SENTIMENT ANALYSIS WITH HARRY POTTER SERIES

TABLE OF CONTENTS

Goal

The goal of this project is to build a sentiment analysis model which will allow us to categorize words in the Harry Potter series based on their sentiments, that is whether they are positive, negative or neutral using the lexicons from the tidytext package and get insights like to know what are the most frequent words used in the series through word count and word clouds which determine important characters, to know top positive-negative words with the help of bing lexicon, to determine which book has the highest negative or positive emotional score, to compare the three lexicons to see how the sentiment over the series changes with the changing lexicon and to analyze the text further more using bigrams.

Abstract

The project is mainly based on Sentiment Analysis within R using tidytext package that comprises of sentiment lexicons like AFINN, BING and NRC that are present in the dataset of “sentiments”. The text data that is used in this project was provided by the Harry Potter R package (harrypotter) on GitHub that contains the text for all seven books in the Harry Potter series, by JK Rowling. The visualization used in this project were created using R programming on R Studio.

The list of seven novels that are used in this analysis are as below:

Harry Potter and the Philosophers Stone (1997)

Harry Potter and the Chamber of Secrets (1998)

Harry Potter and the Prisoner of Azkaban (1999)

Harry Potter and the Goblet of Fire (2000)

Harry Potter and the Order of the Phoenix (2003)

Harry Potter and the Half Blood Prince (2005)

Harry Potter and the Deathly Hallows (2007)

Basic Analysis with the Text

Loading Required Packages

The required text data was loaded into RStudio from the harrypotter package along with the other packages that are required for this analysis like tidytext, textdata, worldcloud2, ggplot2, RColorBrewer, dplyr, rshape2, tidyverse, tidyr.

Creating Tidy Text

The raw text *figure(a)* needed to be shaped properly as, to perform sentiment analysis the data should be in the form of tidy format therefore; conversion of all the seven Harry Potter novels into a data frame or tibble that has each word by chapter by book using unnest_token function has been done *figure(b)*. Using unnest_token function is very important as it splits the entire text into single words, removes punctuation and converts the entire text to lower case.

```
> raw = harrypotter::chamber_of_secrets
> raw
 [1] "THE WORST BIRTHDAY    Not for the first time, an argument had broken out over breakfast a
t number four, Privet Drive. Mr. Vernon Dursley had been woken in the early hours of the morni
ng by a loud, hooting noise from his nephew Harry's room.    \"Third time this week!\" he roare
d across the table. \"If you can't control that owl, it'll have to go!\"    Harry tried, yet ag
ain, to explain.    \"She's bored,\" he said. \"She's used to flying around outside. If I could
 just let her out at night -\"    \"Do I look stupid?\" snarled Uncle Vernon, a bit of fried eg
g dangling from his bushy mustache. \"I know what'll happen if that owl's let out.\"    He exch
anged dark looks with his wife, Petunia.    Harry tried to argue back but his words were drowne
d by a long, loud belch from the Dursleys' son, Dudley.    1    \"I want more bacon.\"    \"Ther
e's more in the frying pan, sweetums,\" said Aunt Petunia, turning misty eyes on her massive s
on. \"We must build you up while we've got the chance .... I don't like the sound of that scho
ol food ......\"    \"Nonsense, Petunia, I never went hungry when I was at Smeltings,\" said Un
cle Vernon heartily. \"Dudley gets enough, don't you, son?\"    Dudley, who was so large his bo
ttom drooped over either side of the kitchen chair, grinned and turned to Harry.    \"Pass the
 frying pan.\"    \"You've forgotten the magic word,\" said Harry irritably.    The effect of th
is simple sentence on the rest of the family was incredible: Dudley gasped and fell off his ch
air with a crash that shook the whole kitchen; Mrs. Dursley gave a small scream and clapped he
r hands to her mouth; Mr. Dursley jumped to his feet, veins throbbing in his temples.    \"I me
ant `please'!\" said Harry quickly. \"I didn't mean -\"    \"WHAT HAVE I TOLD YOU,\" thundered
```

Figure(a)

	book	chapter	word
1	Philosopher's Stone	1	the
2	Philosopher's Stone	1	boy
3	Philosopher's Stone	1	who
4	Philosopher's Stone	1	lived
5	Philosopher's Stone	1	mr
6	Philosopher's Stone	1	and
7	Philosopher's Stone	1	mrs
8	Philosopher's Stone	1	dursley
9	Philosopher's Stone	1	of
10	Philosopher's Stone	1	number
11	Philosopher's Stone	1	four
12	Philosopher's Stone	1	privet

Figure(b)

Word Frequency Analysis

Now that I had a tidy tibble, I used count function to know what are the words that occur the most in the entire series and from the below figure one can observe that the most common words that appear are the "stop words" see the *figure(c)*. Stop words are words that do not have weight or meaning in the analysis like 'the', 'an', 'and', 'a' and many other, so I removed the entire stop words from the text using the "anti_join" function and the resulted word frequency like in the below *figure(d)* made more sense.

	word	n
	<chr>	<int>
1	the	51593
2	and	27430
3	to	26985
4	of	21802
5	a	20966
6	he	20322
7	harry	16557
8	was	15631
9	said	14398
10	his	14264

... with 24,465 more rows

Figure(c)

A tibble: 23,795 x 2

	word	n
	<chr>	<int>
1	harry	16557
2	ron	5750
3	hermione	4912
4	dumbledore	2873
5	looked	2344
6	professor	2006
7	hagrid	1732
8	time	1713
9	wand	1639
10	eyes	1604

