# Analysis of data from common e-commerce website

## Introduction

The project will  analyze reviews from different e-commerce websites such as Amazon. We will make certain assumptions on those data and confirm if those assumptions are true. We will be using an open dataset from Amazon at *https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt* filter data to find out if our assumptions were true.

### Amazon

Amazon is the world's largest ecommerce website with over 12 million products available in the USA currently. It provides those products with a combination of In-house brands such as Amazon Basics, Amazon essentials and Amazon fresh as well as millions of other third party retailers. All these products can be reviewed by customers and rated from one to five stars. Customers can also leave feedback and it also mentions if he is a valid customer or a invalid customer. We are using the open source review data provided by Amazon which contains reviews for 18 million reviews categorized by type or purpose of the product. We have categories such as watches, electronics, books, movies, home improvement amongst others. The list of all categories available in the datasets are as follows:

1. Wireless ,
2. Watches ,
3. Video Games ,
4. Video DVD ,
5. Video ,
6. Toys ,
7. Tools ,
8. Sports ,
9. Software ,
10. Shoes ,
11. Pet Products ,
12. Personal Care Appliances ,
13. PC ,
14. Outdoors ,
15. Office Products ,
16. Musical Instruments ,
17. Music ,

18. Mobile Electronics ,
19. Mobile Apps ,
20. Major Appliances ,
21. Luggage ,
22. Lawn and Garden ,
23. Kitchen ,
24. Jewelry ,
25. Home Improvement ,
26. Home Entertainment ,
27. Home ,
28. Health Personal Care ,
29. Grocery ,
30. Gift Card ,
31. Furniture ,
32. Electronics ,
33. Digital Video Games ,
34. Digital Video Download ,
35. Digital Software ,
36. Digital Music Purchase ,
37. Digital Ebook Purchase  ,
38. Digital Ebook Purchase ,
39. Camera ,
40. Books ,
41. Books ,
42. Beauty ,
43. Baby ,
44. Automotive ,
45. Apparel

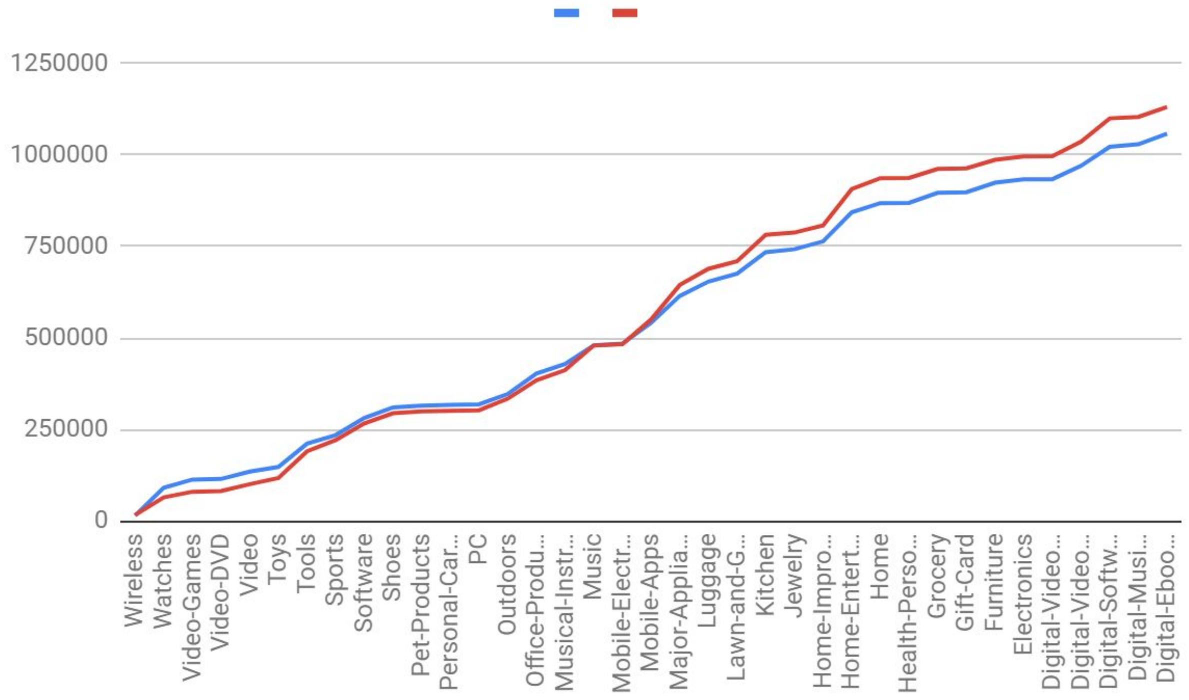
The total collection compressed is about 36 GB is size and uncompressed is about 63 GBs. We are only focusing on products available in the USA and removing reviews earlier than 2005. The data is in form of tsv file with columns containing  the following

1. Marketplace: For our data this is US
2. customer_id : Unique ID assigned to each customer
3. review_id : Unique ID assigned to each review
4. product_id : ASIN id to search product even when it is removed from the store. It keeps history of all products uploaded on the website
5. Product_parent: Unique id assigned to each seller
6. Product_title: Title or name of product
7. Product_category: Category on website product was placed under. We have all data separated in different files based on categories
8. star_rating : Rating given by customer corresponding to the review. The rating is out of five. A product may be rated from one star upto five
9. Helpful_votes: Displays how many people found that review concerning the product useful. Any user can upvote a review and more helpful reviews are placed at the forefront of the review tab.
10. Total_votes: total votes including those that are helpful and not helpful
11. Verified_purchase: Denotes whether the customer is verified or not. That is whether he has purchased the  item or not?
12. Review_headline: Title of the review
13. Review_body: Paragraph of review, limited to 5000 words.
14. Review_date: Date at which review was posted.

We are going to be ignoring reviews posted before 2005, as they are probably irrelevant.

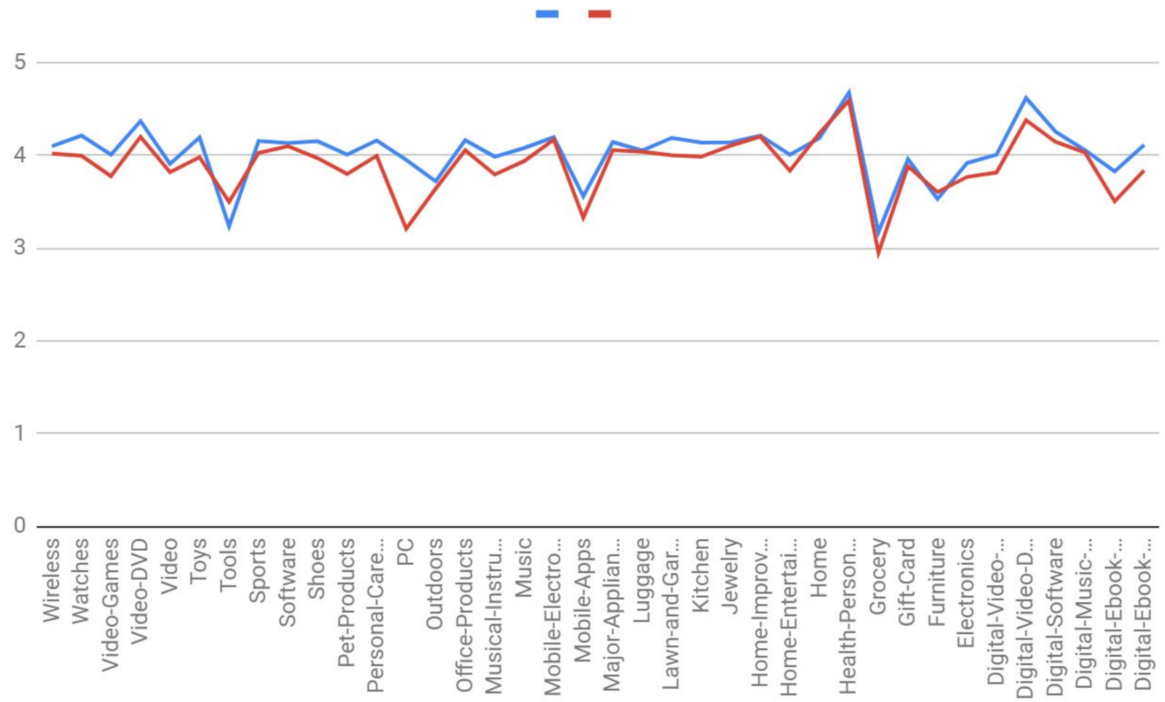
**<u>Observation 1(amazon.java and amazondivert.java)</u>**
**Question:** Who gives on average higher ratings, verified or not verified users?

For this calculation we are looking at the Amazons dataset and calculating the average ratings given by verified and unverified users. We are going to be calculating for each product the average rating given by verified users and average rating given by non verified users. The graph below shows for categories where verified was higher rate and where unverified was higher rated. For example in wireless verified users reviews were higher rated 14127 times as compared to 15579 times where non verified users rated higher.



Red is unverified and blue is verified.

Average  ratings for verified and unverified reviews per category



Red is unverified and blue is verified.

We also run the test on an entire dataset of a single entity and we found out that non verified users rate on average higher for the same products compared to verified users .

And:

Verified higher for:  1058213
non verified higher for :  1131081
average for verified:  4.0322307
Average for non verified:  4.3098882
Non verified customers gives average more output for the same product

Code core logic:

```
 while((line=br1.readLine())!=null){
          ax = line.split("\t");
           if(Integer.parseInt(ax[14].split("-")[0])<2005){continue;}
          if(!validCustomer.containsKey(ax[3])){
              if(ax[11].equalsIgnoreCase("Y")){
                  temp=new Float[4];
                  temp[0] = Float.parseFloat(ax[7]);
                  temp[1] = 0F;
                  temp[2]  =1F;
                  temp[3] = 0F;
                  validCustomer.put(ax[3], temp);
              }
              else{
                   temp=new Float[4];
                  temp[0] = 0F;
                  temp[1] = Float.parseFloat(ax[7]);
                  temp[2]  =0F;
                  temp[3] = 1F;
                  validCustomer.put(ax[3], temp);
              }
          }
          else{
          if(ax[11].equalsIgnoreCase("Y")){
                  temp=validCustomer.get(ax[3]);
                  temp[0] += Float.parseFloat(ax[7]);
                  temp[2] +=1F;
                  validCustomer.replace(ax[3], temp);
              }
              else{
                  temp=validCustomer.get(ax[3]);
                  temp[1] += Float.parseFloat(ax[7]);
                  temp[3] +=1F;
                  validCustomer.replace(ax[3], temp);
              }
          }
        }
        int thy=0;
           float     sumforver=0,sumfornonver=0;
            for (Map.Entry<String, Float[]> entry : validCustomer.entrySet()) {
                temp  =entry.getValue();
//                bw.write(entry.getKey()+"\t"+temp[0]/temp[2]+" "+temp[1]/temp[3]+"\n");
                if(temp[0]==0 || temp[1]==0){continue;}
                   thy++;
                float t1 = temp[0]/temp[2];
                float t2 = temp[1]/temp[3];
                sumforver +=t1;
```

