Twin Cities R User Group: Pricing Sentiment Analysis

Armin Kakas

Wednesday, October 14, 2015

Objective

Below, we will do some very simple strategies in analyzing consumer reviews for roughly 80 Walmart products with the ultimate goal of measuring **product level pricing sentiment scores on a daily basis**.

Written customer comments and reviews on company websites can be a genuine expression of their sentiment on various product attributes. Retailer websites provide a wealth of information related to this, and with some simple text analytics, we can evaluate:
- How customers feel about a product's price over time?
- What customer think about product quality, functionality, etc.?
- How recent shifts in pricing strategy influenced consumer price perception?
- How recent shifts in advertising strategy influenced consumer perception of quality?
- How to segment products based on text mining outcomes?
- What pricing strategies to further pursue based on our text analysis *(elasticity and revenue or margin based strategies no longer suffice)*?

I believe that NLP is a much more powerful way of gauging what customers think and feel about **pricing** as opposed to consumer satisfaction scores (or **NPS scores** for that matter), which are inherently biased for purchasers vs. non-purchasers.

If you have the time, I encourage you to read the code, and make suggestions on how to better improve the methodology of evaluating pricing sentiment (which, admittedly, is quite basic below).

Our data

Obtained product reviews and ratings for appr. 100 items. I used Python's **Beautiful Soup** package that works quite elegantly. The attributes are:
1. **date**: date of the review
2. **rating**: the rating the customer gave to the particular item on a particular day
3. **comment**: the actual text of the written customer comment
4. **wmt_product_name**: the name of the product displayed on the Walmart website
5. **wmt_product_id**: the Walmart product id associated with the product

Data and code can be found on https://github.com/KakasA09/TCRUGOct2015

Keywords

These are pricing keyword (see below). We will parse the Walmart customer comments to see if they include one of these keywords. It is certainly not a bullet-proof way of delineating **pricing-related** comments, and robust NLP methods are needed - *pained to admit, but Python is best for that.*

```
load('comments.Rda')
load('pricing_sentences.Rda')
keywords <- read.csv('pricing_keywords.csv')
keywords <- as.character(keywords$keyword)

keywords[1:5]
```

[1] "bargain" "bill" "bounty" "charge" "cheap"

Parsing pricing-related comments

In the below, we are doing three things:
1. Only keep the customer comments that contain one of our **pricing keywords**.
2. Split the comments into sentences using the **qdap** package's **sentSplit** function. *(imagine splitting a comment comprised of 5 sentences, thus 1 row of data becoming 5 rows).*
3. We repeate the earlier exercise by taking a look at our sentences*, and only keeping the ones that have one or more of the **pricing keywords**. We are, in essence, keeping **pricing-related sentences** only. Keep in mind, we still retained the product, date and rating fields.

Now, ideally we would go a step further, and keep pricing-related **ngrams** only. Reason is simple: a customer can make a long statement (sentence), and while the overall sentiment of the statement may be highly negative, her sentiment about the product's price could have been quite positive..e.g.: *"Product quality was awful, but Walmart's prices are the best!"*

```
################# Don't run this
#parse out pricing only comments
options(width = 1000)
comments = tbl_df(comments) %>% select(wmt_product_id, date, comment)
comments = comments[complete.cases(comments$comment),]

pricing_comments1 <- comments  %>% mutate(keyword_present = grepl(paste(keywords,collapse=" | "), commen
    filter(keyword_present == TRUE) %>% select(-keyword_present)

pricing_comments1 = data.frame(pricing_comments1)

#split comments to form unique sentences
pricing_sentences = sentSplit(pricing_comments1, "comment")

#save(pricing_sentences, file = 'pricing_sentences.Rda')
#################


load('pricing_sentences.Rda')

pricing_sentences <- tbl_df(pricing_sentences)  %>%
    mutate(keyword_present = grepl(paste(keywords,collapse="|"), comment, ignore.case = TRUE)) %>%
    filter(keyword_present == TRUE) %>%
    select(wmt_product_id, date, comment)


#ensure there is complete data
pricing_sentences <- na.omit(pricing_sentences)
```

Creating customized dictionaries for negative, positive, negation words, amplifiers and de-amplifiers

Since our approach for measuring customers' **pricing sentiment score** is a slightly enhanced version of **lexicon-based scoring**, we will make it a bit more robust by adding industry-specific terms.

