ABSTRACT

Current advertising solutions focus on delivering advertisements that are similar to the products which users have viewed. Many shoppers, however, would prefer to buy a set of items that complement each other rather than redundant items. This is especially true for retail shoppers who must make purchases to complete outfits.

Our system addresses the need for a contextual advertising solution that allows retailers to advertise across one another’s sites. The system implements two user interfaces (UI) : the component and the website. The component is the end deliverable and contains targeted advertisements for the consumer to be placed on product pages of retailers. The website is a secure interface for advertisers to submit bids and advertisements. Between these two UI implements is the key innovation: an advertising bidding process and a retail-specific contextual advertising algorithm. By serving complementary products, the system is designed to be mutually beneficial for both retailers instead of competitive. The advertiser gains greater exposure for its product and brand. Meanwhile, the host retailer can be paid a small price by the advertiser for user clicks its products generate.

1. INTRODUCTION

The information available to companies about web users is tremendous and companies are always looking for more effective ways to monetize that data. One method that companies have identified is in online advertising. These advertisements are of particular importance because they have a much higher probability of user engagement, and ultimately sales, compared to traditional online advertising. The Interactive Advertising Bureau (IAB) reports that the revenue from online advertising has grown exponentially over the past two decades [10]. According to the IAB, online advertising revenue topped $36 billion in 2012 and continues to grow. The sheer size of this industry makes the field of online advertising a very relevant topic of inquiry.

This paper investigates a specific subset of online advertising - contextual advertising. Contextual advertising is online advertising that relies on the content of the web page being displayed to determine the best possible advertisement to deliver. Users encounter contextual advertisements online everyday, and the space is still rapidly expanding.

Despite the prevalence of contextual advertising on many parts of the Web, we believe that there is a lack of effective contextual advertising on retail websites. Purchasing clothing inherently requires complementary purchases; it does not do much good to buy a shirt if you do not also buy a pair of pants. Yet retail sites have failed to implement an effective contextual advertising platform that helps them cross-sell complementary items of clothing based on the item on the page.

More specifically, we are interested in contextual advertising between retailers. Because most larger retailers have obvious motivations for avoiding advertising competitors’ products, we have focused our platform on rotating advertisements of small, niche retailers on the sites of more established ones. A focus of our research is in designing a system that does an appropriate job of leveling the playing field for smaller scale users.

Our investigation has a clear, practical application. A platform like the one that is the focus of this paper would drive more sales for retailers by delivering the most relevant ads for the most relevant consumers and aid consumers in finding more appropriate products to complement what they were already looking for. Such a product would add value for retailers and consumers alike.

To this end, we have designed a bidding system that ensures fairness by altering the most widely-used online advertising bidding mechanism. We will later implement algorithms for serving the best advertisements and catching improper input based on current best practices and tailored for the needs of our system. We have created a web crawler to populate data to evaluate the results of our system for the needs of our system. We have created a web crawler to populate data to evaluate the results of our system for this paper and have also implemented a website to interact with potential users.

In this paper, we will first discuss the current research relevant to our topic. We will follow with a high-level discussion of our system before providing a more granular explanation of our specific implementation and design decisions. Finally, we will summarize the results of our research and outline our general conclusions.

2. RELATED WORK

For our bidding system, we researched the most effective auction mechanisms. The Vickrey-Clarke-Groves mech-
anism (VCG) is a mechanism for “sealed-bid auction,” that is, auctions in which the bids are kept private from other bidders [4]. By this mechanism, the winning bidder pays the price of the second-highest bid. The VCG mechanism ensures that bidders submit what they believe is the true value for the item being bid on.

A “generalized second-price” GSP auction is an adaptation of the VCG mechanism but for bids on multiple advertising slots instead of one [6]. In a GSP auction, winning bidders pay the price of the bid one below theirs. The work by Edelman et. al. also explains the motivations for providers to adopt the GSP auction mechanism: to ensure to their users that they are paying a fair price. This research was very relevant to our system implementation, as GSP is currently the dominant auction mechanism in selling online advertising. However, rather than using this mechanism to ensure users paid a fair price, we also believe that implementing this algorithm could provide fair weights in the weighted-randomized selection algorithm we use to serve advertisements.

