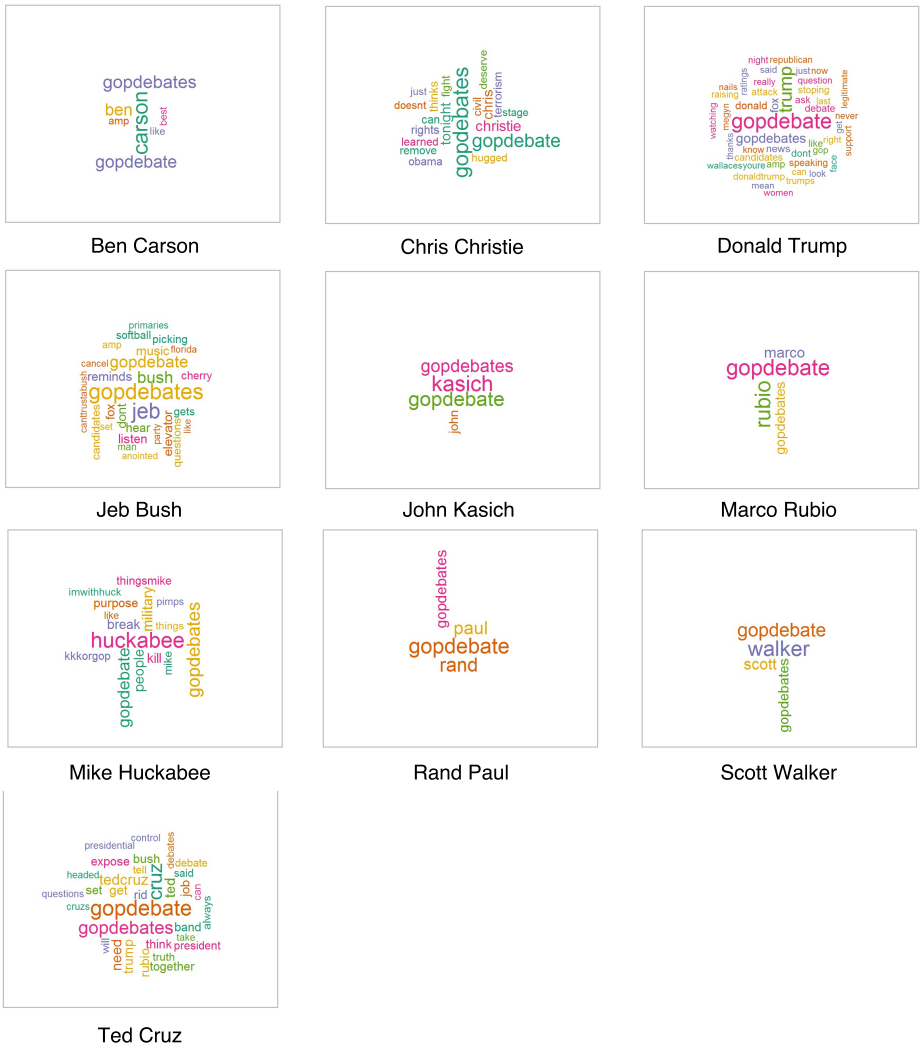

Figura 4. *Wordclouds* relativas às palavras mais frequentes nos "tweets" de cada candidato.

```
  return(data)
}

sentiment_scores_manual <- convertSentimentFactor(sentiment_scores_manual)
sentiment_scores_manual_stemming <-
 ↪   convertSentimentFactor(sentiment_scores_manual_stemming)
sentiment_scores_qdap_default <- convertSentimentFactor(sentiment_scores_qdap_default)
sentiment_scores_qdap_afinn <- convertSentimentFactor(sentiment_scores_qdap_afinn)
sentiment_scores_qdap_default_stemming <-
 ↪   convertSentimentFactor(sentiment_scores_qdap_default_stemming)
sentiment_scores_qdap_afinn_stemming <-
 ↪   convertSentimentFactor(sentiment_scores_qdap_afinn_stemming)
```

```r
########################################
# ######## Evaluation Metrics ############
########################################

# Produces several validation statistics:
# y: Factor vector of true values
# predictions: Factor vector of the predicted values
# positiveClass: Name, as a string, of the positive class
validationStatistics <- function (predictions, y, positiveClass)
{
  # Calculate precision and recall:
  result <- confusionMatrix(predictions, y, positiveClass)
  print(result)
  precision <- result$byClass[['Pos Pred Value']]
  recall <- result$byClass[['Sensitivity']]

  # Calculate F1:
  f1 <- 2 * precision * recall / (precision + recall)

  return(list(precision = precision, recall = recall, f1 = f1))
}

################################
# 1 - Precision, Recall, F1:
################################
# Validate manual approach:
res_man <- validationStatistics(sentiment_scores_manual$sentiment,
 ↪   original_data$sentiment, "Positive")
# Validate manual approach (stemming):
res_man_stem <- validationStatistics(sentiment_scores_manual_stemming$sentiment,
 ↪   original_data_stemming$sentiment, "Positive")
# Validate qdap default approach:
res_qdap_deafault <- validationStatistics(sentiment_scores_qdap_default$sentiment,
 ↪   original_data$sentiment, "Positive")