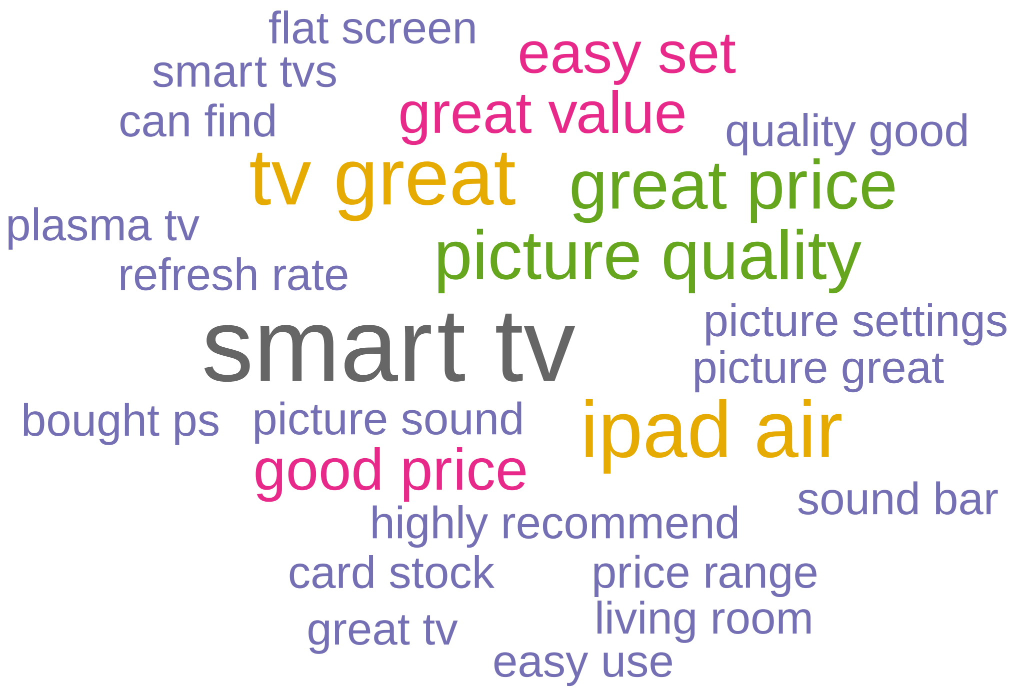
Sentiment scoring with parallelization

In the below, we will use the versatile **qdap** package to formulate **sentiment polarity scores** (constrained between +1 and -1). **Qdap** enables us to score our data by grouping factors: in our case by **product** and by **date**.

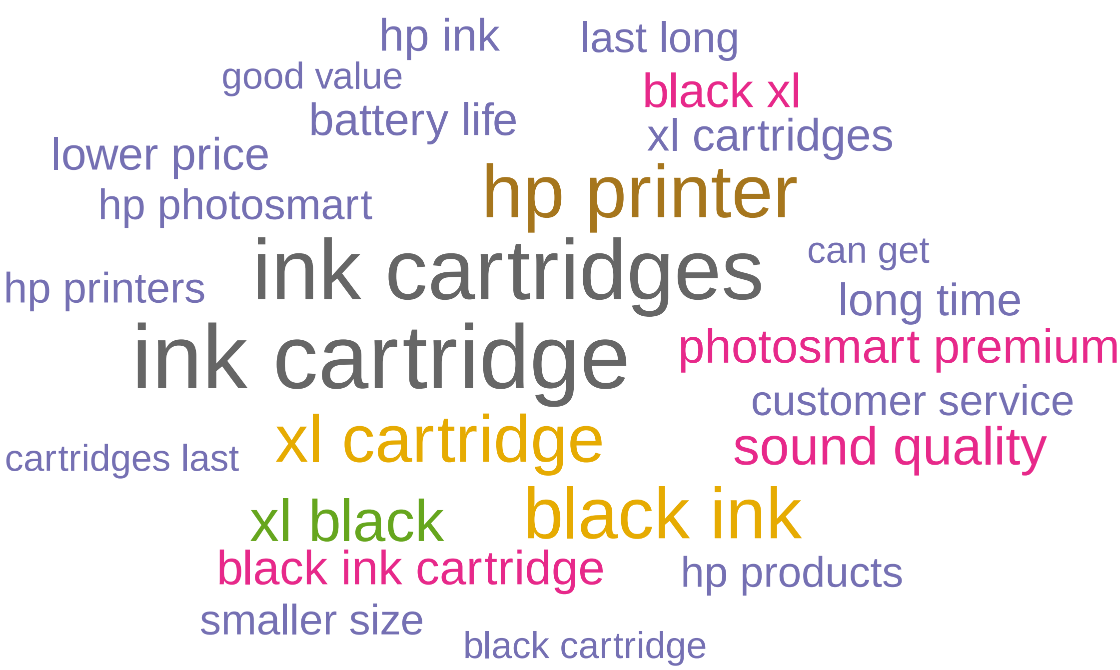
```
wmt_product_list = unique(pricing_sentences$wmt_product_id)

library(doMC)
registerDoMC(cores = detectCores()-2)

pricing_polarity_scores <- foreach(a = 1:length(wmt_product_list),.combine = 'rbind',
                                   .packages = c('dplyr', 'qdap', 'tidyr')) %dopar% {

    pricing_sentences_small <- pricing_sentences %>%
                    filter(wmt_product_id == wmt_product_list[a])

    pricing_polarity_small = with(pricing_sentences_small,
              polarity(comment, polarity.frame = pos_negative_words,
                       negators = negation_words, amplifiers = amplification_words,
                          deamplifiers = deamplification_words,
                          list(wmt_product_id, date), constrain = TRUE))

    colsplit2df(scores(pricing_polarity_small))

}

pricing_polarity_scores <- tbl_df(pricing_polarity_scores) %>%
    select(wmt_product_id, date, ave.polarity)
#save(pricing_polarity_scores, file = 'pricing_polarity_scores.Rda')
```