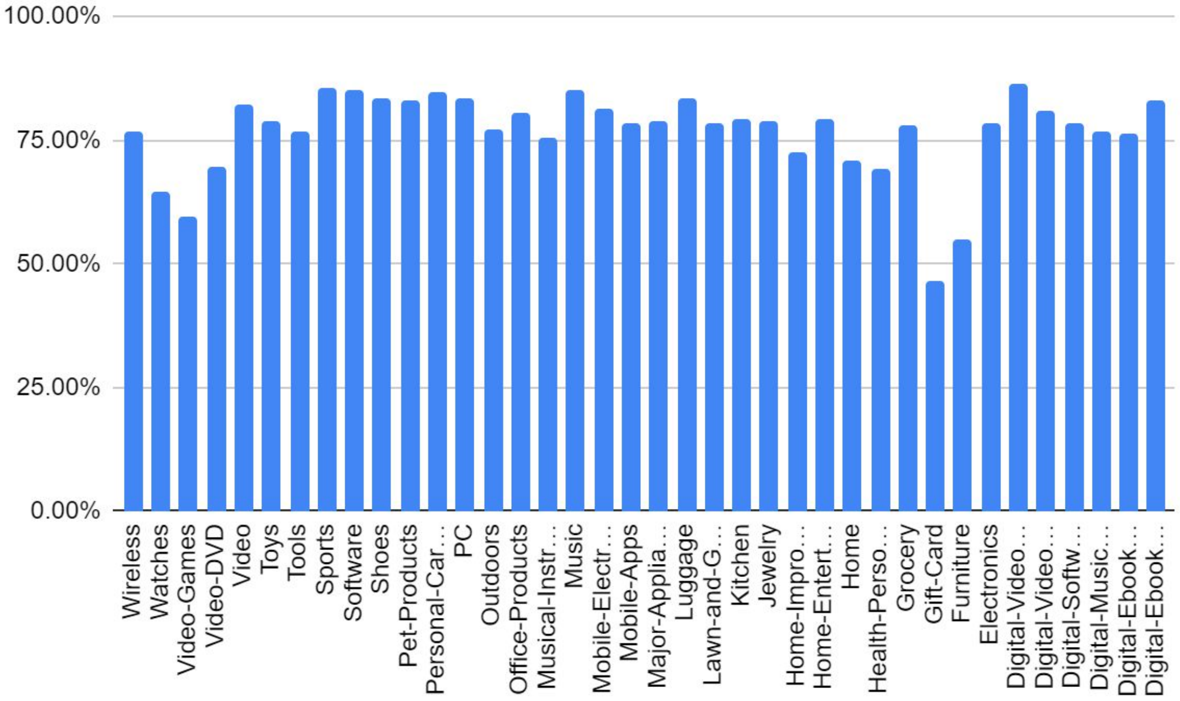
```
             sumfornonver+=t2;
             if(t1>t2){verified++;}
             if(t2>t1){nonVerified++;}
          }
          System.out.println("avg for ver:   "+sumforver/thy+"  avg for non ver:"+sumfornonver/thy+"");
        if(verified>nonVerified){
       System.out.println("verified customers gives average more output for the same product");
       bw.write(file.getName()+"\n"+"verified customers gives average more output for the same
product"+"\n");
       bw.write("verified:  "+verified+"  non verified:  "+nonVerified+"\n");
        }
       else{
       System.out.println("verified customers gives average more output for the same product");
       bw.write(file.getName()+"\n"+"non verified customers gives average more output for the same
product"+"\n");
       bw.write("verified:  "+verified+"  non verified:  "+nonVerified+"\n");
        }
       bw.flush();
```

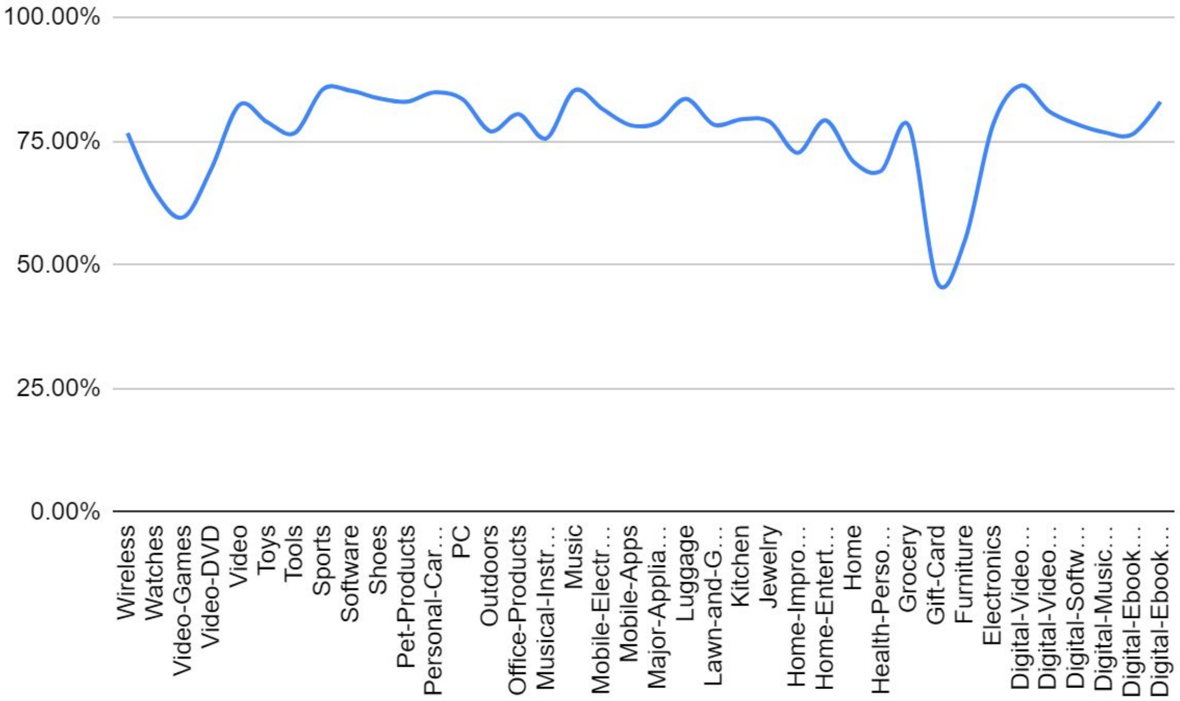
**<u>Observation 2 (amazon1.java and amazon1divert.java)</u>**
**<u>Question:</u>**

What percentage of votes are helpful?

For this calculation we are looking at the Amazons dataset and calculating the percentage of votes that are helpful. This helps us understand the trend for a particular category. So, if a higher percentage of votes are helpful for a particular category of product we can say that the customers are happier regarding that section's review of the product. We can also understand the customer's mindset when purchasing products from a certain category. For example in the home category highest number of votes are helpful @85.34%, which means reviews in home section of website are more relevant. We are representing our findings in the form of a bar chart and a line chart. The dip in chart represents where the reviews are less relevant and and a hill represent higher relevance.

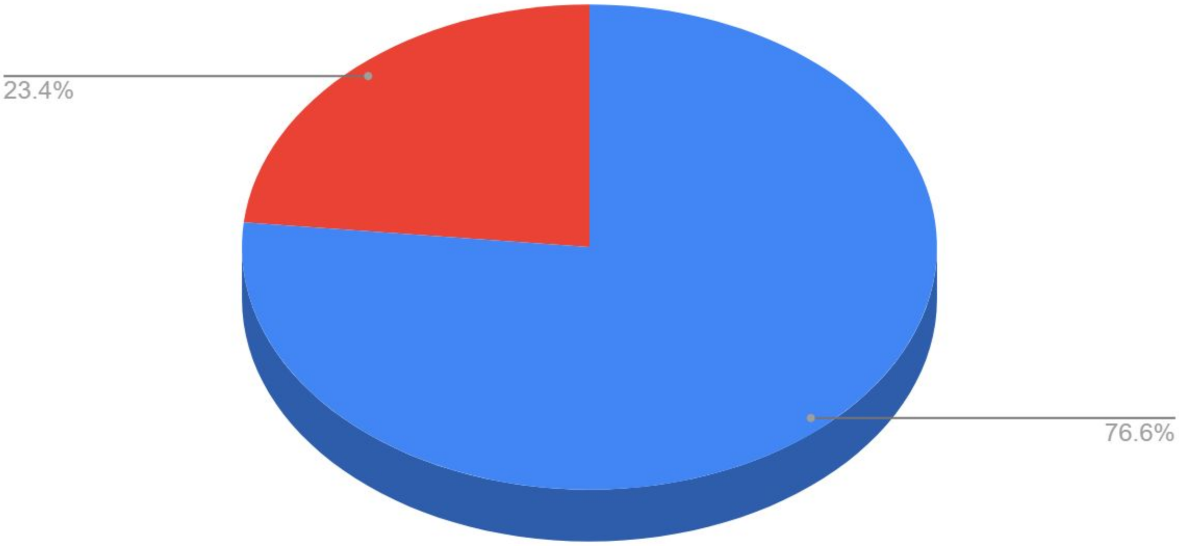
Charts representing % of votes that rated review helpful




100.00%
75.00%
50.00%
25.00%
0.00%
Wireless
Watches
Video-Games
Video-DVD
Video
Toys
Tools
Sports
Software
Shoes
Pet-Products
Personal-Car…
PC
Outdoors
Office-Products
Musical-Instr…
Music
Mobile-Electr…
Mobile-Apps
Major-Applia…
Luggage
Lawn-and-G…
Kitchen
Jewelry
Home-Impro…
Home-Entert…
Home
Health-Perso…
Grocery
Gift-Card
Furniture
Electronics
Digital-Video…
Digital-Video…
Digital-Softw…
Digital-Music…
Digital-Ebook…
Digital-Ebook…

We also ran the process on the entire dataset at once and found that 76.6% of votes were helpful. A higher percentage means that the review section is more relevant and a lower section means lower relevance.

```
helpful:  162637460
Total: 212284009
 percentage helpful:  76.613144%
```