The rest of our research has focused on contextual advertising. A team of researchers from Microsoft, led by Thor Graepel, thoroughly explain the thought process behind constructing one of Microsoft’s most profitable business models [7]. Graepel et. al. developed an algorithm to automate a fair, yet complex process to assign advertising slots to advertisers for given searches. In the paper, the authors referred to an ad impression as a certain advertisement being shown to a specific user and described in detail the three distinct input features to make this decision.

1. The first category is made up from the ad features which includes the ad title, ad text, landing page URL, bid phrases, and other information associated with a given advertisement.

2. The second category consists of query features, which includes the user’s search keywords and possible algorithmic search query expansion.

3. The third category contains the context features that comprise the physical area on the page on which the ad is displayed, the geographic location of the user, the time of day and year, the user’s search history, and other relevant user information.

The relationship between ad, query, and context features is widely discussed in the paper as they are each numerically represented and used to calculate the “best possible advertisement.” Something that we will determine next semester is how to apply these categories to retail.

Researchers at the Norwegian University of Science and Technology have done research into advertising based on a person’s clothing style. Shabib et. al. disassociate the user from their clothing, much in the same way ORCA only takes into account the current item being viewed. Their approach relies on the clothes already being worn, as opposed to clothes that a user is looking to purchase, however their research still relies on suggesting contextual advertisements based on clothing. A benefit to this type of system is “there is no need to know the profile of the user,” which produces more context-specific advertisements, and further improves the user’s security [9].

Traditional contextual advertising systems predominantly rely upon text to extrapolate the context, however work by Tao Mei et. al. has shifted this reliance to now focus on images to infer the context [8]. Their research, analyzes images on a webpage to determine the context of the page, and then displays contextual advertisements based upon this determination. Research into advertisement engines such as that done by Mei et. al., that work in the image domain, could be very helpful to our efforts with ORCA, as it would provide us the ability to generate real-time analysis of the clothing item(s) being viewed by a user. We could then use this analysis to further improve the accuracy of the advertisements shown to that user.

Gerald Benoit of Simmons University wrote a paper that discusses color theory, pixels, and bytes [5]. He examined how different combinations of colors on the color wheel can create various visual harmonies. This is applicable to ORCA in the sense that the selected items to display at complementary items should have aesthetic relevance to the current item being viewed by the user. The approach Benoit examines in regards to detecting complementary colors is taking a given color and setting the most complementary color to be directly opposite on the color wheel.

ORCA has been able to build upon the previous research into contextual advertising, and has been able to specialize in the application of retail contextual advertising. Until now, there has not been specific research done into an advertising system specifically designed to generate complementary clothing suggestions. Previous research in the area has been fantastic, and we relied on this previous research in order to create our optimized algorithm. ORCA has been able to comprehensively integrate each of these independent research efforts in order to produce the advertising system.

3. SYSTEM MODEL

Our system consists of two parts: an embeddable component on the retail webpages and a website through which companies that wish to advertise can manage an advertising profile. The underlying idea for the system is to allow retailers to cross-advertise on each other’s sites. Consumer interaction with the technology occurs in the component, while client interaction is conducted on the website.

The component displays three items that our algorithms pick out to complement the current item the user is viewing (the “source item”). The standard implementation of the component is small in size, approximately 500 × 200 pixels, but it has the ability to be scaled up or down in size in order to accommodate the retail site. Clicking on the item redirects to the product page on the advertiser’s website.

The functional requirements of the algorithms and the web-facing interfaces will be discussed in the subsequent subsections.
3.1 Algorithms

In order to generate complementary items, the system runs three algorithms inside the component. The weeding algorithm filters out items that would be inappropriate to display for the given source item. Then the bidding algorithm and compatibility algorithm work simultaneously to compute scores for all of our items. These scores correspond to the probability of being displayed. Finally, items are randomly selected to be displayed.

