Building a Recommender System using Community Level Social Filtering

Alexandra Balahur and Andrés Montoyo

Department of Software and Computing Systems
University of Alicante, Ap. de Correos 99, Alicante, Spain

Abstract. Finding the "perfect" product among the dozens of products available on the market is a difficult task for any person. One has to balance between personal needs and tastes, financial limitations, latest trends and social assessment of products. On the other hand, designing the "perfect" product for a given category of users is a difficult task for any company, involving extensive market studies and complex analysis. This paper presents a method to gather the attributes that make up the "perfect" product within a given category and for a specified community. The system built employing this method can be useful for two purposes: firstly, it can recommend products to a user based on the similarity of the feature attributes that most users in his/her community see as important and positive for the product type and the products the user has to opt from. Secondly, it can be used as a practical feedback for companies as to what is valued and how, for a product, within a certain community. For the moment, we will consider the community level as being the country, and thus we will apply and compare the method proposed for English and Spanish. For each product class, we first automatically extract general features (characteristics describing any product, such as price, size, and design), for each product we then extract specific features (as picture resolution in the case of a digital camera) and feature attributes (adjectives grading the characteristics, as modern or faddy for design). Further on, we use "social filtering"[1] to automatically assign a polarity (positive or negative) to each of the feature attributes, by using a corpus of "pros and cons"-style customer reviews. Additional feature attributes are classified depending on the previously assigned polarities using Support Vector Machines Sequential Minimal Optimization [1] machine learning with the Normalized Google Distance [2]. Finally, recommendations are made by computing the cosine similarity between the vector representing the "perfect" product and the vectors corresponding to products a user could choose from.

1 Introduction

Finding the opinions different customers expressed about a product on the web is a highly useful task for two reasons. The first one addresses the potential buyers of that product, which can thus gather useful information from the users of the product, without being biased by commercial interests of one store or another. Secondly, opinions over

[1] http://www.techweb.com/encyclopedia/defineterm.jhtml?term=social+filtering

products serve producer companies as feedback regarding how the consumer market views their product, as well as for technological vigilance, regarding the products marketed by other companies and the opinions the users have on them. However, a product with the same features can be viewed as appropriate/ high-quality within a community and as inappropriate/ low-quality within another. Different characteristics of products are perceived differently, depending on the category of users, the country and the general social acceptance that is given to a product. Whenever a company wishes to market a product, it first studies the characteristics that the users the product is intended for see as important, what they value in a product and how. Therefore, a static classification of product feature attributes (such as "small" in size for a car being "negative" or "positive") used in classifying the user opinions of products can not be viewed as correct. In order to address this issue, we propose a method to automatically discover the feature attributes that are viewed as positive and negative within a given community and consequently classify products according to these discovered features. In this manner we wish to avoid product features misclassifications due to overgeneralization and offer a practical and easy method to search for products that are appropriate for different user communities. In our approach, we use the "word-of-mouth" on the web [3], that is, the user reviews of different products, in two steps. Firstly, we automatically obtain a classification of what users within a specified community perceive as positive and negative in a product, using the "pros and cons"-style reviews of the given product. This classification is used on the one hand to build a vector model of the "perfect" product and on the other hand as basis for the SMO SVM learning using the NGD scores in the feature classification and summarization system. Secondly, we employ a system for product feature summarization to extract for the given products the vectors containing the values for the feature attributes found in the reviews. Finally, we compute the similarity between each of the vectors of the products and the "perfect" product within the category and recommend the top matching products.

2 Related Work

The approach we use for the customer review feature summarization system is similar to the feature-driven opinion summarization paradigm, whose theoretical background can be found in [3]. Systems employing this paradigm are described in [4] or [5]. However, our approach is called "feature-driven opinion summarization", and differs from the feature-based systems in that our primary goal is to identify all features of a product and find examples of positive and negative attributes describing these features and consequently extract those from customer review texts. In this manner, we can discover implicit features in user reviews (for example, if the user mentions the word "small" when describing the camera, we can map the small as describing the "size" feature). Moreover, in extracting first the positive and negative examples of feature attributes from reviews pertaining to the same community as the reviews classified, we ensure the semantic coherence of interpretations we give to the classified reviews. The method we propose is robust and customer-review independent, but at the same time it captures the semantics given to feature attributes at the community level.