**Code core logic**:

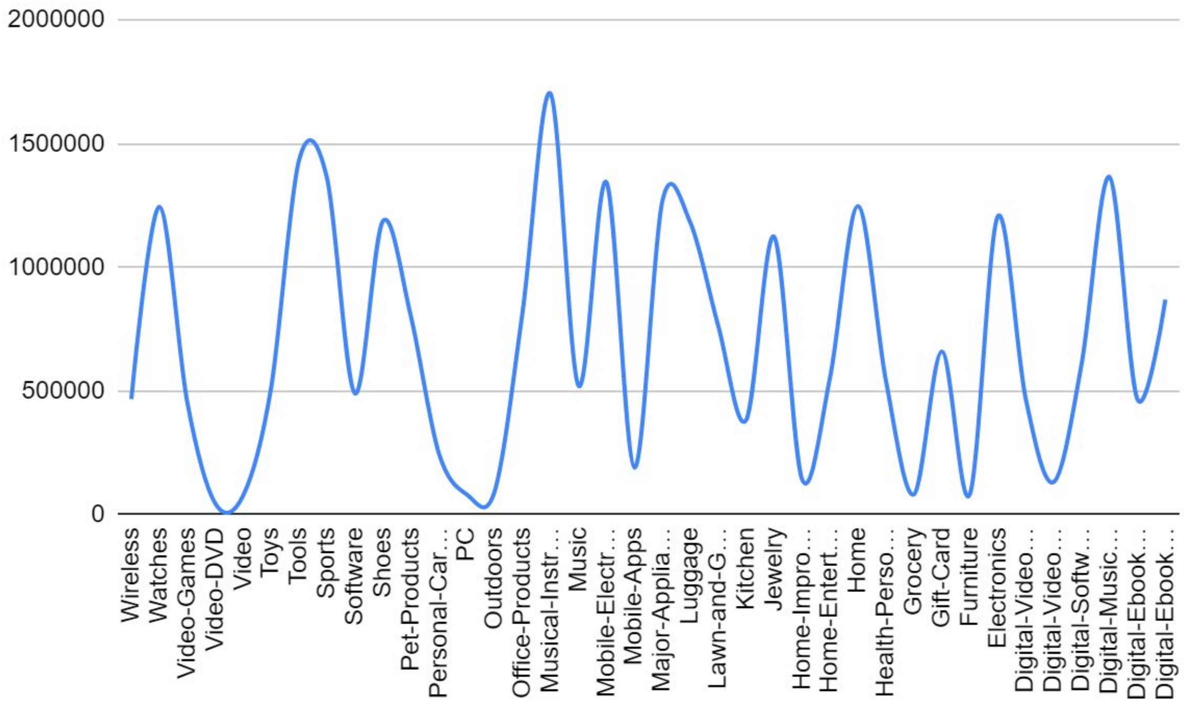
```
For each file in Directory{
  while((line=br1.readLine())!=null){
           ax = line.split("\t");
           if(Integer.parseInt(ax[14].split("-")[0])<2005){continue;}
            bw.write(ax[14]+"\n");
            helpful += Integer.parseInt(ax[8]);
            totals += Integer.parseInt(ax[9]);
         }
         de = (float) (((double)helpful/totals)*100);
         System.out.println("helpful:  "+ helpful+"\nTotal: "+ totals+"\n percentage helpful:  "+de+"%")
         bw.flush();
         bw.close();
}
```

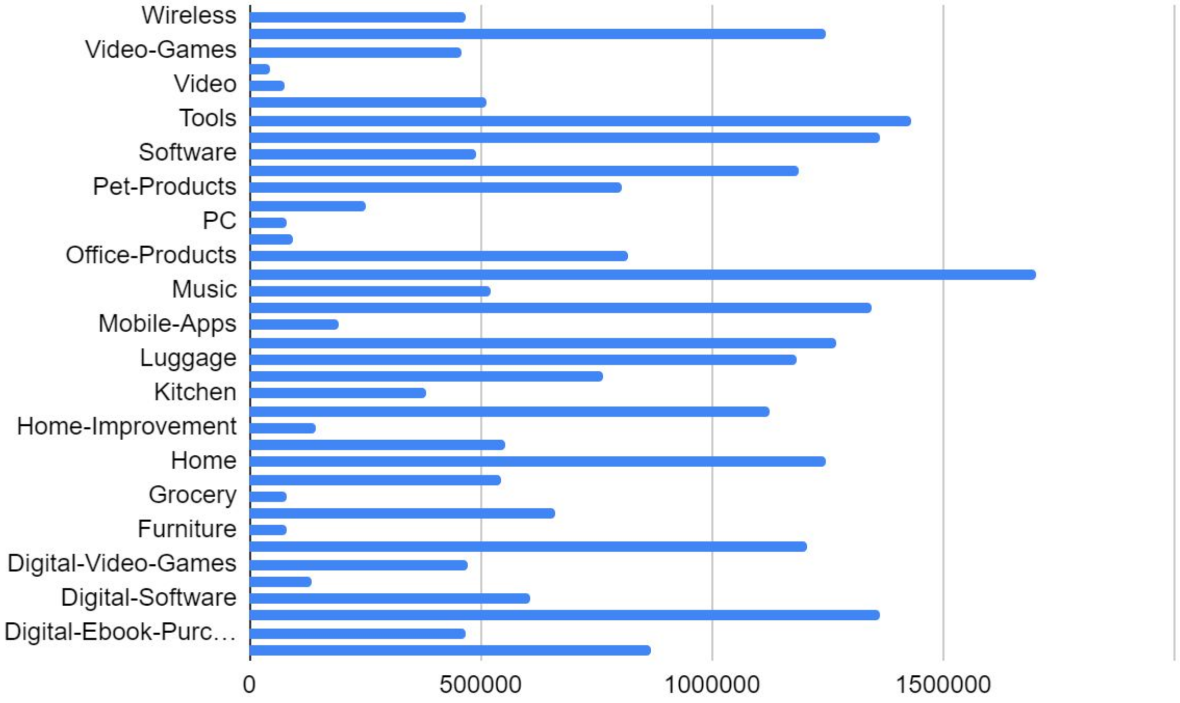
**Observation 3 (amazon2.java and amazon2divert.java)**

**Question:** Are ratings from more frequent customers more helpful?

For this calculation we are looking at the Amazons dataset and finding whether the ratings from more frequent customers is more helpful. We are doing this by first finding the average votes that rate a review high and if average reviews per customer.

If the customer has more reviews than average and his reviews are rated more helpful then we increase the counter for measurement by one. If the customer has more reviews than the average votes but reviews are rated less helpful we reduce the counter by one and vice versa if average reviews for customers are less than average. We find that reviews by more regular customers are always rated more helpful which is very obvious. We can plot the counter to see how helpful the review really is and compare it by category.





We also ran the code on the entire dataset as one and found that the generalized trend that the reviews from more frequent customers were more helpful.

## Code core logic:

```
For file in directory{
 while((line=br1.readLine())!=null){
           ax = line.split("\t");
           if(Integer.parseInt(ax[14].split("-")[0])<2005){continue;}
          if(!customers.containsKey(ax[1])){
              temp = new Integer[3];
              temp[0]=1;
              temp[1]=Integer.parseInt(ax[8]);
              temp[2] = Integer.parseInt(ax[9]);
              customers.put(ax[1], temp);
          }
          else{
            temp = customers.get(ax[1]);
              temp[0]+=1;
              temp[1]+=Integer.parseInt(ax[8]);
              temp[2] += Integer.parseInt(ax[9]);
              customers.replace(ax[1], temp);
          }

            bw.write(ax[14]+"\n");
          helpful += Integer.parseInt(ax[8]);
          totals += Integer.parseInt(ax[9]);
        }
        float de = (float) (((double)helpful/totals)*100);
    //        System.out.println(helpful/totals);
       int sums=0;
        for (Map.Entry<String, Integer[]> entry : customers.entrySet()) {
            sums+=entry.getValue()[0];
        }
        float ty = (float) ((double)sums/customers.size());
        int counter=0;
        float percForind = 0F;
        for (Map.Entry<String, Integer[]> entry : customers.entrySet()) {
            temp = entry.getValue();
            percForind = (float) ((double)temp[1]/temp[2]);
            if(temp[0]>ty){
              if(percForind>=de){
                counter+=1;
              }
              else{
                counter-=1;
              }
            }
            else{
            if(percForind>=de){
                counter-=1;
              }
              else{
                counter+=1;
              }
            }
        }

        System.out.println(counter);
        if(counter>=0){
             System.out.println("yes rating from more frequent customer is more helpful");
        }
        else{
            System.out.println("No rating from more frequent customer is not more helpful");
        }
        bw.flush();
        bw.close();
      }
```

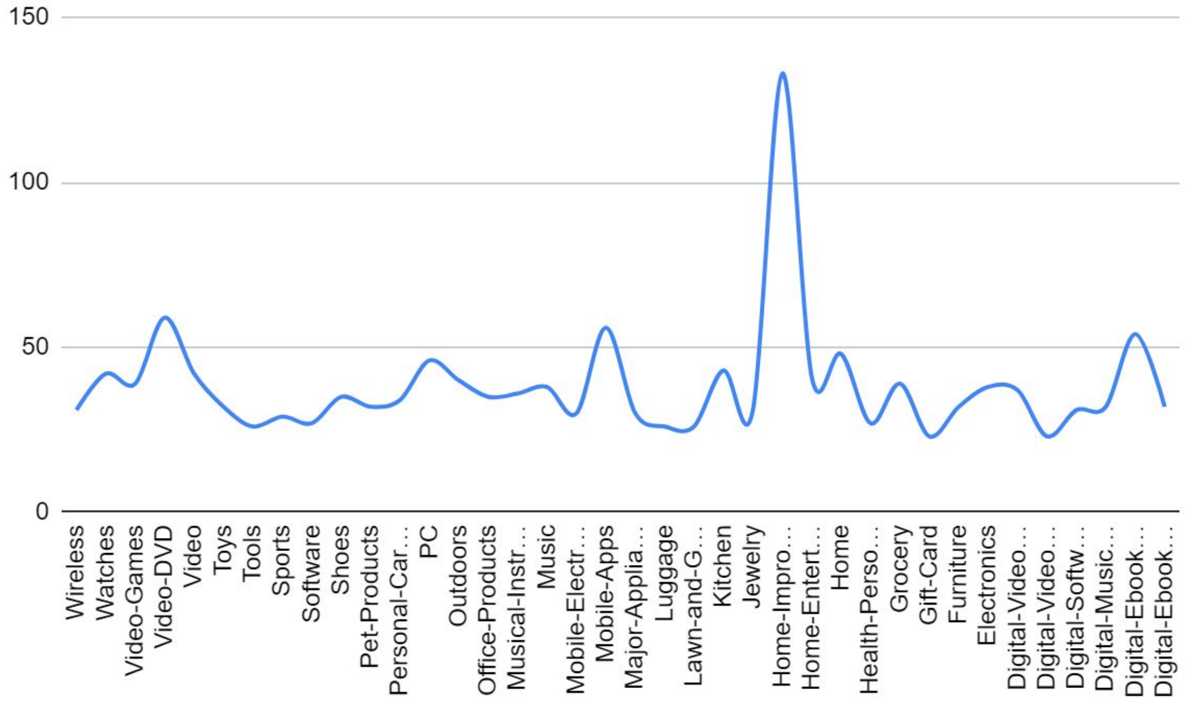
**<u>Observation 4 (amazon3.java and amazon3divert.java)</u>**