# Validate qdap afinn approach:
res_qdap_afinn <- validationStatistics(sentiment_scores_qdap_afinn$sentiment,
 ↪   original_data$sentiment, "Positive")
# Validate qdap default approach (stemming):
res_qdap_deafault_stem <-
 ↪   validationStatistics(sentiment_scores_qdap_default_stemming$sentiment,
 ↪   original_data_stemming$sentiment, "Positive")
# Validate qdap afinn stemming (stemming);
res_qdap_afinn_stem <-
 ↪   validationStatistics(sentiment_scores_qdap_afinn_stemming$sentiment,
 ↪   original_data_stemming$sentiment, "Positive")
```

Apresentam-se seguidamente as várias tabelas obtidas:

Method	Accuracy	Precision	Recall	F1
Manual	0.432	0.260	0.934	0.407
Manual with Stemming	0.399	0.249	0.936	0.394
QDAP Default	0.458	0.269	0.930	0.417
QDAP with AFINN	0.474	0.265	0.860	0.406
QDAP Default with Stemming	0.458	0.269	0.930	0.417
QDAP with AFINN and Stemming	0.474	0.265	0.860	0.406

Tabela 1. Comparação das métricas "Precision", "Recall" e "F1" para as diferentes abordagens.

		Reference	
		Negative	**Positive**
Prediction	*Manual*		
	Negative	2548	148
	Positive	5945	2088
	Manual with Stemming		
	Negative	2187	142
	Positive	6306	2094
	QDAP Default		
	Negative	2828	156
	Positive	5665	2080
	QDAP with AFINN		
	Negative	3164	312
	Positive	5329	1924
	QDAP Default with Stemming		
	Negative	2828	156
	Positive	5665	2080
	QDAP with AFINN and Stemming		
	Negative	3164	312
	Positive	5329	1924

Tabela 2. Comparação das *confusion matrixes* para cada um dos diferentes métodos.

Estes resultados permitem verificar que existe uma maior taxa de acerto no sentimento de "tweets" positivos. Este padrão verifica-se independentemente do algoritmo utilizado. Para além disso, ainda no que diz respeito aos "tweets" positivos, a estratégia que tem maior taxa de acerto é o algoritmo manual, recorrendo ao dicionário Hu & Liu, depois de se ter realizado *stemming* sobre as palavras do texto. Por sua vez, a estratégia com piores resultados nos "tweets" positivos, é utilizar a função *polarity* do *package* "QDAP", com o dicionário AFINN, independentemente de ter *stemming*.