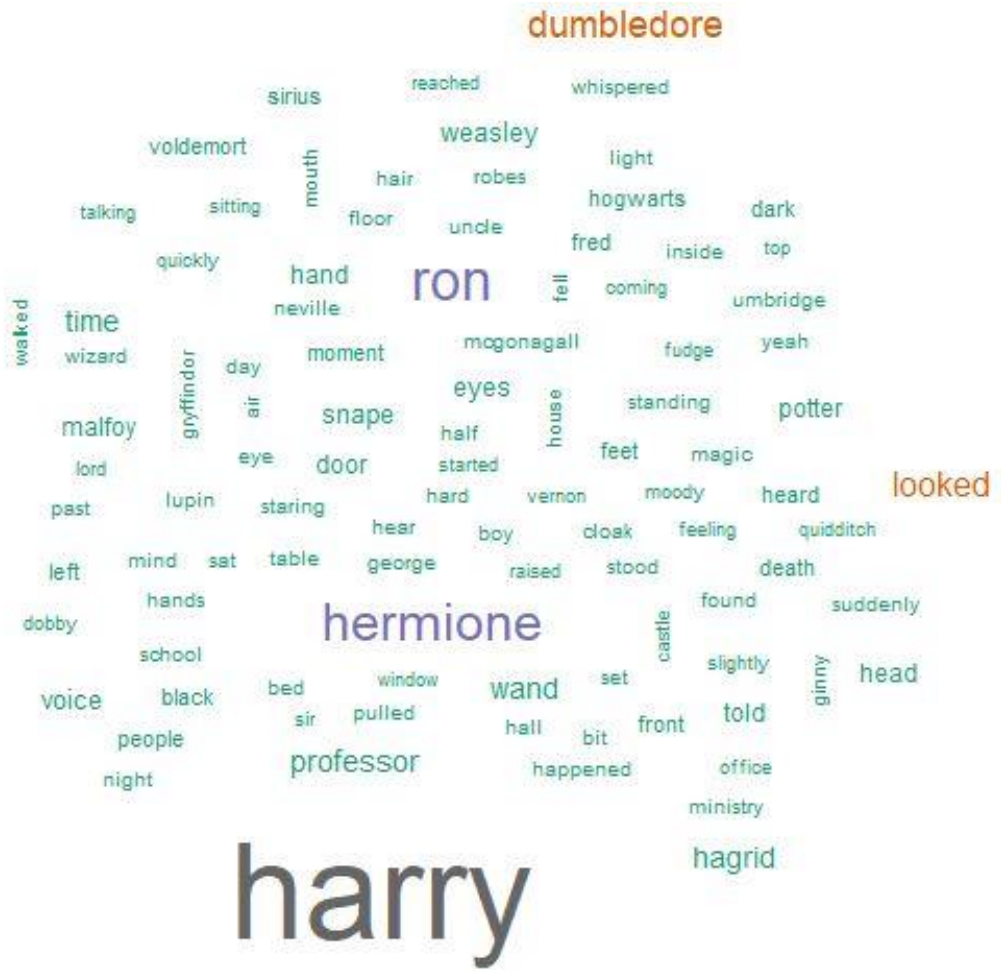
... with 23,785 more rows

Figure(d)

Word Cloud

Plotted a word cloud to see the top 100 words from the entire text which can be seen in the *figure(e).* In word cloud the tokens that appear frequently in the series appears to be in bigger size than the other words. From the below *figure(e)* one can understand "harry", "ron",

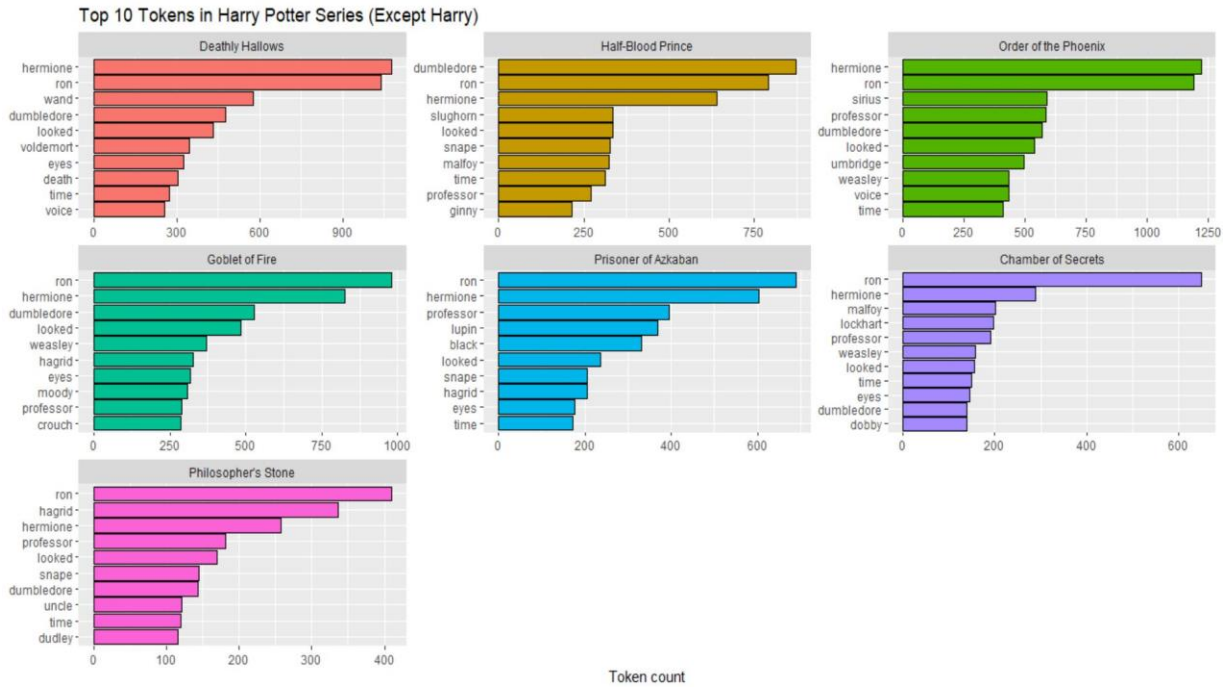
“hermione” and “dumbeldore” are the important characters in the harry potter series as they tend to appear in bigger size when compared with other tokens.