Now, let's do some analysis (see comments in the code below)

Ngrams for best pricing sentiment score products

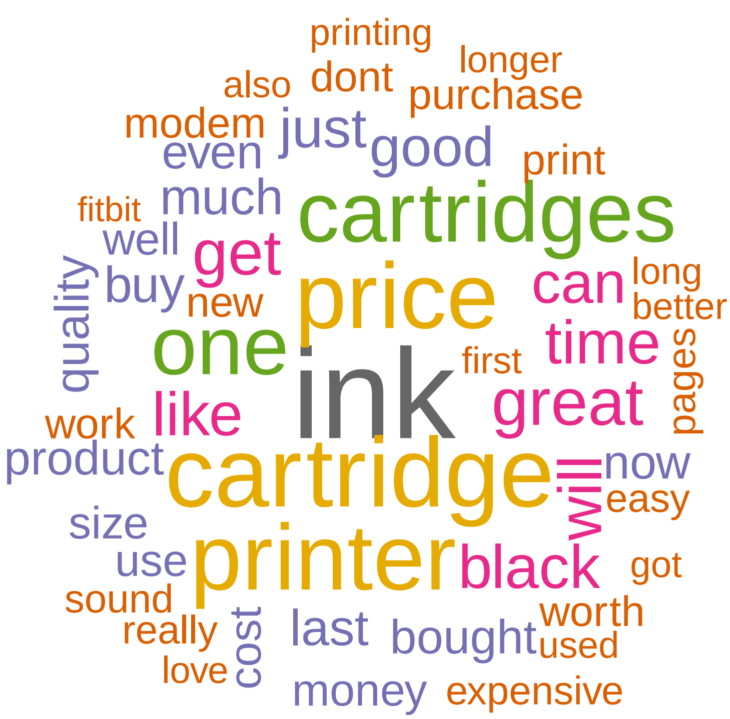


Ngrams for worst pricing sentiment score products



Words for best pricing sentiment score products

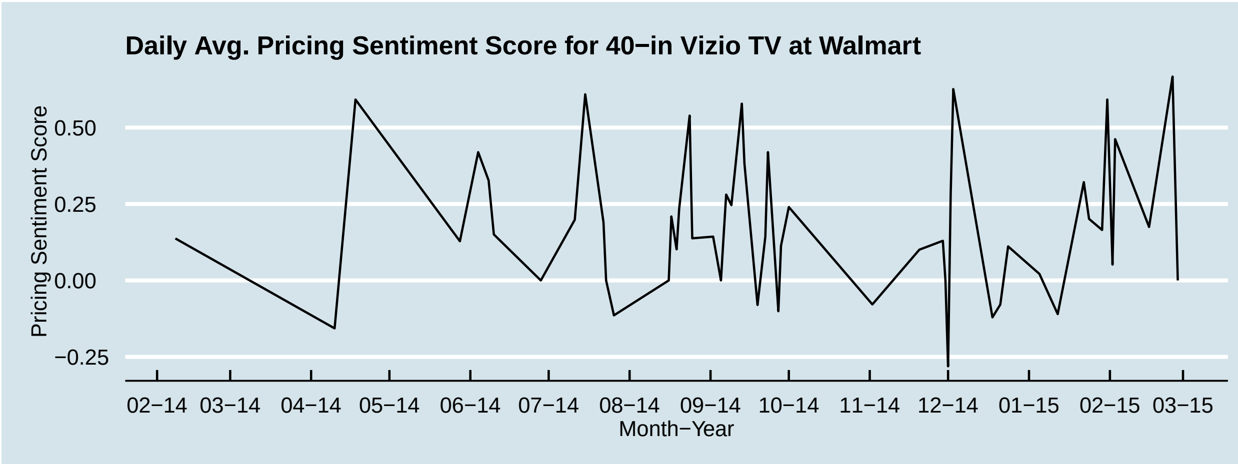
Words for worst pricing sentiment score products