3 Determining Product Features

In the approach proposed, we concentrated on two main problems. The first one was that of discovering the features that will be quantified and the second is determining the feature attributes that describe the discovered features. As previously noticed in [3], features are implicit or explicit. To this respect, apart from a general class of features (and their corresponding attributes), that are applicable to all products, we propose a method to discover product specific features and feature attributes using knowledge from WordNet and ConceptNet, completed by knowledge discovered by applying a series of extraction patterns.

There are a series of features that are product independent and that are important to any prospective buyer. We consider these as forming a core of product features. For each of these concepts, we retrieve from WordNet[6] the synonyms which have the same Relevant Domain [7], the hyponyms of the concepts and their synonyms and attributes, respectively. An example of term from the core of product features is "price", with the synonyms "cost", "value", hyponyms "asking price" and "selling price", having as feature attributes "high", "low", "expensive" and "cheap". The other product independent features are "warranty", "size", "weight", "design", "appearance" and "quality".

Once the product category has been identified, we use WordNet to extract the product specific features and feature attributes. We should notice that, contrary to the observation made in [5], once we establish the product, its corresponding term in WordNet is sense disambiguated and thus the obtained meronyms, synonyms and corresponding attributes are no longer ambiguous. Moreover, the terms obtained in this manner, should they appear in customer reviews, have the meant meaning. We accomplish this in the following steps:

1. For the term defining the product category, we search its synonyms in WordNet[6].
2. We eliminate the synonyms that do not have the same top relevant domain [7] as the term defining the product category
3. For the term defining the product, as well as each for each of the remaining synonyms, we obtain their meronyms from in WordNet, which constitute the parts forming the product.
4. Since WordNet does not contain much detail on the components of most of new technological products, we use ConceptNet[8] to complete the process of determining the specific product features. We explain the manner in which we use ConceptNet in the following section.

The final step consists in finding the attributes of the features discovered by applying the *"has attributes"* relation in WordNet to each of the nouns representing product features. In the case of nouns which have no term associated by the "has attribute" relation, we add as attribute features the concepts found in ConceptNet under the *"OUT"* relations *"PropertyOf"* and *"CapableOf"*. In case the concepts added are adjectives, we further add their synonyms and opposites from WordNet. As result we have for example, in the case of "photo", the parts "raster" and "pixel" with the attributes "blurry", "clarity", "sharp". In order to obtain additional features for the product in question, we add the concepts that are related to the term representing the concept with terms related in ConceptNet by the *"OUT"* relations *"UsedFor"* and *"CapableOf"* and the *"IN"* relations

“PartOf” and *“UsedFor”*. For example, for the product “camera”, the *“OUT” “UsedFor”* and *“CapableOf”* relations that will added are “take picture”, “take photograph”, “photography”, “create image”, “record image” and for the *“IN” “PartOf”* and *“UsedFor”* relations “shutter”, “viewfinder”, “flash”, “tripod”. We employ EuroWordNet[2] and map the features and feature attributes, both from the main core of words, as well as the product specific ones that were previously discovered for English, independent of the sense number, taking into account only the preservation of the relevant domain. The majority of product features we have identified so far are one-word parts constituting products. However, there remains a class of undiscovered features that are indirectly related to the product. These are the features of the product constituting parts, such as battery life, picture resolution, auto mode. We extract these overlooked product features by determining bigrams made up of target words constituting features and other words in a corpus of customer reviews. In the case of digital cameras, for example, we considered a corpus of 200 customer reviews on which we ran Pedersens Ngram Statistics Package[3] to determine target co-occurrences of the features identified so far. As measure for term association, we use the Pointwise Mutual Information score. In this manner, we discover bigram features such as “battery life”, “mode settings” and “screen resolution”.