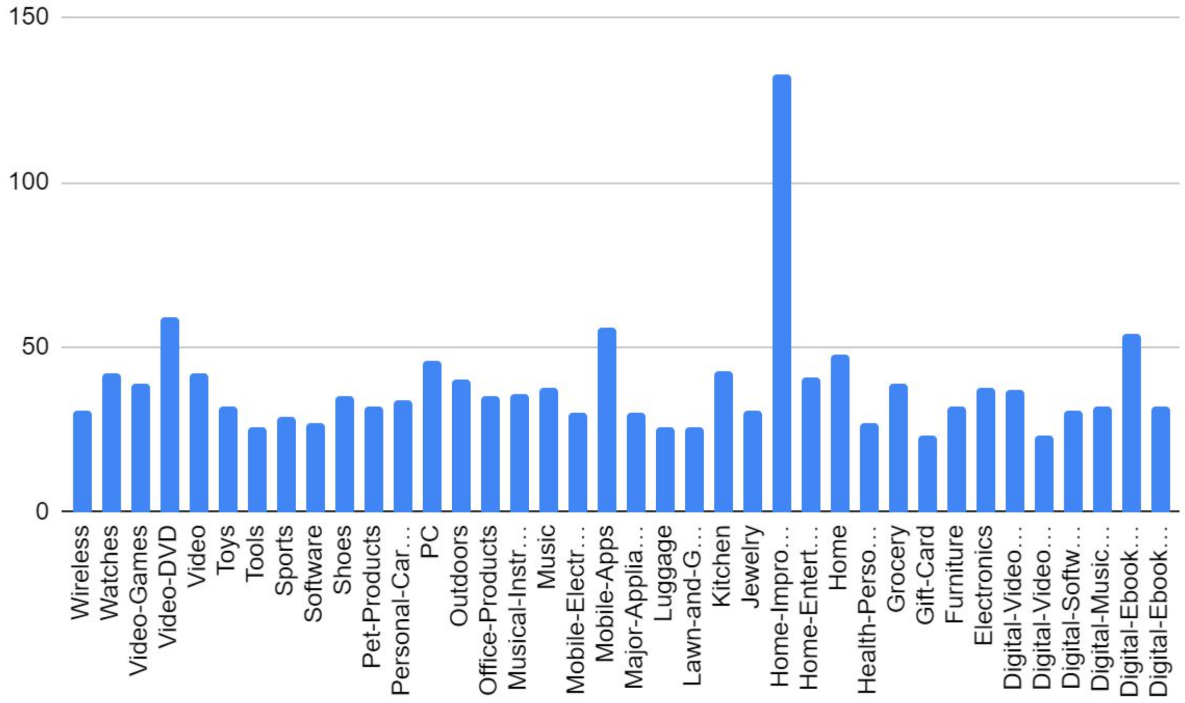
**<u>Question:</u>** Are longer reviews more helpful also average review size?

For this calculation we are looking at the Amazons dataset and finding whether the ratings for larger reviews is more helpful. We are doing this by first finding the average votes that rate a review high and median size of review. If review is larger than the median and it is rated more helpful we increase counter by one and If review is smaller than the median and it is rated more helpful we decrease counter by one.

We find the general trend that longer reviews are more helpful as they have more parts that people can agree on. We also find the average review size for each category and find out that reviews for products in the more expensive section are longer. Home improvement which contains things such as home theatre and furniture(relatively expensive than other product) are longer

Average review size:





We also ran the code on the entire dataset as one and found that the generalized trend that the longer the reviews, more customers were finding it  helpful.

## Code core logic:

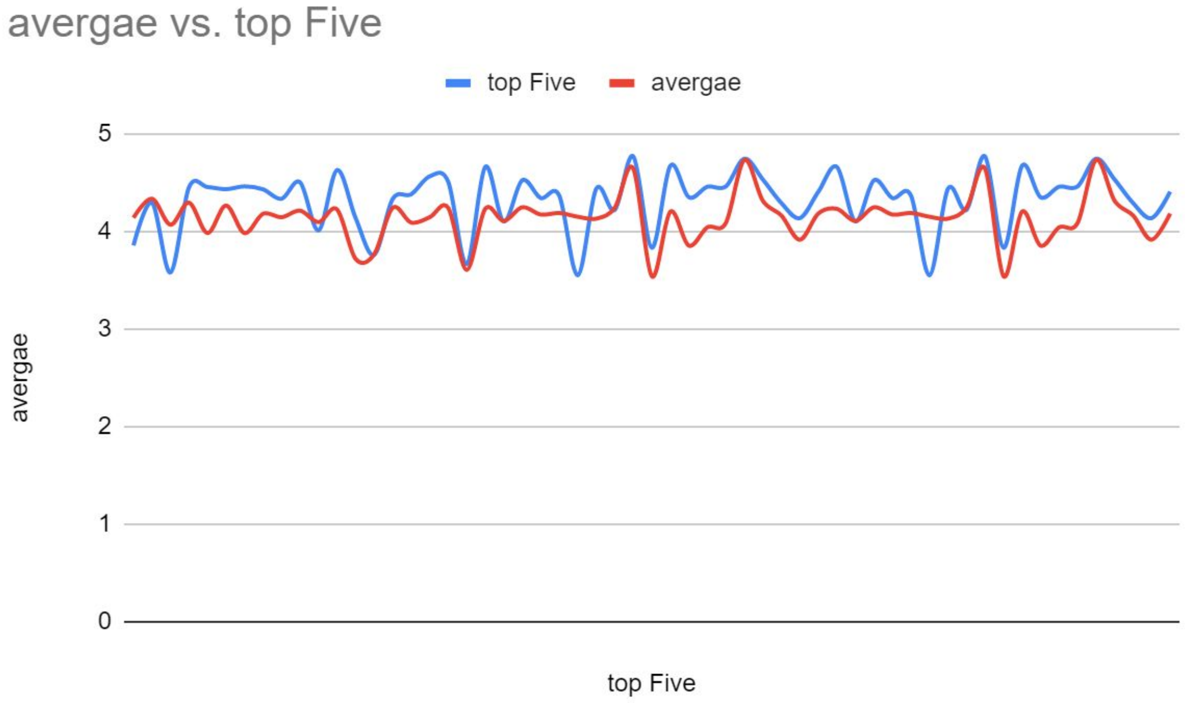
```
For file in Directory{
  while((line=br1.readLine())!=null){
        ax = line.split("\t");

        if(Integer.parseInt(ax[14].split("-")[0])<2005){continue;}
         bw.write(ax[14]+"\n");
         int y12= ax[13].split("[ .,\"]").length;

         al.add(y12);
         temps = new Double[2];
         temps[0] = (double)y12;
         if(Integer.parseInt(ax[8])==0 ||Integer.parseInt(ax[9])==0){continue;}
         temps[1] = ((double)Integer.parseInt(ax[8])/Integer.parseInt(ax[9]))*100;

         reviewUniq.put(ax[2], temps);
         helpful += Integer.parseInt(ax[8]);
```

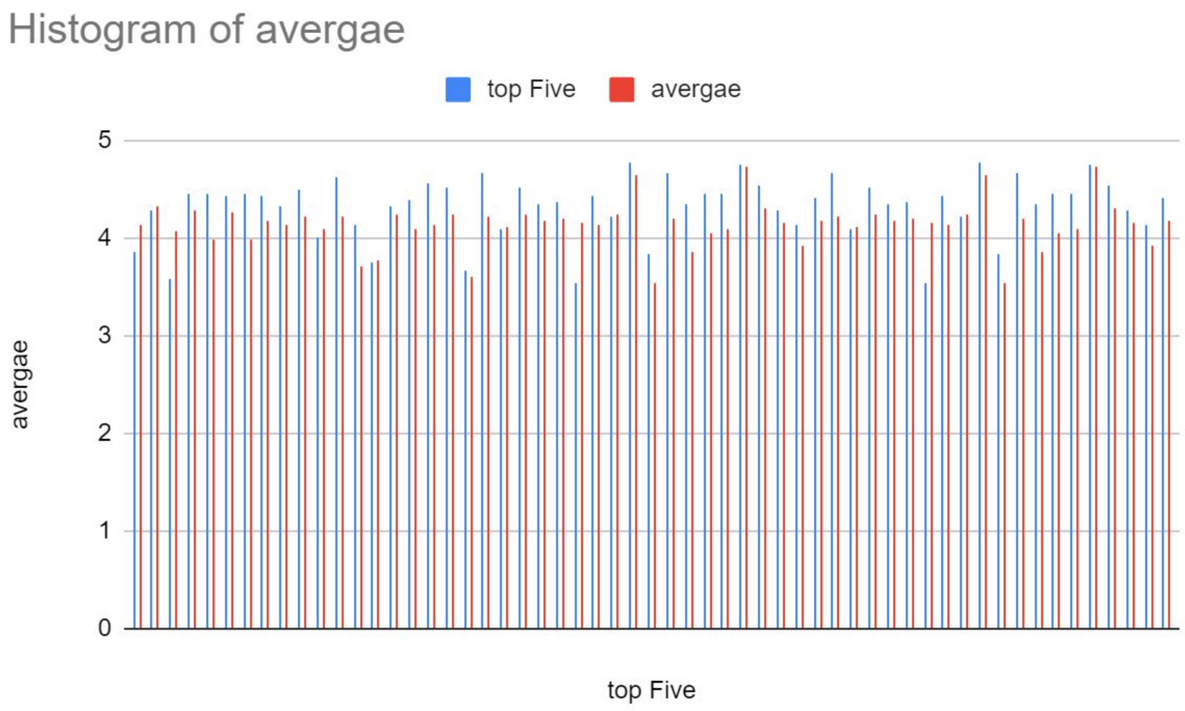
```
            totals += Integer.parseInt(ax[9]);
         }

         de = (float) (((double)helpful/totals)*100);
//          System.out.println(de);
//         System.out.println(helpful/totals);
        avg = getMedian(al);
        int counter=0;
          for (Map.Entry<String, Double[]> entry : reviewUniq.entrySet()) {
                temps = entry.getValue();
                if(temps[0]>=avg){
                    if(temps[1]>=de){
                        counter++;
                                      }
                    else{
                         counter--;
                    }
                }

          }

         bw.flush();
         bw.close();
         System.out.println(counter);
          System.out.println(avg);
         if(counter>0){
         System.out.println("Yes");
         }
         else{
         System.out.println("No");
         }
}
```

**<u>Observation 5 (amazon4.java and amazon4divert.java)</u>**
**<u>Question:</u>** Are more reviewed products rated higher?