Figure(e)

Most Frequent Words used in the Series

To know in more detail, I plotted the bar chart using “ggplot2” package which shows the most frequent words that occur in individual books except the word “Harry” and result is the below *figure(f).* One can observe that Ron and Hermione are the most frequent words that are used in most of the books after Harry but, Dumbledore overcome them in the “Half-Blood Prince” which depicts that his character is more important after Harry in this book but it’s surprising that he doesn’t appear in the book Prisoner of Azkaban. The next frequent words are “wand”, “professor”, “hagrid”, “snape” and others and it makes sense as we all know how important those characters are in the novel.



Figure(f)

Sentiment Analysis using tidytext

Sentiment Analysis is the interpretation and classification of emotions within text data using text analysis techniques

Applications include:

- Social Media Monitoring – Analyzing tweets and Facebook posts for over a period to know the emotions and sentiment of the audience
- Brand Monitoring – Brand Monitoring is important to care to see not only whether people are talking about the brand, but how they are talking about it, are they positive or negative of the brand
- Customer Reviews - By automatically running sentiment analysis on incoming surveys, companies can detect customers who are strongly negative towards the product or service, and can respond to them right away

Performed sentiment analysis within R using tidytext package that comprises of sentiment lexicons like AFINN, bing and nrc that are present in the dataset of sentiments. All three of these lexicons are based on unigrams, i.e., single words. Let me go give you a brief description about each lexicon:

NRC lexicon:

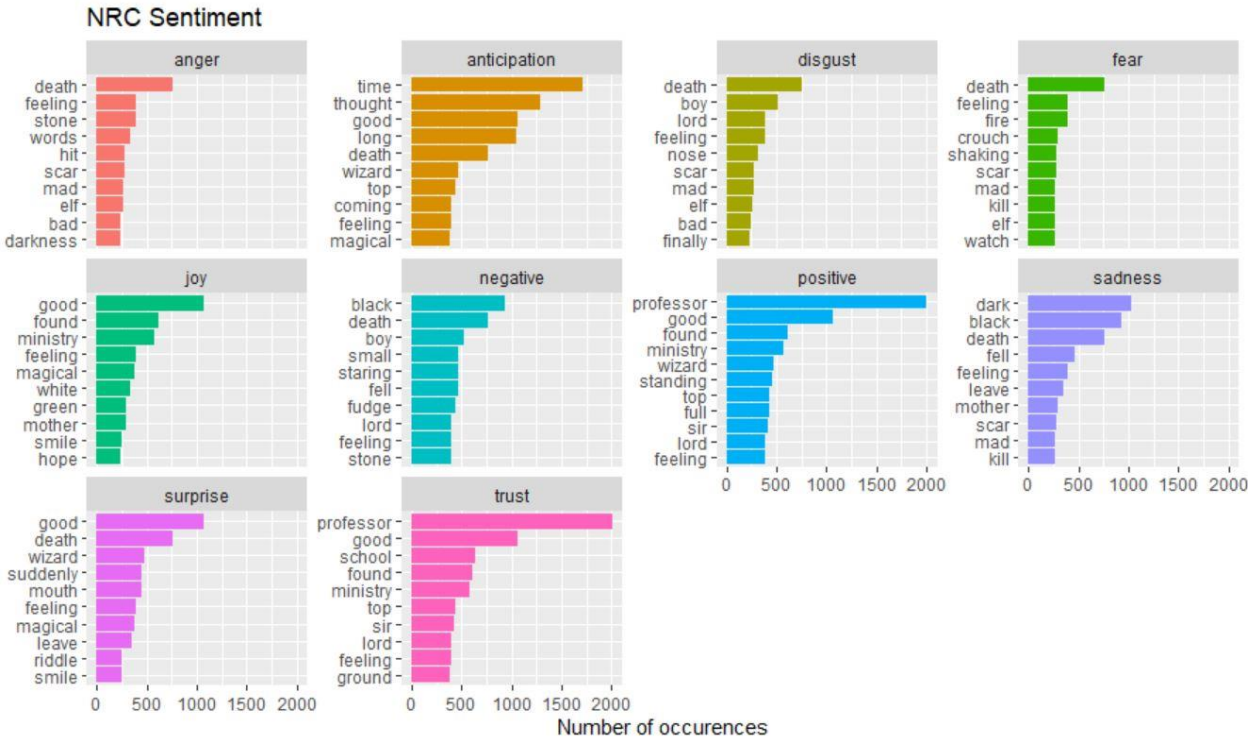
Nrc emotion lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). It gives a brief view of how many emotional words are used in the entire text along with the sentiments i.e. Positive and Negative.