Weeding Algorithm

The weeding algorithm ensures that inappropriate advertisements are not displayed given the source item. The algorithm takes the source item as input and outputs a list of items generated by a filtered query of the database. The algorithm first queries the database for all items. Then, it eliminates items that have a different gender than the source item’s. Next, items that cannot be worn at the same time as the source item are eliminated. For example, when the source item is jeans, the algorithm eliminates jeans, as well as shorts and skirts. The owner of the source item may also have certain blacklisted items, such as items from a specific retailer. The system filters these results to remove blacklisted items and finalize the output set.

Bidding Algorithm

The bidding algorithm takes as input the item list generated by the weeding algorithm and outputs a list mapping each item to a score. Each item has an associated bid price, which refers to the amount the owner of the item would be willing to pay per click-through. The bidding algorithm assigns higher scores to items with higher bid prices, so that those items will have a higher probability of being selected for the component. To do so we use the Generalized Second-Price (GSP) auction mechanism, which is the industry standard for online advertising. This mechanism takes in the bid price as input, ranks the items in descending order, and assigns each item the bid price of the item below it.

One benefit of GSP is that it prevents the “Winner’s Curse” - or the feeling that the winner has overpaid since he or she outbid everyone else. Further, GSP has the added benefit of preventing a single user from abusing the bidding system. A single bidder cannot dominate the weights of the randomized selection because his or her weight is ultimately determined by the bidders below him or her, ensuring that advertisements are rotated in the fairest way possible.

Compatibility Algorithm

The compatibility algorithm takes the output of the weeding algorithm and the source item as input and generates a list mapping each item to a score. The score here is based on compatibility with the source item. Two factors determine the compatibility score: color and price. Color matching is evaluated by ranking items based on how well their primary color matches the primary color of the source item. Color harmony theory states that two colors match best when they are complementary – or, in other words, on the opposite end of the color wheel. Therefore we essentially rank items by how close their primarily color is to the source item’s complementary color.

Price matching is used to display items that are about as expensive as the source item. For example, if the source item is a $30 pair of jeans, we should not display a shirt worth $250. A more reasonably price matched shirt might be one worth $20. We price match because users viewing an expensive item will seek other expensive items to wear in the same outfit. The same idea applies for less expensive items.

The resulting scores of the color matching and price matching are combined to produce a single compatibility score. Again, items with higher scores have a greater probability of being displayed.

Final Selection

Finally, we generate a final score by combining the bidding and compatibility scores generated by the preceding algorithms. We then sort this list based on final score and the bottom 50% of items are thrown out, because they are weaker matches. Then the scores of the remaining items are used as choice probabilities and the advertisement set is randomly selected and returned as output.

3.2 Website

The system also incorporates a website to allow different retailers to advertise their products through our tool. On this page, users can add and manage items that they wish to advertise on the ORCA component. They can fill in important item information that the system uses in the classification of the items when advertising.

The website acts as the interface an advertiser has with ORCA. The following is a list of the website high-level requirements that were addressed:

- Manage Profile: Advertisers are able to create and manage a profile through which all interaction with ORCA will occur. The profile contains information about the client company, a contact in the company and payment information.
- Manage Advertised Items: Advertisers are able to create a listing for each item they wish to advertise. The item listing contains metadata to classify the item. Further, advertisers are able to manage items they have previously listed and edit their attributes.
- Set Bid Prices: Advertisers input a bid price for each item. This price is the value they are willing to pay per-click when someone clicks on their advertisement.
- Track Input: The site has tools in place to trap “bad” input\(^1\) from entering our system. This serves as the first line of defense from malicious advertisers. Validations are in place to ensure that all data is representative of the item.
- Backend Capabilities: The backend supports the website through Parse [3], a cloud oriented database system. The tables are managed through the Parse’s website.
- Security: The system is secure and does not expose key information that could be used maliciously against it. Furthermore, the majority of the pages are not accessible unless the user has logged in.
- Meet Expected Performance Standards: The site performs well on all standard web browsers (Firefox, Chrome, IE, Safari, Opera). A CSS file, view.css, handles this issue.