Por outro lado, no que diz respeito aos "tweets" com sentimento negativo, os resultados alcançados são opostos. Ou seja, o algoritmo com pior acerto é o manual com *stemming*, enquanto que o algoritmo com melhor taxa de acerto é a função *polarity* do *package* "QDAP", com o dicionário AFINN, independentemente de ter *stemming*. De entre estes dois algoritmos considerou-se que o melhor para analisar o sentimento é o *polarity* do *package* "QDAP", com o dicionário AFINN, uma vez que apresenta maior valor de *accuracy* global (0.47, por oposição a 0.43 no outro algoritmo).

4.4 Análise de sentimento dos candidatos

Depois de se encontrar um modelo de classificação de sentimento, este foi utilizado para inferir o sentimento associado a cada um dos candidatos às eleições

nos Estados Unidos. Para fazer esta análise considerou-se apenas a abordagem que recorre à função *polarity* do *package* "QDAP", com o dicionário AFINN, já apresentada nos capítulos anteriores. Para calcular o sentimento para cada candidato, foi utilizado o seguinte *script* de R:

```
include(ggplot2)

#merge our predicted result with the original pre-processed data
final_dataframe <- cbind(original_data, sentiment_scores_qdap_afinn$sentiment)
names(final_dataframe)[9]<-paste("classification")

#calculate the sentiment frequencies for each candidate
frequencies <- table(final_dataframe$candidate, final_dataframe$classification)

#calculate the relative frequencies for each candidate
frequencies <- frequencies/(frequencies[,1] + frequencies[,2])

#manipulate the result to use with ggplot
freq_dataframe <- as.data.frame(frequencies)
freq_dataframe <- freq_dataframe[-c(8, 19),]
names(freq_dataframe)[1]<-paste("Candidate")
names(freq_dataframe)[2]<-paste("Sentiment")
names(freq_dataframe)[3]<-paste("Freq")

ggplot(data=freq_dataframe, aes(x=Candidate, y=Freq, fill=Sentiment)) +
  geom_bar(stat="identity")
```

Obtiveram-se os resultados apresentados na fig. 5.

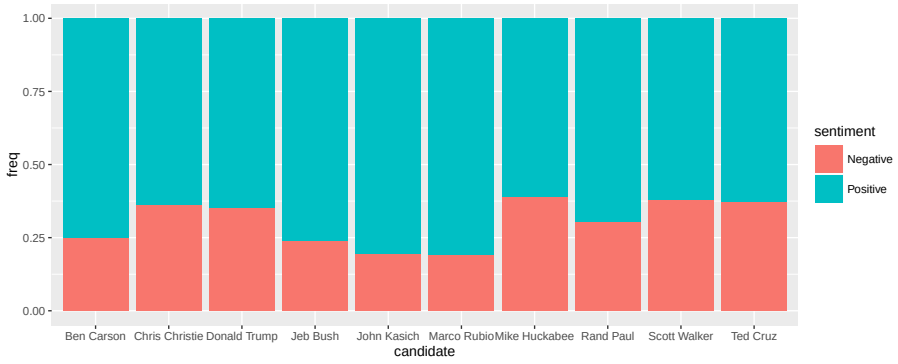

Figura 5. Sentimento associado a cada candidato, em frequências relativas sobre o total de "tweets" para cada candidato.

Estes resultados mostram que o sentimento geral para com todos os candidatos é positivo, contudo, os candidatos "Chris Christ", "Donald Trump", "Mike Huckabe", "Scott Walker" e "Ted Cruz" são os que apresentam sentimento mais negativo, no contexto do debate GOP.

5 Discussão de Resultados e Conclusões

Neste relatório foram apresentadas várias abordagens e foram realizadas diversas experiências, com o objetivo de construir um mecanismo de análise de sentimento em "tweets".

A primeira abordagem foi a mais simples e consistia apenas em confrontar as palavras dos "tweets" com dicionários de sentimento positivo ou negativo. Apesar de simples, esta abordagem com *stemming* foi a que conseguiu melhores resultados na identificação de sentimento positivo em tweets.

Por sua vez, para melhorar os resultados ao nível da classificação de "tweets" com sentimento negativo, utilizou-se a função *polarity* do *package* "QDAP", com o dicionário AFINN. De facto, esta nova abordagem permitiu melhorar os resultados a nível da identificação de sentimento negativo.