When applied to the text it separates each token with respect to their sentiment and from below *figure(g)* one can observe that it categorizes each token to one or more sentiments like how token “proud” is categorized to four different sentiments.

```
# A tibble: 264,705 x 4
   book                chapter word   sentiment
   <fct>                 <int> <chr>  <chr>
 1 Philosopher's Stone       1 boy    disgust
 2 Philosopher's Stone       1 boy    negative
 3 Philosopher's Stone       1 proud  anticipation
 4 Philosopher's Stone       1 proud  joy
 5 Philosopher's Stone       1 proud  positive
 6 Philosopher's Stone       1 proud  trust
 7 Philosopher's Stone       1 expect anticipation
 8 Philosopher's Stone       1 expect positive
 9 Philosopher's Stone       1 expect surprise
10 Philosopher's Stone       1 expect trust
# ... with 264,695 more rows
```

Figure(g)

The below graph in *figure(h)* shows the count of each word in their respective emotions.



Figure(h)

Bing lexicon:

The bing lexicon categorizes tokens in the text in a binary fashion into positive and negative categories. When applied to the text it separates each token with respect to their sentiment and from below *figure(i)*

```
# A tibble: 65,094 x 4
   book                chapter word       sentiment
   <fct>                 <int> <chr>      <chr>
 1 Philosopher's Stone       1 proud      positive
 2 Philosopher's Stone       1 perfectly  positive
 3 Philosopher's Stone       1 thank      positive
 4 Philosopher's Stone       1 strange    negative
 5 Philosopher's Stone       1 mysterious negative
 6 Philosopher's Stone       1 nonsense   negative
 7 Philosopher's Stone       1 useful     positive
 8 Philosopher's Stone       1 finer      positive
 9 Philosopher's Stone       1 greatest   positive
10 Philosopher's Stone       1 fear       negative
# ... with 65,084 more rows
```

Figure(i)

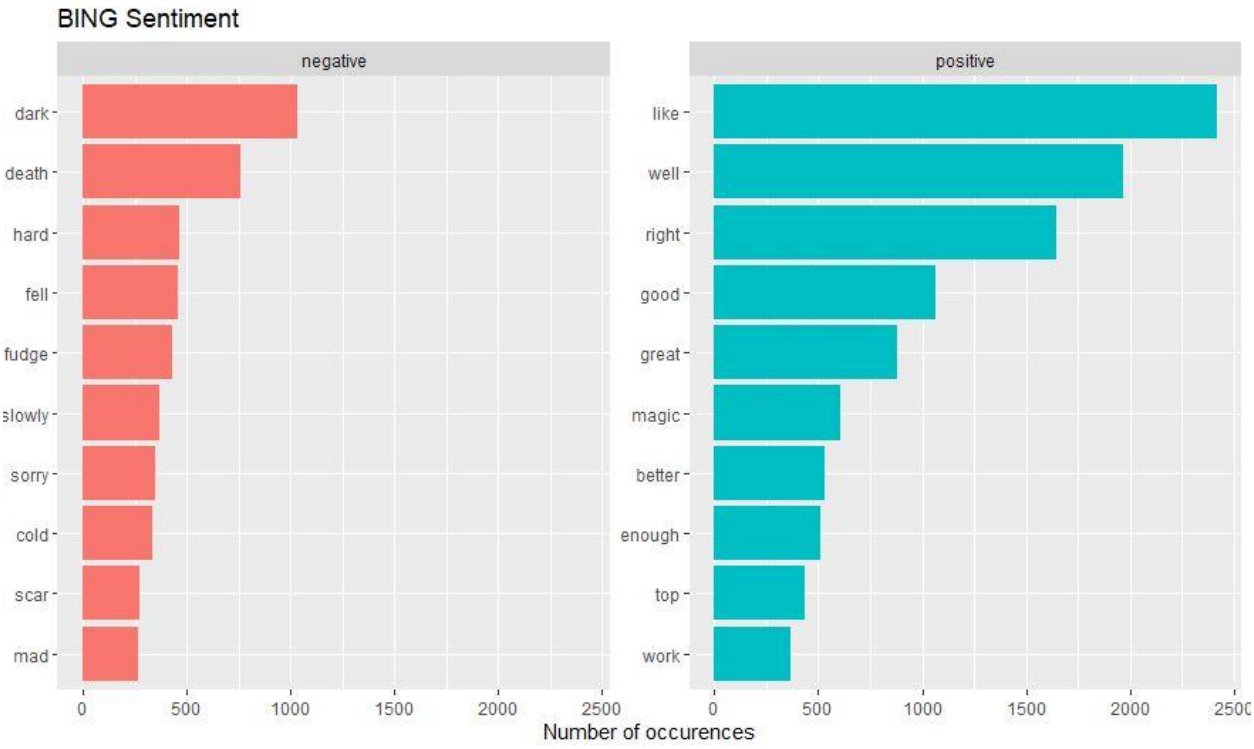
To know the total number of positive and negative tokens: Here, in *figure(j)* I observed that the series is dominated by the negative sentiment than having positive emotion.

```
# A tibble: 2 x 2
  sentiment     n
  <chr>     <int>
1 negative  38227
2 positive  28686
```

Figure(j)

For better understanding of the bing lexicon a bar graph has been plotted *figure(k)* that displays how each word in the entire text is categorized into either “Positive” or “Negative” using bing lexicon. According to the series, tokens like “dark”, “death”, “cold” and many others are