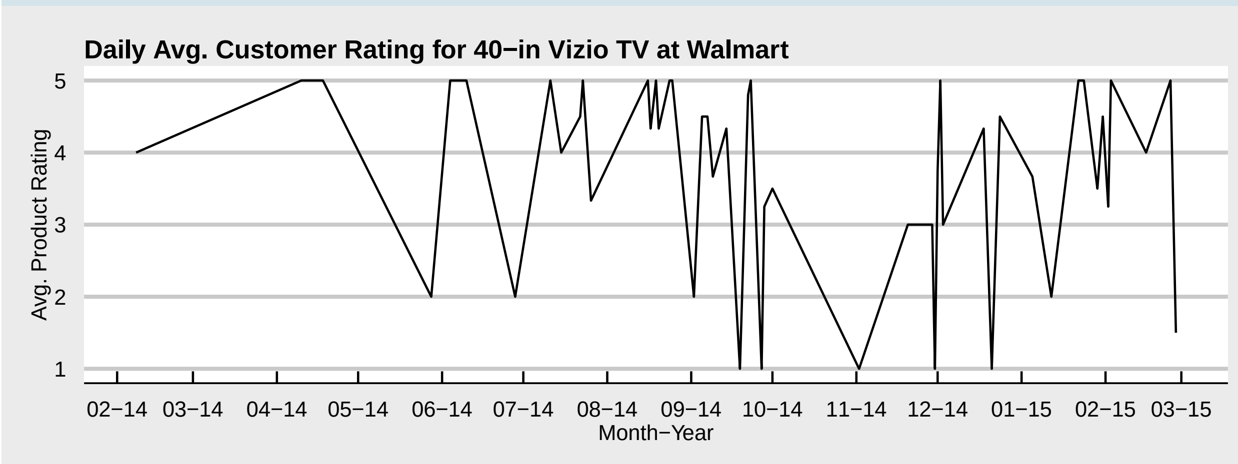


Correlations between product ratings and pricing sentiment

wmt_product_name	wmt_product_id	correlation
VIZIO E400i-B2 40“” 1080p 120Hz Full-Array LED Smart HDTV	33054851	0.4603
HP Consumables CN684WN#140 564XL Black Ink	20657638	0.4518
ARRIS/Motorola SBG6580 SURFboard DOCSIS 3.0 Cable Modem and WiFi-N Router	15567546	0.3955
NETGEAR WNDR3400-100NAS N600 Wireless Dual Band Router	15539745	0.3751
Samsung 32“” 1080p 60Hz LED Smart HDTV, UN32H5203AFXZA	36483178	0.3599
Fitbit Flex Wireless Activity Sleep Band	26469465	0.3152
HP 61 Black/Tri-color Original Ink Cartridges, 2 pack (CR259FN)	15084439	0.1927
Samsung Galaxy Tab S 10.5“” Tablet 16GB	37065363	0.1499
Google H2G2-42 HDMI Streaming Media Player - Wi-Fi - 1080p - Netflix, YouTube, HBO GO, Hulu Plus - Black	33142918	0.148
Roku Streaming Stick, 3500R with Get 60 days FREE* of Rdio Unlimited and Try 3 months FREE of Hulu Plus	35030305	0.1414
Samsung Galaxy Tab 4 7.0“” Tablet 8GB	35822395	0.1304
ZZ Motorola 575319-019-00 SURFboard Docsis 3.0 Cable Mod	20742485	0.07075
Samsung 40“” 1080p 60Hz LED HDTV, UN40H5003AFXZA	39405700	0.05769
Samsung 40“” 1080p 60Hz LED Smart HDTV, UN40H5203AFXZA	36483179	-0.06063

Example of relatively strong correlation between rating and sentiment





Example of a weak correlation between rating and sentiment