4 Polarity of Feature Attributes with Social Filtering

The concept of social filtering, also called collaborative filtering, denotes a series of techniques for identifying information a given user might be interested in. The basic principle of social filtering is developing a rating system for matching incoming material, depending on certain defined criteria. We will employ this concept in creating the prototype of the “perfect” product, based on the criteria extracted from “pros and cons”-style customer reviews. Further on, we will be able to filter any product pertaining to the category depending on the similarity score it has with respect to the constructed “perfect” prototype. Firstly, we create a corpus of “pros and cons”-style reviews for each of the procut categories we are interested in. For English, sites that contain such types of reviews are “newegg.com” and “eopinions.com”, which are American, or “shopping.com”, on which the regional site can be chosen also for European countries. For Spanish, sites containing reviews in the form of pros and cons (“a favor” and “en contra” or “ventaja” and “desventaja”) are “quesabesde.com” or “ciao.es”. Further on, we prepare the obtained corpus by eliminating the stopwords and performing lemmatization. We split each of the paragraphs in phrases, considering as separators the commas and fullstops. Next, for each feature discovered in the previous section, we extract from the phrase it appears in the word preceeding and succeeding it. In case of compound features (such as “battery life”), the process is identical. If the phrase is contained in a “pro” section, then the extracted words are classified as positive feature attributes. In the contrary case, the word is classified as negative. Extracted pairs are removed from the corpus. Finally, we verify if the corpus contains implicit features by identifying the feature attributes that remained in the corpus. In the case such feature attributes

[2] www.illc.uva.nl/EuroWordNet/
[3] www.d.umn.edu/ tpederse/nsp.html

are found, their polarity is positive if they are found within the pros section and negative if they are found in the cons section. The feature attributes, together with their assigned polarity, are added to the list of feature attributes that belong to the same feature. For example: `Pros: Beautiful pictures, ease of use, high quality, 52mm lens. Cons: high price, a bit big and bulky.`

Features encountered in text: picture, use, quality, lens, price. Feature attributes extracted: (positive): beautiful (picture), easy (use), high (quality), 52mm (lens); (negative): high (price). Feature attributes remaining: big, bulky, which are both negative and correspond, according to the feature categorization made in section 4, to the "size" feature.

5 Customer Review Feature-driven Summarization

When user asks the system to recommend a product from a given product category in Spanish or English, the system will select the standard against which to compare the discovered products. Since we have set the goal of comparing reviews in Spanish and English-speaking communities, we will process product reviews in both languages, using the same system, but different resources.

In order to solve the anaphoric references on the product features and feature attributes, we employ two anaphora resolution tools - JavaRAP [4] for English and SUPAR [9] for Spanish. Using these tools, we replace the anaphoric references with their corresponding referents and obtain a text in which the terms constituting product features could be found.

Further on, we split the text of the customer review into sentences and identify the named entities in the text. We use LingPipe [5] to split the customer reviews in English into sentences and identify the named entities referring to products of the same category as the product queried. In this manner, we can be sure that we identify sentences referring to the product queried, even the reference is done by making use of the name of another product. For example, in the text "For a little less, I could have bought the Nikon Coolpix, but it is worth the extra money.", anaphora resolution replaces <it>with <Nikon Coolpix>and this step will replace it with <camera>. We employ FreeLing [6] in order to split the customer reviews in Spanish into sentences and identify the named entities referring to products of the same category as the product queried.

Having completed the feature and feature attributes identification phase, we proceed to extracting for further processing only the sentences that contain the terms referring to the product, product features or feature attributes. Each of the sentences that are filtered by the previous step are parsed in order to obtain the sentence structure and component dependencies. In order to accomplish this, we use Minipar [10] for English and FreeLing for Spanish. This step is necessary in order to be able to extract the values of the features mentioned based on the dependency between the attributes identified and the feature they determine.

[4] http://www.comp.nus.edu.sg/ qiul/NLPTools/JavaRAP.html
[5] http://www.alias-i.com/lingpipe/
[6] http://garraf.epsevg.upc.es/freeling/

Further on, we extract features and feature attributes from each of the identified sentences, using the following rules:

1. We introduce the following categories of context polarity shifters[11], in which we split the modifiers and modal operators in two categories - positive and negative: - negation: no, not, never etc. - modifiers: positive (extremely, very, totally etc.) and negative (hardly, less, possibly etc.) - modal operators: positive (must, has) and negative (if, would, could etc.)
2. For each identified feature that is found in a sentence, we search for a corresponding feature attribute that determines it. Further on, we search to see if the feature attribute is determined by any of the defined modifiers. We consider a variable we name valueOfModifier, with a default value of -1, that will account for the existence of a positive or negative modifier of the feature attribute. In the affirmative case, we assign a value of 1 if the modifier is positive and a value of 0 if the modifier is negative. If no modifier exists, we consider the default value of the variable. We extract triplets of the form (feature, attributeFeature, valueOfModifier). In order to accomplish this, we use the syntactic dependency structure of the phrase, we determine all attribute features that determine the given feature. For the sentence "The hazy image displayed on the screen disappointed me.", the tuples extracted are (image, hazy, -1) and (image, disappointed, -1).
3. If a feature attribute is found without determining a feature, we consider it to implicitly evoke the feature that it is associated with in the feature collection previously built for the product. "The camera is small and sleek." becomes (camera, small, -1) and (camera, sleek, -1), which is then transformed by assigning the value "small" to the "size" feature and the value "sleek" to the "design" feature.