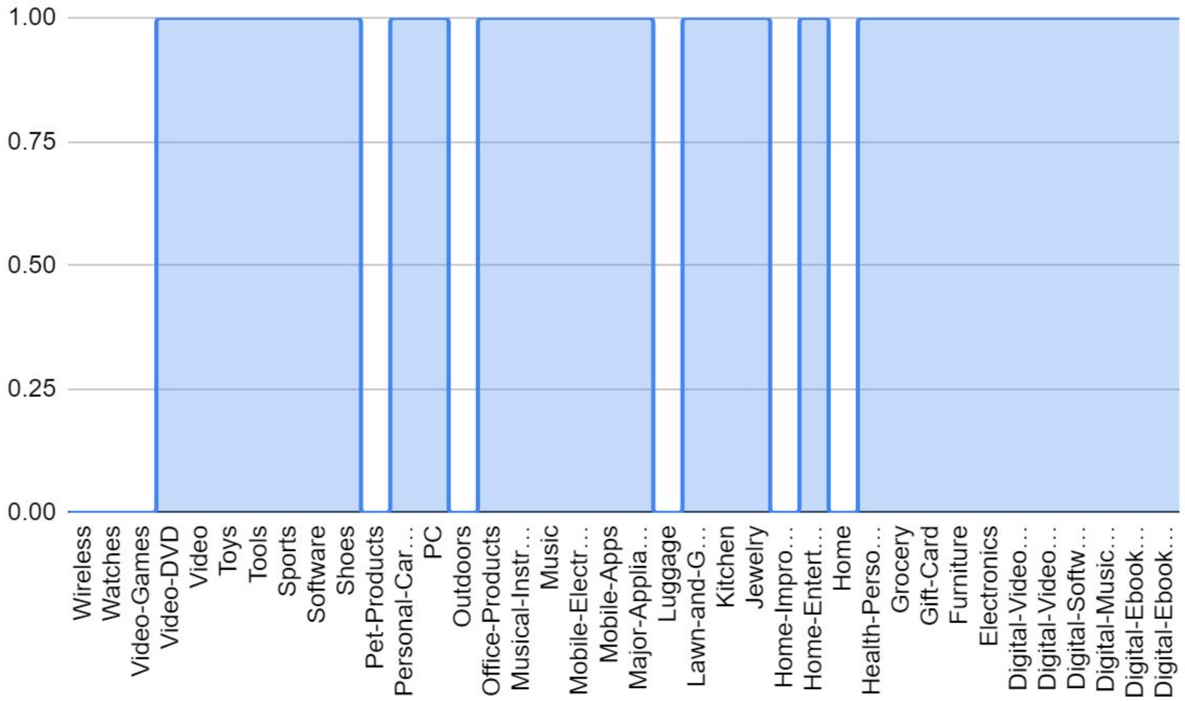
For this calculation we are looking at the Amazons dataset and finding whether more reviewed products are also rated higher. We are doing this by first finding the top five most reviewed products and their average rating as well as average rating for all the products in the dataset. We don't find a general trend and it differs from category to category whether more reviewed products are rated higher.

The below graphs represent the rating for top 5 products in each category as well as the average rating for all products in the category.





Whether more reviewed items are reviewed higher:



We also run the code on taking the entire dataset as a single data and find that more reviewed products are also rated higher. While the top 5 average rating is 4.4361687 and average rating is 4.1756396
yes more reviewed products are rated higher

**Core Code Logic:**

```
For file in Directory{

  while((line=br1.readLine())!=null){
       ax = line.split("\t");

       if(Integer.parseInt(ax[14].split("-")[0])<2005){continue;}
       totalr++;
       rate = rate+ Integer.parseInt(ax[7]);
       if(!topFive.containsKey(ax[4])){
          topFive.put(ax[4], 1);
          ratings.put(ax[4], Integer.parseInt(ax[7]));
       }
       else{
          int u = topFive.get(ax[4]);
//           System.out.println(u);
          topFive.replace(ax[4], u+1);
          int ew = ratings.get(ax[4]);
          ratings.put(ax[4],ew+Integer.parseInt(ax[7]) );
       }
      }
      topFive = sortByValue(topFive);
      int yup = 0;
      int sum_for_five=0;
      int denominator_for_five=0;
      for (Map.Entry<String,Integer> entry : topFive.entrySet())  {
         yup++;
         if(yup>5){break;}
//          System.out.println(entry.getKey()+"\t"+entry.getValue());
//          System.out.println(entry.getKey()+"\t"+ratings.get(entry.getKey()));
         sum_for_five  += ratings.get(entry.getKey());
         denominator_for_five += entry.getValue();
      }
```

```
         float average_for_five = (sum_for_five/(float)denominator_for_five);
        float average_for_total = (rate/(float)totalr);
        System.out.print(average_for_five+"\t");
        System.out.println(average_for_total);
        if(average_for_five>average_for_total){
        System.out.println("yes more reviewed products are rated higher");
        }
        else{
            System.out.println("no more reviewed products are  not rated higher");
        }
}
```

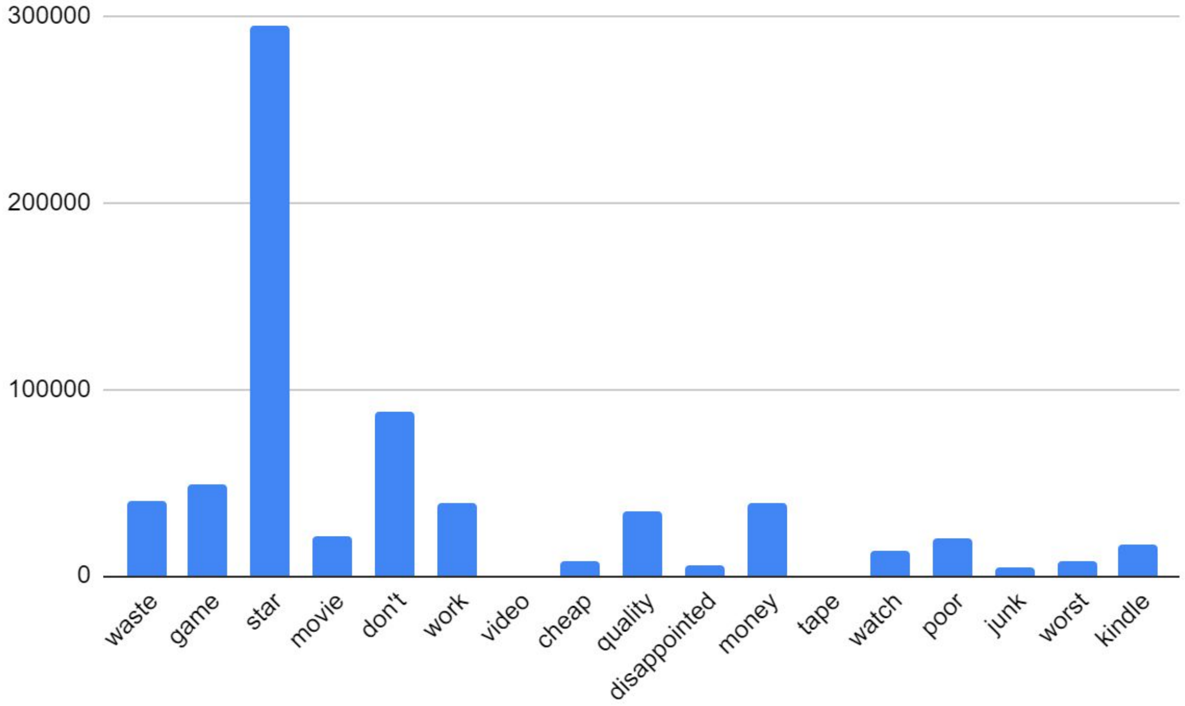
**Observation 6 (amazon8.java)**

**Question:** Most common word in 1-5 star reviews title.

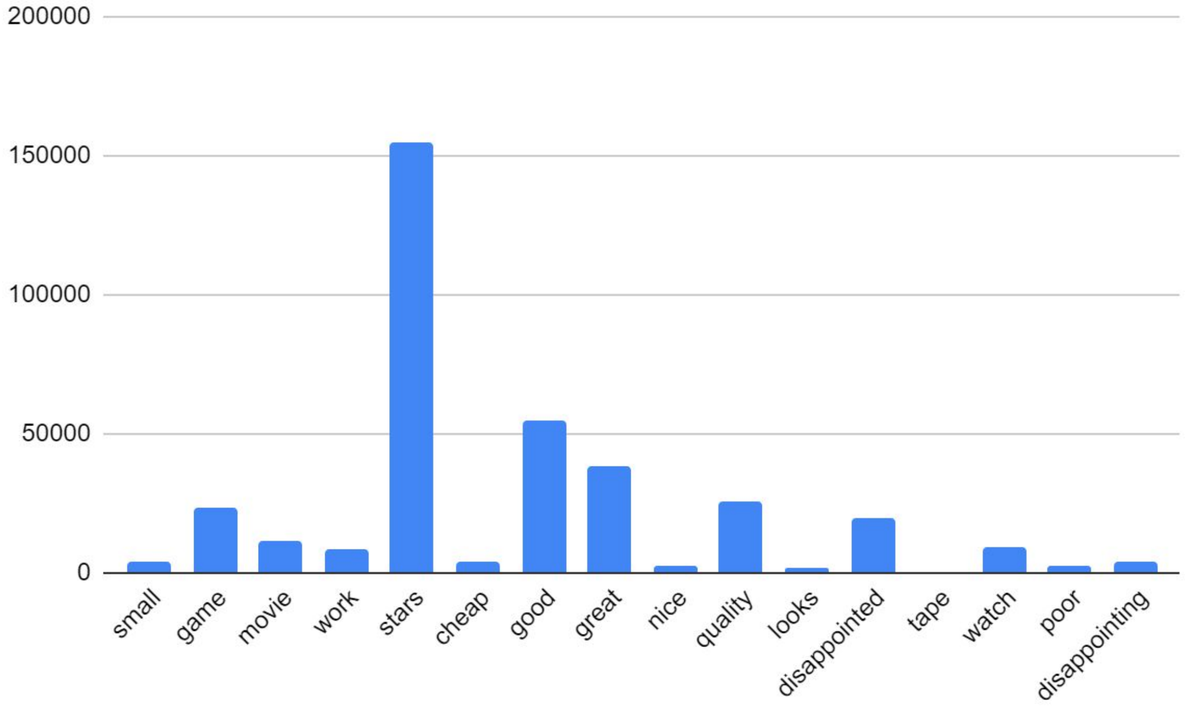
For this calculation we are looking at the Amazons dataset and finding most common words used in the review title as well as their frequency in a plot.