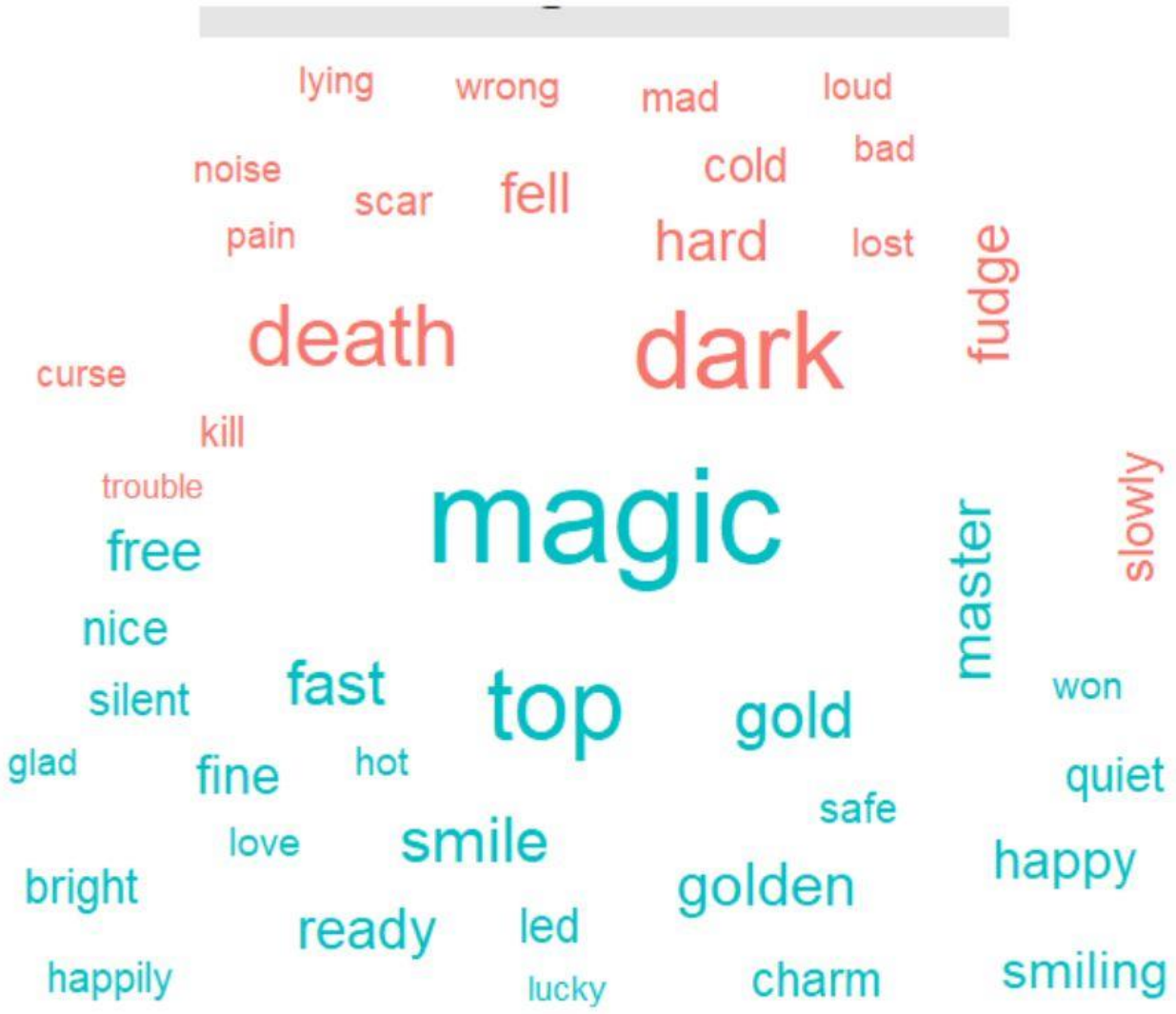
categorized as negative tokens which as words themselves used in negative context whereas tokens like “magic”, “like”, “good” and many others are categorized as positive lexicon i.e. words used in positive context.



Figure(k)

Bing Comparison Cloud

Bing emotion lexicon is used to make a comparison cloud that displays the 50 most frequently occurring words in the series that were categorized by ‘bing’ and color-codes them based on negative or positive sentiment. One can observe from *figure(l)* that words like “Harry”, “Hermione” and “Ron” don’t appear in this cloud, because character names are not classified as positive or negative in ‘bing’.


lying
wrong
mad
loud
noise
scar
fell
cold
bad
pain
hard
lost
fudge
death
dark
curse
kill
trouble
magic
free
master
slowly
nice
silent
fast
top
gold
won
glad
fine
hot
quiet
safe
love
smile
happy
bright
golden
ready
led
happily
lucky
charm
smiling

figure(l)

AFINN Lexicon:

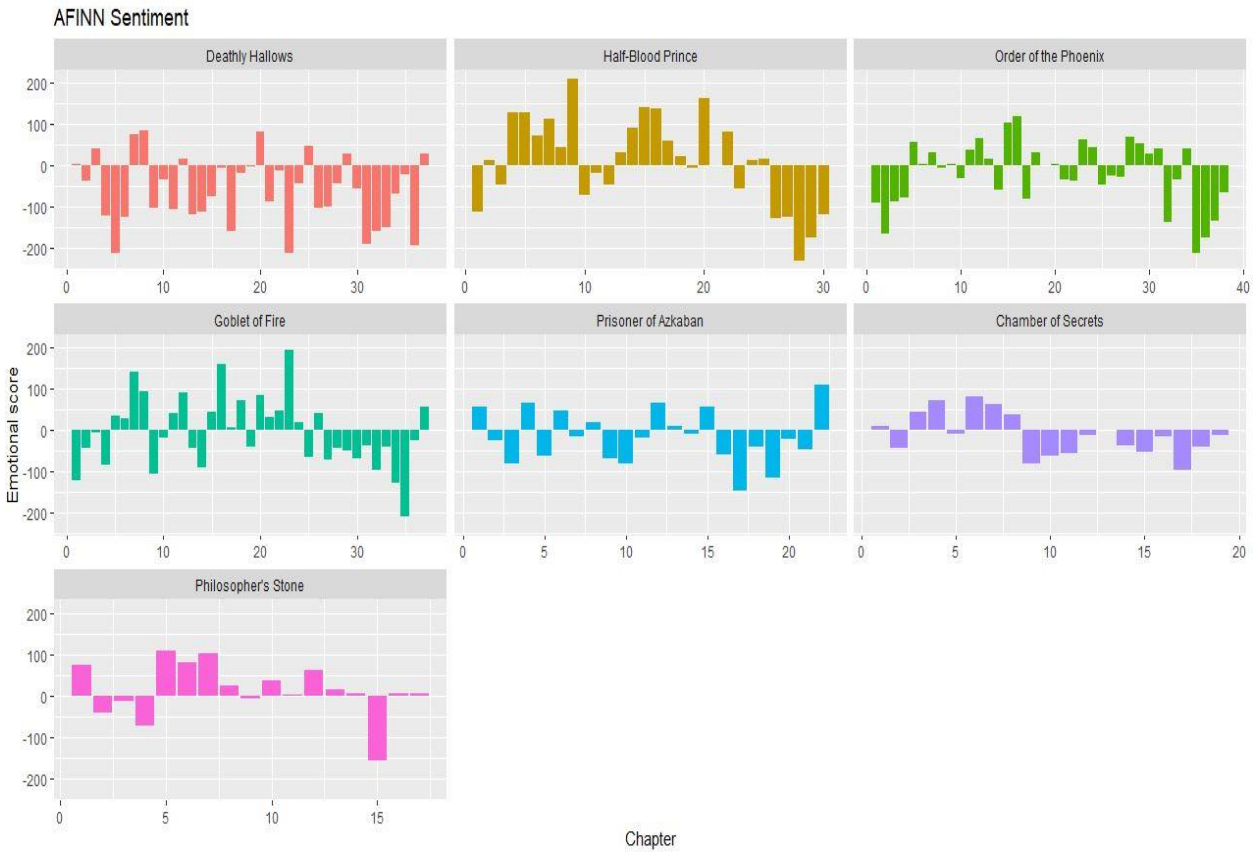
The AFINN lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

After performing the AFINN lexicon on the entire series it resulted in the following way: *figure(m).* Here, one can observe that sentimental words like “proud”, “perfectly”, “thank”, “fear”, “strange” and many others have give their respective scores keeping positive and negative as their extremes.

```
# A tibble: 56,311 x 4
   book                chapter word      value
   <fct>                 <int> <chr>     <dbl>
 1 Philosopher's Stone       1 proud         2
 2 Philosopher's Stone       1 perfectly     3
 3 Philosopher's Stone       1 thank         2
 4 Philosopher's Stone       1 strange      -1
 5 Philosopher's Stone       1 nonsense     -2
 6 Philosopher's Stone       1 big           1
 7 Philosopher's Stone       1 useful        2
 8 Philosopher's Stone       1 no           -1
 9 Philosopher's Stone       1 greatest      3
10 Philosopher's Stone       1 fear         -2
# ... with 56,301 more rows
```

Figure(m)

To get more insight about how AFINN sentiment effects the harry potter series I plotted to the see the emotonal score of each chapters in their respective books like the *figure(n)* below.

