Shelf Unique IP: Provisional Patent Claims + Trade Secret

Lauren Jung
Shelf Inc.

Atul Singh
Shelf Inc.

Abstract

Systems and methods are disclosed for analysis and inference of promotions sent by retailers; to find the lowest price and availability of retailers items; to recommend ways to redeem promotions.

1 Claim: A system to discover, analyze, and infer promotions

Input:
Retailer Promotion Sources (Web URL, promotional email, social media URLs)

Output:
A list of promotions, each a tuple containing:
<Discount info (type, code, discount value, threshold)>, <Validity>, <Availability>, <Applicability>

Type = {Store-wide, Buy1-Get1, Aggregate, Shipping}
Code = {alpha-numeric or None}
Discount value = {E.g., 20%}
Threshold = {E.g., $100 for a promotion "Spend $100 to Save $25"}
Validity= {Start-date, End-date}
Availability = {In-store, Online, Both}
Applicability = {Everything, Category list, Excludes}

High-level idea We first get the raw information from the sources (email, HTML from the store's online pages as well as Facebook/Twitter feeds) and find out promotion related information in easy to analyze textual format. This requires us to parse the information present in different syntaxes (HTML, email, images) to a common format. Next, the raw input is parsed and mapped to a set of promotion rules using machine-learning based pattern matching algorithm. Figure **??** captures this.

1.1 Raw-input to text translation

For Web URL source (similar for other sources) Start:
a) Crawl the home page using a standard off-the-shelf web browser emulator (such as Selenium) to crawl the Web URL
b) Find all leaves of the DOM tree of the web-page that contain some text inside
c) Filter out leaves that contain promotion-related keywords {promotion, discount, off, save, sale, shipping, sale, season, $, %}
d) For each such leaf: find the sibling DOM elements and their associated text or image (this forms a promotion unit)
e) Find all image elements in the DOM, and for each image, obtain the value of the "alt", "title" text values, id of the "area" map, and click on sibling or children nodes that have text "see details", "details", "more"
f) Find the DOM element of type area with the id found in the previous step, get it's "href"

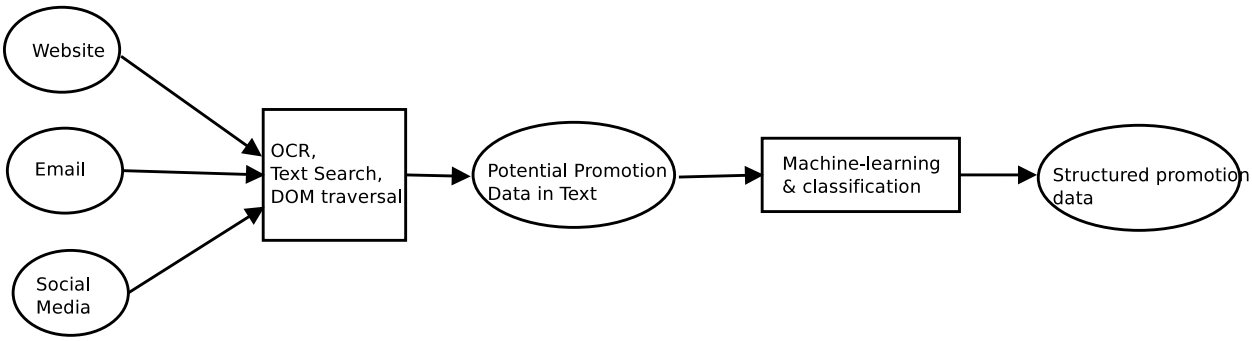

Figure 1: High-level idea.

tag and store it in the next URL to visit next
g) For each image found, perform OCR using a standard off-the-shelf OCR solution (e.g., Google's Tessaract) to extract text from the image (this forms a promotion unit as well)
h) Repeat the above steps for the content obtained by visiting the URL's in the next list

Output of this step is a set of promotion units: a set of text-lines containing potential promotion information.