For this we are storing the frequency of each word appearing in the dataset. Now, for each dataset we are picking the top 5 words, after doing this for all categories we are taking an union of all these top 5 words from all categories. We have also excluded common words such as  this , that , it's , more , have , with , product , just , very , when , they , your , these , would , from , will , after , like , them , back , only , /><br , what from the list to give a more relevant result .The words are what we expect based on the ratings

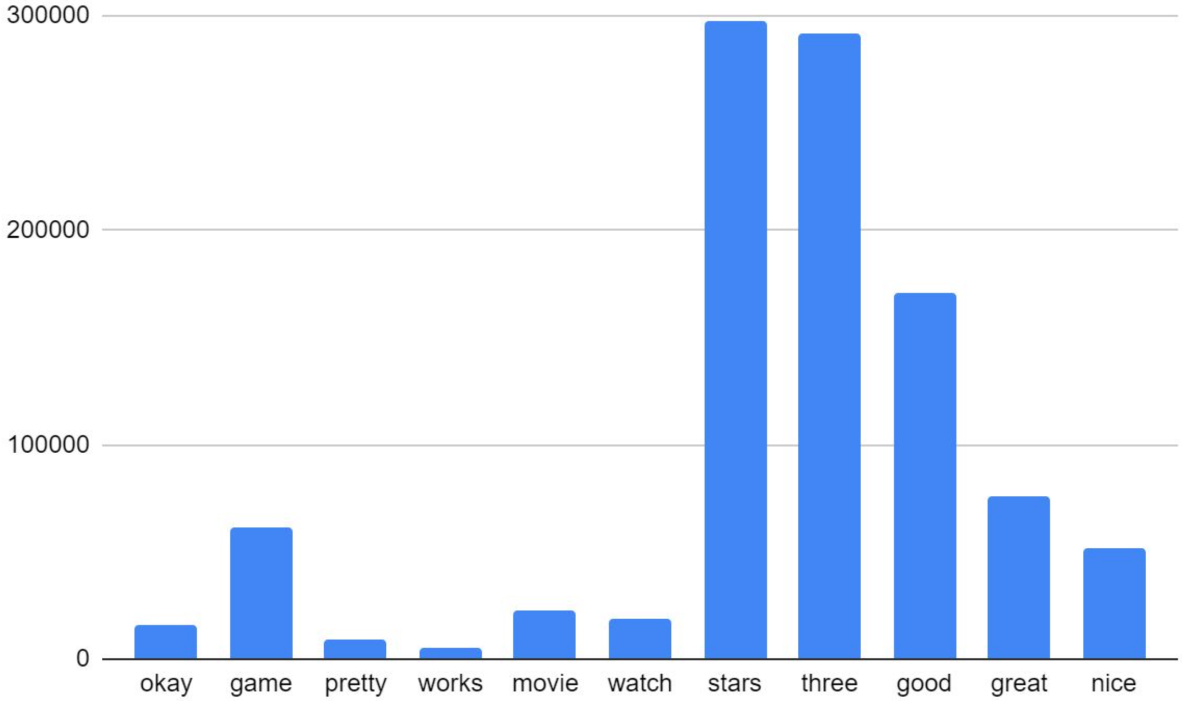
1 star



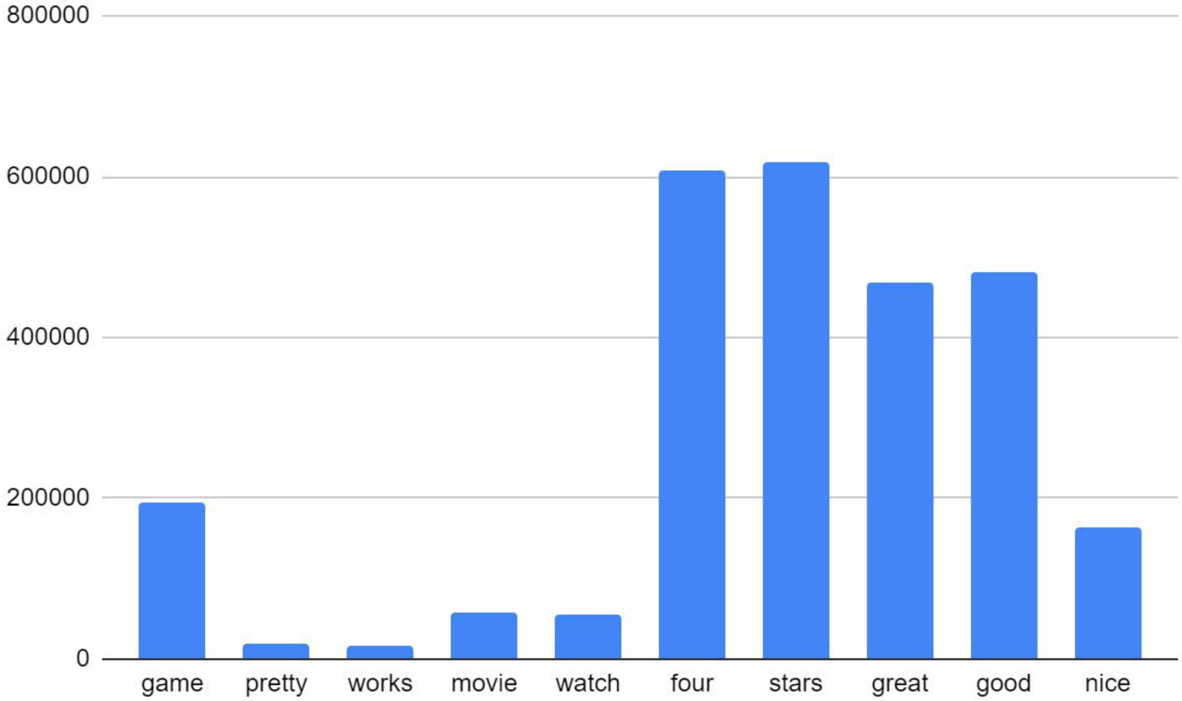
2 star


200000
150000
100000
50000
0
small
game
movie
work
stars
cheap
good
great
nice
quality
looks
disappointed
tape
watch
poor
disappointing

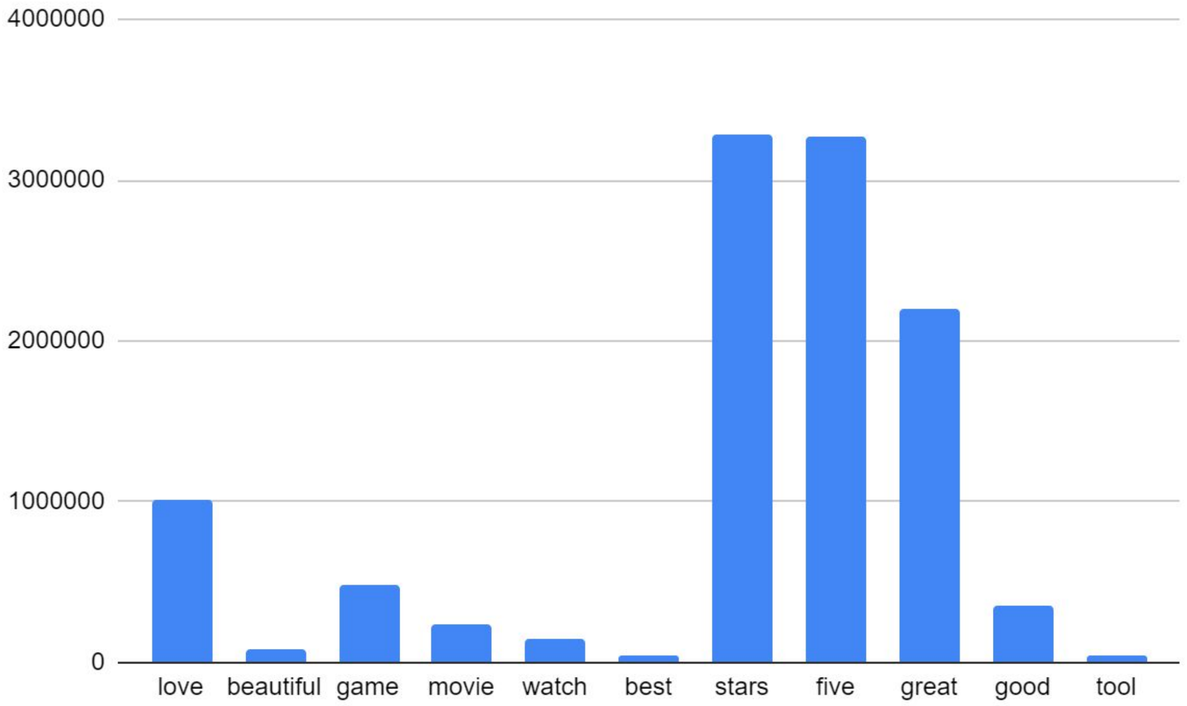
3 star


300000
200000
100000
0
okay
game
pretty
works
movie
watch
stars
three
good
great
nice

4 Star


800000
600000
400000
200000
0
game
pretty
works
movie
watch
four
stars
great
good
nice

5 star



## Code core logic:

```
if(readFile.isDirectory()) {
    for(File file : readFile.listFiles()) {
        //process all files in the directory
    y++;
     HashMap<String,Integer> topFive  = new HashMap<>();
             HashMap<String,Integer> ratings  = new HashMap<>();
        FileReader fileReader = new FileReader(file);
         BufferedReader br1 = new BufferedReader(fileReader);
          File myObj = new File("D:\\amzntext\\asw"+y+".txt");
    if (myObj.delete()) {
    } else {
    }
          FileWriter myWriter = new FileWriter("D:\\amzntext\\asw"+y+".txt");
             BufferedWriter bw = new BufferedWriter(myWriter);
       br1.readLine();
        String[] ax;
        Float[] temp;
        String line;
        int totalr = 0,rate=0;
        ArrayList<Integer> al = new ArrayList<>();

            Double[] temps = new Double[2];
            HashMap<String,Double[]> reviewUniq = new HashMap<>();
         while((line=br1.readLine())!=null){
           ax = line.split("\t");

           if(Integer.parseInt(ax[14].split("-")[0])<2013){continue;}
           if(Integer.parseInt(ax[7])!=5){continue;}
            bw.write(ax[13]+"\n");

    }
         bw.flush();
         myWriter.close();
         HashMap<String,Integer> counts  = new HashMap<>();
      try (BufferedReader br = new BufferedReader(new
FileReader("D:\\amzntext\\asw"+y+".txt"))) {
    String line1;
    while ((line1 = br.readLine()) != null) {
       String[] po  = line1.split("[ .,\"]");

       for(String fer: po){
           fer = fer.toLowerCase().strip();
           if(fer.length()<4){continue;}
           String[] listToRemove =
{"this","that","it's","more","have","with","product","just","very","when","they","your","these","would
","from","will"
           ,"after","like","them","back","only","/><br","what"};
       boolean check = false;
       for(String grt:listToRemove){
           if(fer.equals(grt)){
               check = true;

           }
       }
           if(check){
               continue;
                   }
       if(!counts.containsKey(fer)){
           counts.put(fer, 1);
       }
       else{
```

```
       int tempest = counts.get(fer);
       counts.replace(fer, tempest+1);
       }

       }
    }
    counts = sortByValue(counts);
    int yup=0;
    for (Map.Entry<String,Integer> entry : counts.entrySet())  {
             yup++;
             if(yup>5){break;}

              if(!finalCat.containsKey(entry.getKey())){
                  finalCat.put(entry.getKey(),entry.getValue());
              }
              else{
                  int tpo = finalCat.get(entry.getKey());
                  tpo += entry.getValue();
                  finalCat.replace(entry.getKey(), tpo);
              }

    }

}
}
    }
          for (Map.Entry<String,Integer> entry : finalCat.entrySet()){
        System.out.println(entry.getKey()+"\t"+entry.getValue());
    }
```