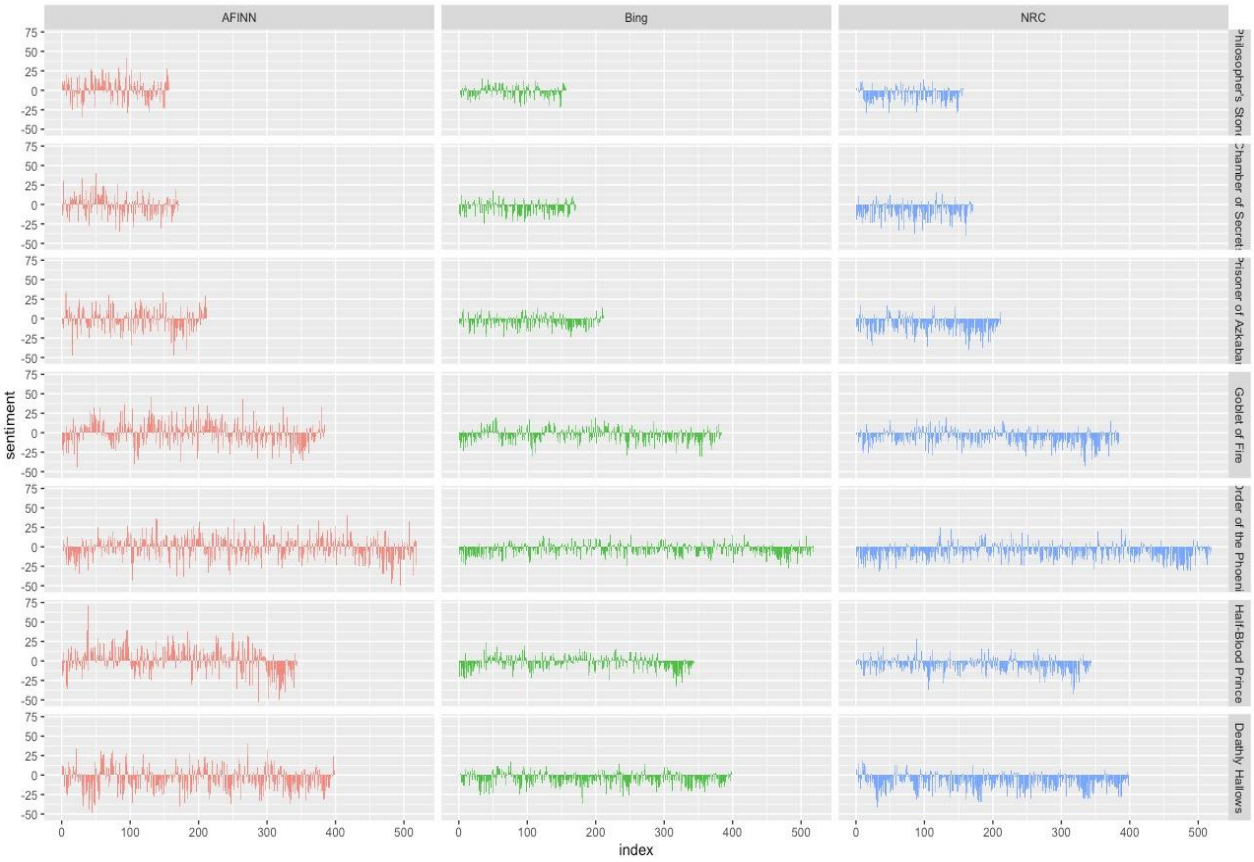


Figure(n)

From this one can observe that the book “Harry Potter and the Deathly Hallows” have more words that tend to be negative whereas the book “Philosophers Stone” has less negative words over all the chapters for except 15th chapter comparitive to other books. When it comes to positive score books like “Half-Blood Prince” has more chapters that involve positive words.

Comparing Three Lexicons

From below *figure(o)* one can observe there are similar dips and peaks in sentiment at about the identical places within the novel, but the absolute values are significantly different. The AFINN lexicon gives the largest absolute values, with high positive and negative values. The lexicon from Bing has lower absolute values for positive and seems have more values towards negative sentiment. The NRC tends to have negative values like AFINN but does not have much of positive values. One can also observe the length of each book where “Order of Phoenix” is the lengthiest book and “Philosophers Stone” being the smallest.



Figure(o)

Analyzing Using Bigrams

To analyze the text in more detail I choose to use bigrams, for example: text “dark magic” when using unigrams gives us two separate words “dark” with negative sentiment and “magic” with positive sentiment but as sentence has more negative context to it. Bigrams are a pair of words that appear consecutively in a text. Bigrams give more scope to understand the sentiment over the text cause sometimes using just single tokens might not give better understanding. Example of bigrams for sentence “harry is my friend” is (harry, is), (is, my), (my, friend).

I repeated the process of shaping the text data from the beginning of the document, but this time I specified that bigrams should be used to tokenize the text rather than single words and resulted output is in the below *figure(p)*

```
A tibble: 1,089,186 x 3
   book                chapter bigram
   <fct>                 <int> <chr>
 1 Philosopher's Stone       1 the boy
 2 Philosopher's Stone       1 boy who
 3 Philosopher's Stone       1 who lived
 4 Philosopher's Stone       1 lived mr
 5 Philosopher's Stone       1 mr and
 6 Philosopher's Stone       1 and mrs
 7 Philosopher's Stone       1 mrs dursley
 8 Philosopher's Stone       1 dursley of
 9 Philosopher's Stone       1 of number
10 Philosopher's Stone       1 number four
… with 1,089,176 more rows
```

Figure(p)

Word Frequency Analysis

After this, used count() function to see the most frequently occurred bigrams and one can observe from the *figure(q)* that most of the bigrams like the single words have stop words. I removed the stop words using the “anti_join” function and the resulted output made more sense, from the *figure(r)* one can observe that the topmost used bigrams are “professor mcgonagall”, “uncle Vernon”, “harry potter” and others. The only bigrams in the top ten that don’t contain character names are “Death Eaters”, “Invisibility Cloak” and “Dark Arts”.

A tibble: 340,021 x 2

	bigram	n
	<chr>	<int>
1	of the	4895
2	in the	3571
3	said harry	2626
4	he was	2490
5	at the	2435
6	to the	2386
7	on the	2359
8	he had	2138
9	it was	2123
10	out of	1911

… with 340,011 more rows

Figure(q)

A tibble: 89,120 x 2

	bigram	n
	<chr>	<int>
1	professor mcgonagall	578
2	uncle vernon	386
3	harry potter	349
4	death eaters	346
5	harry looked	316
6	harry ron	302
7	aunt petunia	206
8	invisibility cloak	192
9	professor trelawney	177
10	dark arts	176

… with 89,110 more rows

Figure(r)

Term Frequency – Inverse Document Frequency

Used above bigrams to observe tf-idf (term frequency -inverse document frequency). Tf-idf is an analysis that is used to identify how common a word appears in a particular text, given how many times it occurs in a group of texts. For example, Professor Lupin had played an important role in “The Prisoner of Azkaban”, but not so much in the other books so for a person who had not read