\(^1\)Input where the item does not match the description or the item is a duplicate of a previously posted item.
Fig. 2 is an image of the sitemap. The website consists primarily of pages that can only be accessed while a user is logged in. There are other pages, such as the welcome page and the sign up page, which can be accessed by non-members. As you can see above in our sitemap, these classifications split the website into two main parts.

The website works in conjunction with the component to display items that advertisers have added. Besides the metadata that a user adds, the website also uses the ColorTag API[1] to determine a primary and secondary color for the item that has been added. In order to understand the full flow of the system, consider the use case diagram seen in Fig. 3. When a user clicks at a page that contains our embedded component, we analyze the item they are viewing and select three items from our database to display. These items are picked from a pool of all the items that advertisers have added to our database through the website. If a user clicks on one of these advertisements, then the appropriate advertiser is charged and we record the input. The user is then redirected to the item’s webpage.

4. SYSTEM IMPLEMENTATION

4.1 Algorithms

The algorithms currently run in Javascript. We have a large corpus of about 2,950 items in the database and find that the component loads rapidly. However, as the corpus grows, time performance will suffer. To solve this scaling problem the algorithms should run on a server. But the primary purpose of our research is to discover if we can effectively display complementary retail ads, so we disregarded potential scaling problems in order to focus on our target goal.

Wedding Algorithm

Our computations begin by querying the Parse database for items. The algorithm filters the queried items so that it only receives items that are the same gender and can be worn in the same outfit as the source item. It also filters the items so that blacklisted items are not returned.

Bidding Algorithm

The bidding algorithm filters the query so that items are returned by bid price in descending order and are already ranked. To implement the GSP auction, the algorithm iterates through the results and assigns each item the bid price of the item ranked below it. Since the last item has no item ranked below it, it is eliminated.

Compatibility Algorithm

The system requires the owner of the source item to provide the component with metadata about the item as input. From this metadata, the compatibility algorithm can extract the necessary information. To color match we first grab the image URL of the source item and input it to the ColorTag API, which extracts and returns the primary color of the image. The algorithm computes its complementary color by subtracting each color (red, green, and blue) of the source item from 255 – the maximum value for each color. For example, the complementary color of (30, 100, 55) would be (225, 155, 200). The algorithm then iterates through the items to compute the three-dimensional distance – the Euclidean distance – from the complementary color. To favor items that have smaller distances, the color score is computed by subtracting the distance from the maximum Euclidean distance from the complementary color. For example the maximum distance from (100, 200, 75) would be \(\sqrt{155^2 + 200^2 + 180^2}\).

To price match, the algorithm first queries the database for the items of the same gender and of the same type as the source item. For example, male jeans produce male jeans, and female sandals produce female sandals. It also filters the query so that the items are sorted by ascending price. It then adds the source item to the list so that it is still ordered by ascending price. Dividing the index of the source item in the array by the total number of items in the list computes the price percentile - or how expensive an item is within its gender and type. The items in the database already have this metric computed, so now the algorithm can iterate through the previously queried items and find the distance between each item’s percentile and the source item’s percentile. To favor items that have smaller distances, the price score is computed by subtracting the distance from the maximum Euclidean distance from the source item’s percentile.

To compute a final compatibility score, the system must normalize the color and price scores by dividing their scores...
by their maximums. It then multiplies each color score by .6 and each price score by .4 and adds them together, so that the system favors color matching slightly.

**Final Selection**

We compute a final score by normalizing both compatibility and bid price scores. We then multiply the compatibility score by .95 and the bid price score by .05 and add them together. To properly use the bid price score in a business model, we would increase its weight from the current 5%. However, because our research focuses on displaying the most compatible results, we currently heavily favor the compatibility score.

Now that we have a final score, we drop the bottom 50% to avoid poor matches. We then compute random numbers for as many ads as we want to display (usually three). These random numbers fall in the range between 0 and the sum of the remaining items’ scores. When we compute a random number, we iterate through the queried items and keep a running total of the scores. We select the item which causes the running total of the scores to exceed the random number. This method ensures that the scores of the items are used as probabilities of being selected, so that more highly scored items are more likely to be displayed.