Neste sentido, para conjugar os melhores resultados da classificação de "tweets" positivos, com os melhores resultados de classificação de "tweets" negativos, em futuros trabalhos poderia implementar-se uma abordagem alternativa, que tentasse conjugar o melhor dos dois algoritmos.

Desde logo, é ainda de salientar que, ao se utilizar a função *polarity*, é indiferente aplicar-se *stemming*. Isto acontece porque a função *polarity* já inclui no seu processamento interno este procedimento, pelo que realizar o *stemming* previamente se torna redundante. É portanto de salientar que, nas análises realizadas, de sentimento com dicionários, o processo de *stemming* permitiu melhores resultados.

No que toca ainda aos algoritmos utilizados, é ainda de notar que a função *polarity* com o dicionário AFINN é a que permite encontrar melhores resultados, uma vez que é o algoritmo com maior taxa de acerto.

Para além disso, um resultado que deve ser analisado é o porquê da maior taxa de acerto no caso dos "tweets" positivos. Uma hipótese para este resultado é a forma como é feita a classificação de "tweets" em positivos e negativos. Tal como foi referido previamente, a revisão de literatura permitiu concluir que a presença de "tweets" neutros dificulta o processo de classificação. Neste sentido, estes "tweets" foram removidos do *dataset*. Contudo, mesmo assim, os algoritmos utilizados podem retornar o valor "0" para sentimento. Perante este resultado, dado que se assume que não existem "tweets" neutros, é necessário classificar este valor. Foi então decidido que ao valor nulo é atribuído o sentimento positivo. Portanto, à partida, esta abordagem terá mais tendência a acertar em positivos e a falhar negativos, explicando a disparidade encontrada na taxa de acerto dos dois tipos de sentimento.

Por fim, e em jeito de conclusão, apresentámos neste relatório a abordagem seguida pelo grupo na análise de sentimento em "tweets", que aparenta cumprir com os objetivos da disciplina de ECD II, uma vez que foram realizadas várias experiências e realizado um procedimento que permite análise de sentimento em texto. Ao mesmo tempo, esta abordagem é apenas introdutória, uma vez que nos últimos anos a análise de sentimento tem sido uma área em crescimento, com algoritmos cada vez mais completos. Neste sentido, poderiam ser feitas muitas outras experiências, desde utilização de dicionários tendo em conta o tópico do

texto, bem como dicionários que evoluem no tempo, entre outras análises. Para além disso, também seria possível fazer-se extração de *features* do texto e tirar partido de algoritmos de aprendizagem supervisionados.

Referências

1. Vishal Gupta e Gurpreet S Lehal. «A survey of text mining techniques and applications». Em: *Journal of emerging technologies in web intelligence* 1.1 (2009), pp. 60–76.
2. Bo Pang e Lillian Lee. «Opinion mining and sentiment analysis». Em: *Foundations and trends in information retrieval* 2.1-2 (2008), pp. 1–135.
3. Mahmoud Parsian. *Data Algorithms: Recipes for Scaling Up with Hadoop and Spark*. "O'Reilly Media, Inc.", 2015.
4. Kaggle. *First GOP Debate Twitter sentiment*. 2016. URL: `https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment` (acedido em 01/04/2016).
5. Moshe Koppel e Jonathan Schler. «The importance of neutral examples for learning sentiment». Em: *Computational Intelligence* 22.2 (2006), pp. 100–109.
6. Liu e Hu. *Opinion Mining, Sentiment Analysis, and Opinion Spam Detection*. 2004. URL: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html` (acedido em 01/04/2016).
7. Tyler W. Rinker. *qdap: Quantitative Discourse Analysis Package*. 2.2.4. University at Buffalo/SUNY. Buffalo, New York, 2013. URL: `http://github.com/trinker/qdap`.
8. F. Å. Nielsen. *AFINN*. Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, mar. de 2011. URL: `http://www2.imm.dtu.dk/pubdb/p.php?6010`.