1.2 Text-to-promotion-rule classification

At a high-level, our algorithm uses decision-tree based machine-learning algorithms to perform automatic classification of text-based promotion units to structured promotion data (or a promotion rule). A promotion rule, as described above, consists of:
<Discount info>, <Validity>, <Availability>, <Applicability>

Attributes: We have identified following attributes:
Discount related: {buy 1, buy one, get, off, a number, spend, free, sale, clearance, shipping, use code, code, use, $, %,}
Validity related: {valid, last, day, tick, tock, (any) day of the week, (any) month of the year, limited time, today, }
Applicability related: {everything, <category name>, men, women, boys, kids, girls,}
Availability related: {store, online, both}

Next, we searched for these attributes in the potential promotion units found in the previous section. The output of this process is a list of vectors, each corresponding to a potential promotion unit. A vector for a token corresponds to the values assigned to the different attributes found in its text. The results are presented in a special format .arff, so that they can be directly processed by most machine learning tools, such as Weka.

2 Trade Secret: Automatic Promotion Application and Availability Testing

Input: - Product URL
- html tags for color/size elements for the store
- html tags for add-item-to-cart for the store
- html tags for checkout for the store
- html tags for element that is needed for adding promo code for the store

Output: for the given product:
a) current lowest price for each color/size,
b) applicable promotion code (if any),

c) availability for each color/size

Algorithm: Our algorithm consists of following steps:
- Start an Internet browser (like Firefox) by invoking Selenium, a browser automation system
- Visit product url
- Find all colors (finding the list of selectable items on the page for the color specific elements)
- For each color, click on it (or select it) and observe the available sizes and prices (after refreshed by the browser)
-This gives us the availability, base price for color/size
- Select a random color and a random size combination
- Add item to the cart by clicking on 'add-to-cart' button
- Save the current price
- Enter each promotion code (obtained from the algorithm of claim #1) to the html DOM element for entering promotion code
- Get the final price and calculate savings due to the promo code
- Remove the promotion code and repeat if there are more promotion codes left

3 Claim 2: Recommendation of Ways to Redeem Promotions

3.1 Aggregate Type of Promotions

Input: A promotion P of type Aggregate (e.g.: "Spend X and Save Y: X > Y") from retailer A and a set of items from retailer A.

Output: for each item, maximal savings achieved due to the promotion.

Algorithm: Our algorithm consists of following steps:

For each item i in items:
if item.price > X:
    savings.amount = Y
    savings.promo = P
    savings.required_item = None
    item.savings = savings
    continue
if item.price > X-Y:
    diff = X - item.price
    // save 25 off $100. Item is 80, so diff = 20, Y =25. So, we end up saving $5
    // this is a feasible space where a user can spend diff and save Y, so overall a savings of Y-diff
    find an item δ from retailer A such that δ.price $\geq$ diff $\wedge$ δ.price is minimum
    // suggest user to buy this item and become eligible for redeeming this promotion
    savings.amount = (Y - δ.price)
    savings.required_item = δ
    savings.promo = P
    item.savings = savings

3.2 B1G1 Type of Promotions

Input: A promotion P of type B1G1 (e.g.: "Buy 1 Jeans, and get another at 50% off") from retailer A, a user U of Shelf, list of items the user is interested in (for buying) from retailer A, and U's online social network S (friends, family: connected via email or online social network) present on Shelf.

Output: List of item-sets.

Basic idea is to find other items that are required for activating the promotion. We first search in the list of items U has shelved and then search in the list of items U's social network has shelved.

Algorithm: Our algorithm consists of following steps:

$C = P$.category
threshold = P.threshold
//testing if promotion P works on an item is our trade secret: can we re-use this?
items_eligible = items from retailer A shelved by U and promotion P works on them
if num(items_eligible) $\geq$ threshold:
    result = find subsets of size threshold from items_cat
    return (U, result)
else:
    // Not enough items shelved by U, so should look at friends items next
    result_set = ()
    for user F in S:
        item_eligible_F =items shelved by F and promotion P works on them
        if num(items_eligible_F) $\geq$ threshold:
            result = find subsets of size threshold from items_eligible_F
            add (F, result) to result_set
    sort result_set based on variety of settings: price-based, popularity-based, or by interest-based
    return result_set