**Color Extraction**

In order to efficiently interpret the primary and secondary colors of the items in the corpus, each picture URL is processed through the ColorTag API that returns a list of the colors in the picture in order from highest to lowest concentration. Additionally, whenever a user enters a new item to be advertised, the same process is applied to the picture URL that is provided. Further post-processing has to be done once the results of the color API call are returned, as both background colors and skin tones attribute to invalid color results. Through analysis of a number of skin tones, the system determines a list of 90 common skin tones as well as 14 background tones. We find that most shopping retail pictures are shot with backgrounds of pure white or varying shades of grey. Therefore, the algorithm first screens these colors, since the background colors tend to dominate picture concentrations. With regard to skin tones, certain items of clothing (shorts, tank tops, swimwear, sleepwear, underwear, and more) reveal more skin than others. As a result, the color results of items with these classifications are screened to remove the skin tones. In order to create a flexible screening for both the background colors and skin tones, colors that are found to be within a range of 10 for the red, green, and blue color values are declared matches.

4.2 Website

As previously discussed, the website consists of two main sections: the member section and the non-member section. Upon arriving at the website, all users are taken to the non-member section and can enter the member section by signing up or logging in. On the member side of the website, advertisers are given the following functionality:

- **Add Item**: Advertisers may add new items that will be appropriately classified and stored in our database to advertise.
- **Edit Item Attributes**: Advertisers can look at the items they have previously listed for advertisement and edit the different attributes for each of them.
- **Manage Bids**: Advertisers can manage the bids they have set for each of their items. The bid is the price an advertiser agrees to pay every time someone clicks on their advertisement.

All front-end functionality is implemented in JavaScript. We use JavaScript because it is highly flexible and very familiar, given many of our backgrounds in Java. These scripts are responsible for grabbing user inputs, setting text fields, and relaying information to and from the database. All pages have their own .js file which they use in conjunction with a databaseController that handles all interaction with the back end.

Fig. 4 shows the different tables the database has. These tables are used to manage the advertiser website and advertise through the component. The database is implemented in Parse, an online cloud based database system. We use Parse for a few reasons. First, we wanted to minimize the amount of programming languages in our code. We thought JavaScript with embedded SQL strings would become cumbersome and difficult to manage. Secondly, Parse is quick and highly portable, making it ideal for our embeddable component which needs to run quickly and on any retail clothing website.

<table>
<thead>
<tr>
<th>Name</th>
<th>Columns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>ID (PK), Username, Password</td>
<td>Table used to store information about client users. This table will be written to when signing up (‘Sign Up’) or editing profile settings (‘Edit Profile Settings’).</td>
</tr>
<tr>
<td>Items</td>
<td>ID (PK), UserID (FK), Brand, Title, Brand Type1, Brand Type2, Price, URL, Picture URL, Primary Color, Secondary Color</td>
<td>Table used to store individual item information. This table will be written to when adding items (‘Item Addition’) or when editing item settings (‘Edit Profile Settings’).</td>
</tr>
<tr>
<td>Payment</td>
<td>UserID (FK), Name, Credit Card Number, Exp Date, Country, Address, City, State, Zip, Country</td>
<td>Table used to store payment information for each user. Also contains billing address information. This table will be written to when signing up (‘Pay Offline’) or when editing profile settings (‘Edit Profile Settings’).</td>
</tr>
<tr>
<td>Types</td>
<td>ID (PK), Type</td>
<td>Table used to store the various product types. These types cannot be written to. See below for product types. Lists Level A and Level B classifications in same table.</td>
</tr>
<tr>
<td>ClassRel</td>
<td>DB, ID</td>
<td>Table used to store the link each Level B classification to its Level A classification. This table cannot be written to. See below for Level A/Level B breakdown.</td>
</tr>
</tbody>
</table>

![Figure 4: Database Tables](image)

As seen in Fig. 5, item classification is handled via the “Types” table that contains two levels of classifications for items: Level A and Level B. Level A classifications are more high level and are used for grouping similar items (pants, shirts, shoes, jackets). Level B classifications are used to be more specific (jean pants, formal slacks, athletic pants). Advertisers set the Level B classification and we handle the Level A classification on our end.