## Observation 7 (amazon7.java)

**Question:** Most common phrases in 1-5 star reviews title.
For this calculation we are looking at the Amazons dataset and finding most common phrases used in the review title. Phrases are combination of 2-3 words
For this we are storing the frequency of phrases appearing in the dataset. Now, for each dataset we perform a polling to see which phrases are most common for each category. These gives us an idea base of the category

## 1 star

Watches --> [junk, disappointed, one star]
Video DVD --> [terrible, disappointed, one star]
Video Games --> [do not buy, waste of money, one star]
Video --> [disappointed, disappointing, one star]
Personal_Care_Appliances --> [don't waste your money, waste of money, one star]
Tools --> [don't waste your money, junk, one star]
Mobile_Apps --> [horrible, sucks, one star]
Home --> [waste of money, disappointed, one star]
Jewelry --> [cheap, disappointed, one star]
Kitchen --> [waste of money, disappointed, one star]
Lawn and Garden --> [junk, waste of money, one star]
Luggage --> [disappointed, poor quality, one star]
Major Appliances --> [do not buy, junk, one star]
Mobile_Electronics --> [do not buy, junk, one star]
Outdoors --> [poor quality, junk, one star]
PC --> [waste of money, junk, one star]
Pet Products --> [don't waste your money, waste of money, one star]
Shoes --> [poor quality, disappointed, one star]
Software --> [don't waste your money, waste of money, one star]
Sports --> [waste of money, junk, one star]
Apparel --> [disappointed, too small, one star]
Automotive --> [waste of money, junk, one star]
Baby --> [disappointed, waste of money, one star]
Beauty --> [disappointed, waste of money, one star]
Camera --> [waste of money, junk, one star]
Digital_Music_Purchase --> [not good, nope, one star]
Digital_Software --> [don't waste your money, waste of money, one star]
Digital_Video_Download --> [boring, don't waste your time, one star]
Digital_Video_Games --> [do not buy, waste of money, one star]

## 2 star

Watches --> [cheap, disappointed, two stars]
Video DVD --> [disappointing, disappointed, two stars]
Video Games --> [meh, disappointed, two stars]
Video --> [poor quality, disappointed, two stars]
Personal_Care_Appliances --> [didn't work for me, disappointed, two stars]
Tools --> [poor quality, disappointed, two stars]
Mobile_Apps --> [disappointed, boring, two stars]
Home --> [poor quality, disappointed, two stars]
Jewelry --> [cheap, disappointed, two stars]
Kitchen --> [poor quality, disappointed, two stars]
Lawn and Garden --> [poor quality, disappointed, two stars]
Luggage --> [poor quality, disappointed, two stars]
Major Appliances --> [noisy, disappointed, two stars]
Mobile_Electronics --> [not worth it, disappointed, two stars]
Outdoors --> [too small, disappointed, two stars]
PC --> [poor quality, disappointed, two stars]
Pet Products --> [too small, disappointed, two stars]
Shoes --> [disappointed, too small, two stars]
Software --> [not what i expected, disappointed, two stars]
Sports --> [too small, disappointed, two stars]

### 3 star

Watches  --->  [it's ok, nice watch, three stars]
Video DVD  --->  [good, okay, three stars]
Video Games  --->  [it's ok, meh, three stars]
Video  --->  [good video, okay, three stars]
Personal_Care_Appliances  --->  [its okay, it's ok, three stars]
Tools  --->  [it's ok, just ok, three stars]
Mobile_Apps  --->  [good, fun, three stars]
Home  --->  [okay, just ok, three stars]
Jewelry  --->  [okay, nice, three stars]
Kitchen  --->  [okay, just ok, three stars]
Lawn and Garden  --->  [it's ok, just ok, three stars]
Luggage  --->  [just ok, too small, three stars]
Major Appliances  --->  [noisy, just ok, three stars]
Mobile_Electronics  --->  [just ok, good, three stars]
Outdoors  --->  [meh, just ok, three stars]
PC  --->  [it's okay, it's ok, three stars]
Pet Products  --->  [it's okay, okay, three stars]
Shoes  --->  [runs small, too small, three stars]
Software  --->  [it's ok, okay, three stars]
Sports  --->  [okay, it's ok, three stars]

### 4 star

Watches  --->  [great watch, nice watch, four stars]
Video DVD  --->  [good, good movie, four stars]
Video Games  --->  [good game, good, four stars]
Video  --->  [good classics, good, four stars]
Personal_Care_Appliances  --->  [good product, good, four stars]
Tools  --->  [good product, good, four stars]
Mobile_Apps  --->  [good game, fun, four stars]
Home  --->  [good, nice, four stars]
Jewelry  --->  [pretty, nice, four stars]
Kitchen  --->  [good, good product, four stars]
Lawn and Garden  --->  [good, good product, four stars]
Luggage  --->  [good, nice, four stars]

```
Major Appliances  --->   [good product, good, four stars]
Mobile_Electronics  --->   [good, good product, four stars]
Outdoors  --->   [good product, good, four stars]
PC  --->   [good product, good, four stars]
Pet Products  --->   [good, good product, four stars]
Shoes  --->   [good, nice, four stars]
Software  --->   [good, easy to use, four stars]
Sports  --->   [good product, good, four stars]
```

**5 star**

```
Watches  --->   [love it, great watch, five stars]
Video DVD  --->   [great, great movie, five stars]
Video Games  --->   [great, great game, five stars]
Video  --->   [great, great movie, five stars]
Personal_Care_Appliances  --->   [great, great product, five stars]
Tools  --->   [great product, great, five stars]
Mobile_Apps  --->   [fun, awesome, five stars]
Home  --->   [great product, love it!, five stars]
Jewelry  --->   [love it, beautiful, five stars]
Kitchen  --->   [love it!, great product, five stars]
Lawn and Garden  --->   [great, great product, five stars]
Luggage  --->   [great bag, love it!, five stars]
Major Appliances  --->   [great, great product, five stars]
Mobile_Electronics  --->   [great, great product, five stars]
Outdoors  --->   [great, great product, five stars]
PC  --->   [great, great product, five stars]
Pet Products  --->   [great, great product, five stars]
Shoes  --->   [great shoe, great shoes, five stars]
Software  --->   [great, great product, five stars]
Sports  --->   [great, great product, five stars]
Apparel  --->   [perfect, love it, five stars]
Automotive  --->   [great, great product, five stars]
Baby  --->   [great product, love it!, five stars]
Beauty  --->   [love it!, great product, five stars]
Books  --->   []
Camera  --->   [great, great product, five stars]
Digital_Ebook_Purchase  --->   []
Digital_Music_Purchase  --->   [love it, great song, five stars]
Digital_Software  --->   [great, great product, five stars]
```

Digital_Video_Download  --->   [great movie, great show, five stars]

Digital_Video_Games  --->   [great, great game, five stars]

Electronics  --->   [great, great product, five stars]

Furniture  --->   [love it, love it!, five stars]

Gift Card  --->   [great gift, gift card, five stars]

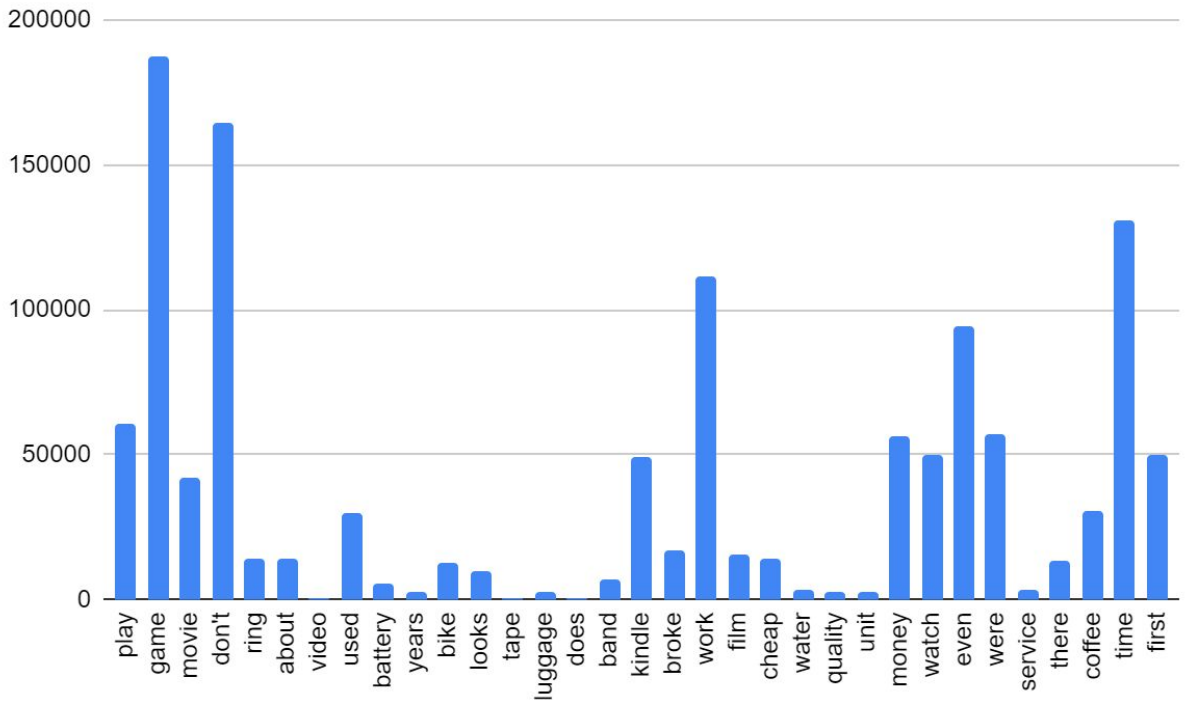
**Observation 7 (amazon5.java)**

**Question:** Most common word in 1-5 star reviews body.

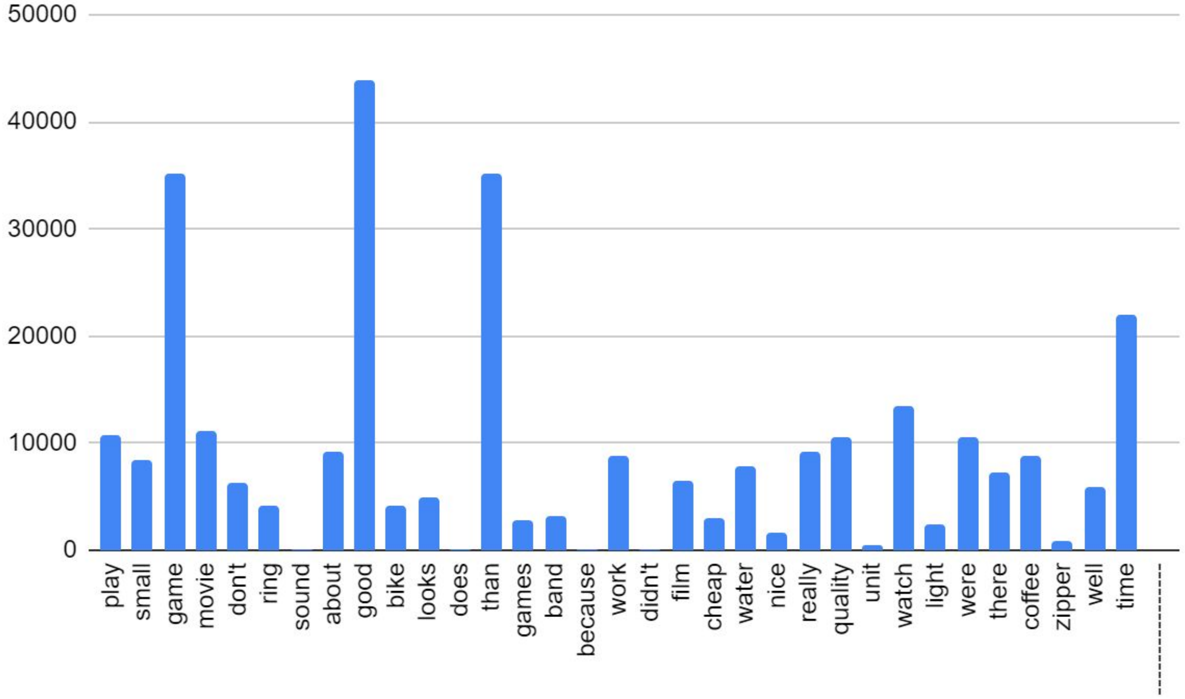
For this calculation we are looking at the Amazons dataset and finding most common words used in the review body as well as their frequency in a plot.