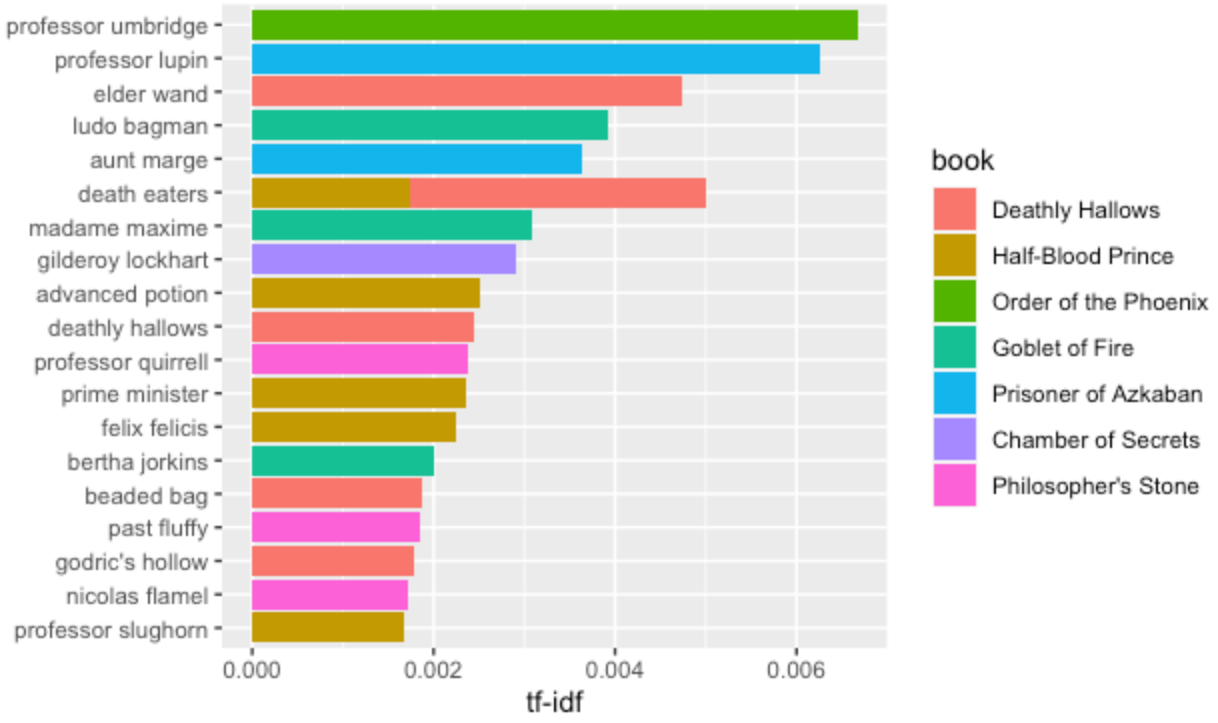
all of the books should be able to determine this by simply counting the number of times the name ”Professor Lupin" occurs in “The Prisoner of Azkaban” and comparing that number to the frequency of that bigram in the rest of the books in the series.

To quantify this concept, the term frequency (the number of times a token appears during a document divided by the overall number of tokens within the document) is multiplied by the inverse document frequency (the total number of documents divided by the amount of documents containing the token). Below *figure(s)* shows the td-idf results

```
# A tibble: 107,016 x 6
   book                bigram                n      tf   idf  tf_idf
   <fct>               <chr>             <int>   <dbl> <dbl>   <dbl>
 1 Order of the Phoenix professor umbridge   173 0.00533 1.25  0.00667
 2 Prisoner of Azkaban  professor lupin      107 0.00738 0.847 0.00625
 3 Deathly Hallows      elder wand            58 0.00243 1.95  0.00473
 4 Goblet of Fire       ludo bagman           49 0.00201 1.95  0.00391
 5 Prisoner of Azkaban  aunt marge           42 0.00290 1.25  0.00363
 6 Deathly Hallows      death eaters         139 0.00582 0.560 0.00326
 7 Goblet of Fire       madame maxime         89 0.00365 0.847 0.00309
 8 Chamber of Secrets   gilderoy lockhart     28 0.00232 1.25  0.00291
 9 Half-Blood Prince    advanced potion       27 0.00129 1.95  0.00252
10 Deathly Hallows      deathly hallows       30 0.00126 1.95  0.00245
# … with 107,006 more rows
```

Figure(s)

I plotted a bar chart using ggplot2 package visualizing the top 20 bigrams which has the highest td-idf scores among the seven books in the series. As one can observe from the *figure(t)* “Professor Umbridge”, from the “The Order of the Phoenix” has the highest tf-idf score.



Figure(t)

Conclusion

- Most frequent words in the series are: Harry, Ron, Hermione and Dumbledore
- We saw how three lexicons show effect on the text by visualizing comparison cloud, emotional score and different categorized emotions using nrc lexicon
- “Deathly Hallows” has the highest negative score in entire series and “Half Blood Prince” has the most positive score
- Since mostly the series is dominated by negative words NRC and Bing lexicons showcased this properly

- Using Bigrams made us analyze in even more detail about the top characters that played a prominent role in the entire series
- "Order of Phoenix" is the lengthiest book keeping "Philosophers Stone" the shortest.

References

https://www.tidytextmining.com/sentiment.html

https://cran.r-project.org/web/packages/wordcloud2/vignettes/wordcloud.html#lettercloud-function

https://uc-r.github.io/tidyr

https://github.com/EmilHvitfeldt/R-text-data#harrypotter

https://monkeylearn.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20is%20the%20interpretation,or%20services%20in%20online%20feedback.