<table>
<thead>
<tr>
<th>Level A</th>
<th>Level B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pants</td>
<td>Athletic</td>
</tr>
<tr>
<td>Shirts</td>
<td>Athletic</td>
</tr>
<tr>
<td>Shoes</td>
<td>Athletic</td>
</tr>
<tr>
<td>Jackets</td>
<td>Athletic</td>
</tr>
<tr>
<td>Other</td>
<td>Other</td>
</tr>
</tbody>
</table>

![Figure 5: Item Classification](image)

The system also verifies which users can use certain tools on the website. A user profile implementation manages users
across pages in order to ensure the safety and security of the website.

The website is designed to be safe by the use of a profile management system. Upon logging in, a user’s ID and a given token are tracked. The token is used as validation every time the advertiser navigates from page to page. The ID is used in order for the system to know who is logged in. The token is masked and not visible to outside users, so that the system is secure.

5. SYSTEM PERFORMANCE

5.1 Algorithms

When evaluating our component, the most important considerations are the selection algorithms. The criteria upon which we evaluate the initial selection algorithms (bidding and compatibility) is on how similarly they select items each time, how well these items go with the source item, how well they go with each other, and ensuring that they do not conflict with the source item or amongst themselves. We must also consider the time it takes to run these algorithms and ensure that it is a reasonable amount of time (i.e. < 1 sec.).

We currently measure our bidding algorithm quantitatively by sending test sets as input and randomly selecting the ad to be displayed a large amount of times (e.g. 100,000 times). We can find the expected percentage of selections for each item by dividing each item’s weight by the sum of all the weights. The actual percentage is the number of times each item is chosen divided by the total number of test selections. These values should be about equal to ensure quantitative success.

Of course, we will need to measure our bidding algorithm qualitatively, too. We spoke with potential users about the results of our bidding algorithm. It is important to confirm that they feel their advertisements are being displayed a fair amount. We anticipate modifications of our bidding algorithm if the feedback we receive suggests certain tweaks will improve user satisfaction.

We had users evaluate how well certain clothing items complement each other. With enough feedback, we used these metrics to both evaluate the compatibility algorithm’s current effectiveness and learn about which factors are most important in determining compatibility.

We evaluated our weeding algorithm by manually populating bad items into our test sets and observing whether the weeding algorithm identifies and removes them. For example, an advertiser should not bid with the same item multiple times. Additionally, an advertiser should not lie about an item with which it is bidding. An array of different bad items should be tested, because without proper weeding, our system will be vulnerable and ineffective.

We generated a corpus of 2,950 items to be used for testing and algorithm creation. By using technology developed by Kimono Labs[2], we efficiently populated the database. To begin, we created a scraping template and connected it to the page listings for both men’s and women’s clothing and shoes. Once this template was finalized, each clothing item page was scraped for the following components: price, URL, picture URL, brand, item title, and the classification (outwear, pants, shorts, etc.). The data from Kimono Labs’ API is available in JSON (JavaScript Object Notation) format, and we created a Python script that downloaded the JSON data and then processed the data in order to generate the proper classifications for each item and to determine primary and secondary colors.

5.2 Website

Evaluating the advertiser website is neither as difficult nor as crucial as evaluating the effectiveness of our algorithms. Verifying that the website runs as intended consists mostly of ensuring that users are able to get from page to page seamlessly, can add items to the database, can edit items that are already in the database and could edit their user settings.

In order to verify all of this functionality, we designed a few user trials, both random and scripted. The scripted trials consists of predetermined steps a user takes when using our website in order to make sure that the most standard functionality is working. The random trials give the user total freedom on our website to add and edit as they pleased. In both cases, we found that our website flowed smoothly and that no consistent errors or bugs occurred.