For this we are storing the frequency of each word appearing in the dataset. Now, for each dataset we are picking the top 5 words, after doing this for all categories we are taking an union of all these top 5 words from all categories. We have also excluded common words such as this , that , it's , more , have , with , product , just , very , when , they , your , these , would , from , will , after , like , them , back , only , /><br , what from the list to give a more relevant result .The words are what we expect based on the ratings.

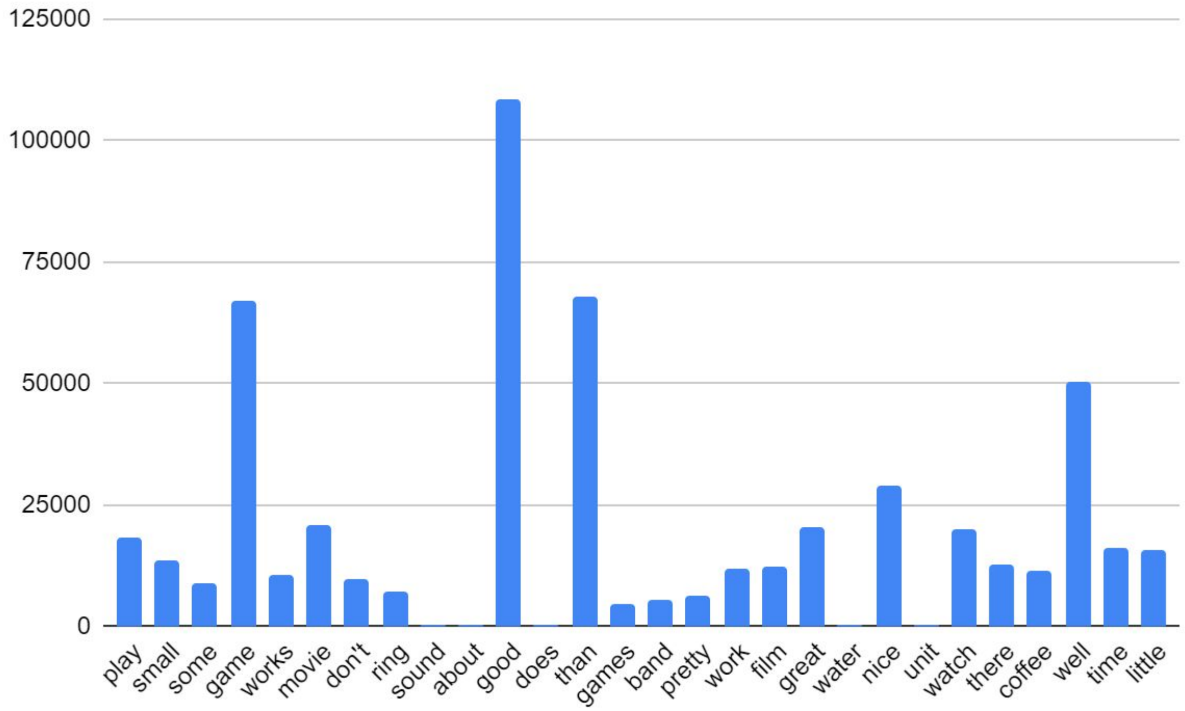
**1 star**



**2 star**

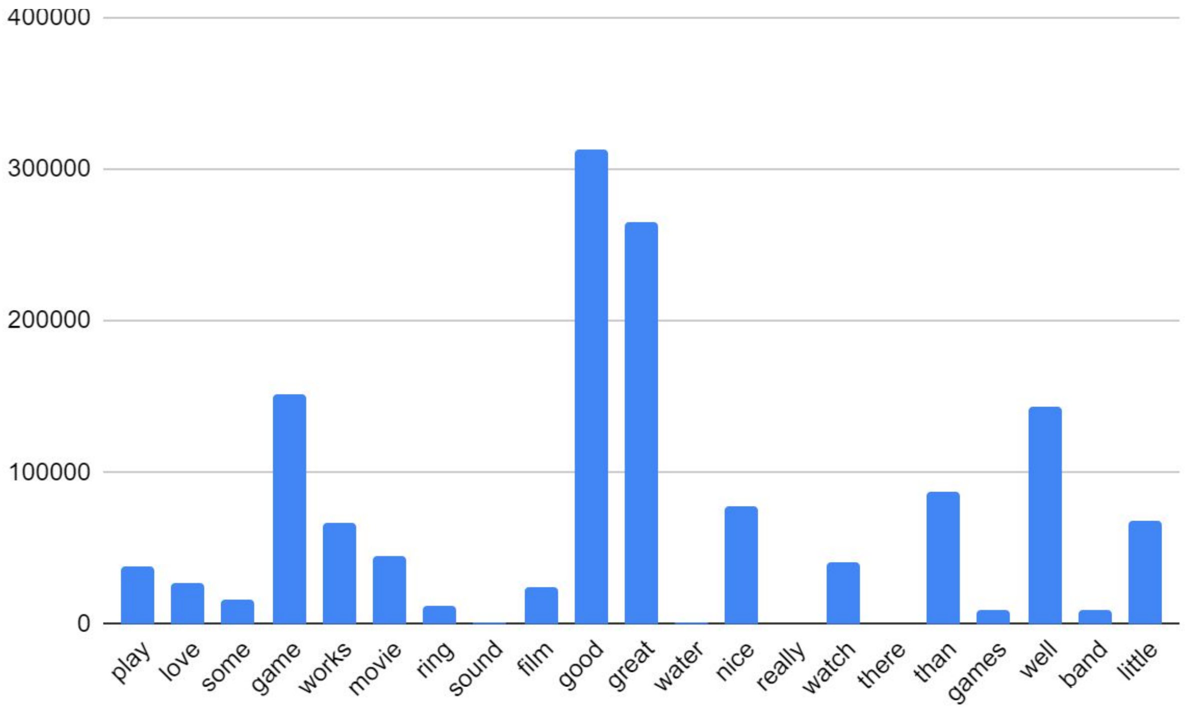

50000
40000
30000
20000
10000
0
play
small
game
movie
don't
ring
sound
about
good
bike
looks
does
than
games
band
because
work
didn't
film
cheap
water
nice
really
quality
unit
watch
light
were
there
coffee
zipper
well
time
-----------

**3 star**



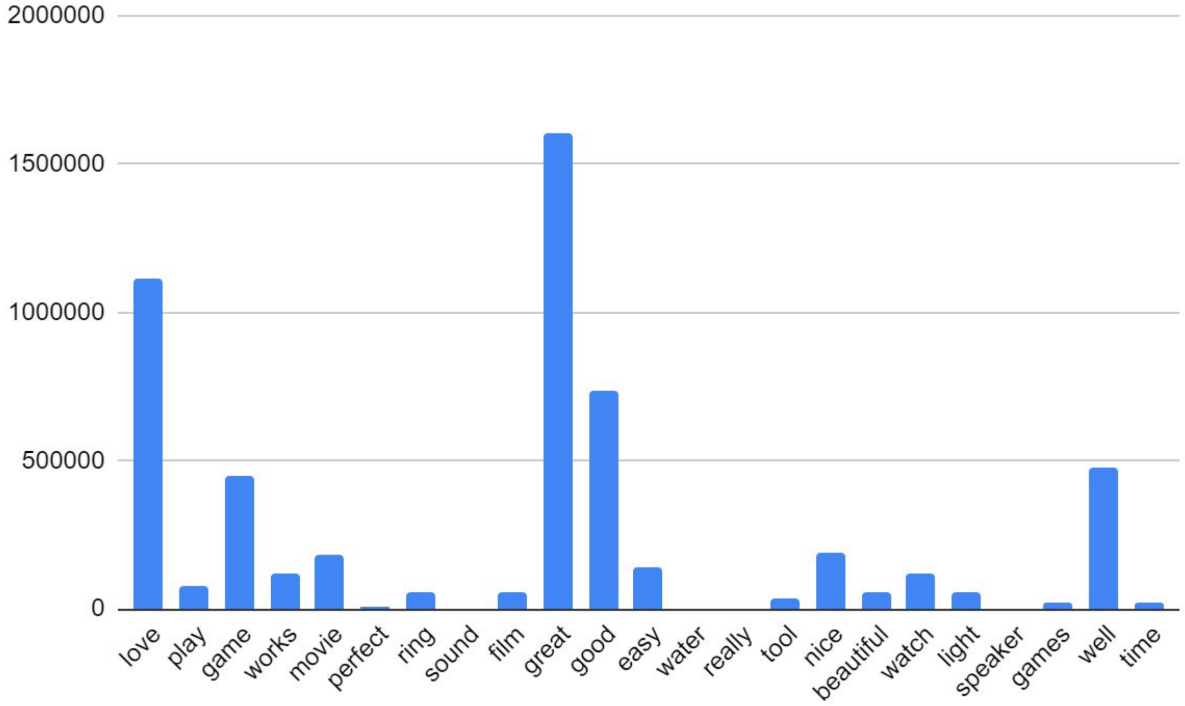
125000
100000
75000
50000
25000
0
play
small
some
game
works
movie
don't
ring
sound
about
good
does
than
games
band
pretty
work
film
great
water
nice
unit
watch
there
coffee
well
time
little

**4 star**



400000
300000
200000
100000
0
play
love
some
game
works
movie
ring
sound
film
good
great
water
nice
really
watch
there
than
games
well
band
little

**5 star**


2000000
1500000
1000000
500000
0
love
play
game
works
movie
perfect
ring
sound
film
great
good
easy
water
really
tool
nice
beautiful
watch
light
speaker
games
well
time