The website is built to be able to handle large amounts of users in the network at once. Getting many users on the site simultaneously is not an easy task so we decided instead to write an automated script to add many items from the website at once. Running the script yielded positive results, with the backend being able to support a few hundred item additions at once. We do not have a set number of how many users can be on the system at once, but given the high capacity for item addition, we estimate that it is over any practical amount that we would require.

6. RESULTS

Evaluating the effectiveness of our research is difficult. Our goal is to advertise complementary items, which is a subjective quality. We cannot quantitatively evaluate how well we match items. People have varying opinions of what it means for clothing items to be complementary and whether certain matches are better than others.

Therefore, we conducted a 14 question survey via Qualtrics to measure how successfully people think we match items. Each question consists of a picture of an item and asks the participant to choose the item which would best fit in an outfit with the item in the picture. The two potential answers are also pictures. One potential answer is generated using our algorithm and the other is a randomly selected item from our corpus.

Between 89 and 96 people answered each question, creating 1,265 data points. Of those data points, the item generated by our algorithm was selected 79.6% of the time. Furthermore, the ORCA generated ad was heavily preferred over the random ad in 13 of the 14 questions. In one question, 67% chose the random ad. Participants wrote in the comments that their choice was based on the fact that swimming trunks (the item in question) should not be worn with shoes (the ORCA generated ad). We have since fixed our algorithm to only display sandals with swimwear. If that had question is thrown out, the participants selected the ORCA generated ad 82.5% of the time. These results confirm that our ads are significantly better than random ads.

We would like to test our algorithm against something better than a randomly chosen item, which would theoretically make the quality of the competing item better. However, we do not currently have access to another algorithm that generates complementary items. Nevertheless, our sur-
vey results demonstrate that our algorithms are effective at displaying good matches.

7. FUTURE WORK

Over time, the algorithm could be further improved in a number of different areas. First, we could fine-tune the algorithms based on user behavior. By adding this, the algorithm will be able to determine a number of factors that would make the results it generates more accurate and effective as advertisements. For instance, if over time it were realized that more people click advertisements for shorts when the item they are viewing is a tank top, then the algorithm should therefore display more shorts. Second, the algorithm should implement geo targeting, which would allow the algorithm to display results based on the current location of the user. This would allow the algorithm to take into account the current weather, which could allow listings to be adjusted as necessary. For instance, if a user is in a cold location, the algorithm could suggest more outerwear and sweaters.

The JavaScript embeddable component is just a single example of how the algorithm can be used, and we have envisioned a number of different potential use-cases for this algorithm that could be implemented over time. One feasible retail application could be a barcode-scanning application that a customer could use in a store in order to see other items that complementarily match the scanned item. For instance, if someone were browsing a pair of pants, they could scan the barcode of the pants, and find other items (either in-store or online) that complementarily match that pair of pants. A second possible application could be an application that allows customers to provide a certain item type and color, not necessarily a specific item, and the application would generate matches that go with the user’s input. For instance, the user could enter “teal tank top,” and the application would generate items that complement the item tank tops. This would allow users to find complements for items they might already own and takes away the association to a specific clothing brand and item.

Going forward, our website will not need to change much, if at all. Since it consists only of forms that advertisers fill out to update our databases, and since those form fields won’t change, we don’t need to worry about drastic changes to the design of the website.

8. ETHICS

Our system has a few potential ethical problems associated with it. First, it is conceivable that some items in our system are never selected by our algorithms to be advertised. The system mitigates this risk by the way it randomizes the final selection. After weeding and scoring the potential ad items, the system randomly picks items from the top 50% of results. This randomization should theoretically enable all items in the database to be chosen at one time or another. To further ensure that every item is displayed, the system could track how many times items are selected. If it found that certain items are never chosen, it could make modifications to the algorithms to boost these items’ chances of being selected. We could add a portion of an item’s score to be based on either how many times an item has been displayed or the last time an item has been displayed. Tweaking our algorithms in this way could increase the probability that items that are never advertised are chosen in the future.

Another ethical issue we face is the balance between the compatibility score and the bidding score in assigning probabilities to the items. The system is based on advertising complementary items, which we implement entirely in our compatibility algorithm. The bidding algorithm is simply a way to drive more revenue for our business and for our customers. While building out a successful business is important, we do not want to do so at the expense of the quality of the advertisements. Our partners and customers would be unhappy to discover that the system is not advertising truly complementary items. To avoid this issue the system could track click-through rates of the component as well as total revenue received for different balances of the compatibility and bidding scores. We strive to find the perfect balance where we are displaying quality items and also driving maximum revenue. We could constantly monitor these statistics to ensure that the quality of our ads never suffer as a result of trying to drive revenue.

9. CONCLUSIONS

Current contextual advertising solutions for retail products are redundant. However, online shoppers often seek more complete outfits. Therefore, ORCA displays complementary rather than redundant ads based on the item currently being viewed by the shopper. Ads that we display do not compete with currently viewed item, which enables us to build a platform through which retailers advertise on each others’ websites.

Our results indicate that our ads are effective at generating complementary products. Learning from user behavior over time and tweaking our algorithms will produce even higher quality ads. Our results prove that it is possible to produce complementary retail items, which can be repurposed to enable many other exciting applications.

10. REFERENCES


APPENDIX

A. CODE EXAMPLES

Figure 6: Add Item Code

```html
On page load (from profile.html)
- Populate relevant fields that can be completed from profile information

On "Save" click
- If all required fields are NOT filled
  - Prompt user ("Please fill in required fields")
- Else
  - Create a new entry in the "Items" table
  - For each attribute
    - Set corresponding column for that item entry
  - Set any columns that are set by the system (i.e. timestamp)
  - Redirect the user to profile.html

On "Cancel" click
- Do not add any item to DB
- Redirect the user to profile.html
```

Figure 7: Edit Item Attributes Code

```html
On page load (from listed.html OR profile.html)
- Grab relevant item info into incoming "editItem" object
- Use the item into to load all attributes into proper fields
- If "reactivateFlag" on "editItem" object is true
  - Set page mode to REACTIVE
- Else
  - Set page mode to ACTIVE

On "Save" click
- For each attribute
  - Grab the current item attribute from DB
  - Grab the corresponding attribute from the text field
  - If the two fields are different
    - Update the field in the DB
    - If page mode is REACTIVE
      - Set "Active" column for entry in "Items" table to true
      - Redirect the user to manage.html
    - If page mode is ACTIVE
      - Do not update any of the item attributes in DB
      - Redirect the user to manage.html

On "Cancel" click
- If page mode is REACTIVE
  - Prompt user with window saying "Do you want to abandon reactivation of this item?"
- If "Yes" click
  - Do not update any of the item attributes in DB
  - Redirect the user to manage.html
- If "No" click
  - Stay on page
  - If page mode is ACTIVE
    - Do not update any of the item attributes in DB
    - Redirect the user to manage.html
```

Figure 8: Manage Bids Code

```html
On page load (from manage.html)
- Grab user profile info
- Use profile info to load a user’s items
- For each item
  - Grab item name, item bid price
  - Access ‘bucket’ of similar items (from all users) to this item
  - Get average bid price of items in this bucket
  - Make new table entry with name, price, and similar avg. price
- If item "Active", load "Edit" button

On "Edit" click
- Hide "Edit" button
- Show "Save" button
- Set "editFlag" so the site knows a user is in "Edit" mode
- For the specific item, change bid price field from label to textbox

On "Save" click
- Grab new bid price from textbox
- Update table entry for item with new bid price
- Run algorithm to get average price for similar items (from page load)

On "Go Back" click
- If "editFlag" is true
  - Prompt user with warning message that unsaved changes will be lost;
    "Continue" button and "Cancel" button will be in window
  - If "Continue" clicked
    - Redirect the user to manage.html
  - If "Cancel" clicked
    - Close window, Do nothing.
- Else
  - Redirect the user to manage.html
```