6 Assignment of Polarity to Feature Attributes

In order to assign polarity to each of the identified feature attributes of a product, we employ SMO SVM machine learning and the Normalized Google Distance (NGD). The main advantage in using this type of polarity assignment is that NGD is language independent and offers a measure of semantic similarity taking into account the meaning given to words in all texts indexed by Google from the world wide web. The set of anchors contains the terms $\{$*featureName, happy, unsatisfied, nice, small, buy*$\}$, that have possible connection to all possible classes of products. Further on, we build the classes of positive and negative examples for each of the feature attributes considered. From the list of classified feature attributes in section 4, we consider all positive and negative terms associated to the considered attribute features. We then complete the lists of positive and negative terms with their WordNet synonyms. Since the number of positive and negative examples must be equal, we will consider from each of the categories a number of elements equal to the size of the smallest set among the two, with a size of at least 10 and less or equal with 20. We give as example the classification of the feature attribute "tiny", for the "size" feature. The set of positive feature attributes considered contains 15 terms such as "big", "broad", "bulky", "massive", "volumious", "large-scale" etc. and the set of negative feature attributes considered is composed as opposed examples, such as "small", "petite", "pocket-sized", "little" etc. We use the anchor words to convert each of the 30 training words to 6-dimensional training vectors defined as v(j,i) = NGD(wi,aj), where aj with j ranging from 1 to 6 are the anchors and wi, with i from 1 to 30 are the words from the positive and negative categories. After obtaining the total 180 values for the vectors, we use SMO SVM to learn to distinguish the product specific nuances. For each of the new feature attributes we wish to classify, we calculate a new value of the vector vNew(j,word)=NGD(word, aj), with j ranging from 1 to 6 and classify it using the same anchors and trained SVM model. In the example considered, we had the following results (we specify between brackets the word to which the scores refer to): (small)1.52,1.87,0.82,1.75,1.92,1.93,positive; (little)1.44,1.84,0.80,1.64,2.11,1.85,positive; (big)2.27,1.19,0.86,1.55,1.16,1.77, negative; (bulky)1.33,1.17,0.92,1.13,1.12,1.16,negative. The vector corresponding to the "tiny" attribute feature is: (tiny)1.51,1.41, 0.82,1.32,1.60,1.36. This vector was classified by SVM as positive, using the training set specified above. The precision value in the classifications we made was beween 0.72 and 0.80, with a kappa value above 0.45.

7 Summarization of Feature Polarity in Reviews

For each of the features identified, we compute its polarity depending on the polarity of the feature attribute that it is determined by and the polarity of the context modifier the feature attribute is determined by, in case such a modifier exists. Finally, we statistically summarize the polarity of the feature attributes, as ratio between the number of positive quantifications and the total number of quantifications made in the considered reviews to that specific feature and as ratio between the number of negative quantifications and the total number of quantifications made in all processed reviews. This process is repeated for all possible choices of products within the sought category.The formulas can be summarized in

F_pos(i)=#pos_feature_attributes(i)/#feature_attributes(i);
F_neg(i)=#neg_feature_attributes(i)/#feature_attributes(i).

8 Product Recommendation Method

In order to recommend a product for purchase, we present a method to compute the similarity between a product (whose features are summarized using the customer review summarization system) and what is seen as the "perfect" product in the category. For each product category, we consider a list containing the general and product-specific features. The perfect product within that category can thus be represented as a vector whose indices correspond to the list of features and whose values are all 1, signifying that all features are positive. At this point, it is interesting to note that the semantics of "positive" and "negative" for the product category are given by the feature attributes we extracted for each of the categories (positive for size thus includes "small", "tiny", "pocket-fit" etc.). In order to find recommendable products, we use the customer review summarization system presented in section 7 for each product model and its corresponding collection of reviews. In this manner, we build vectors corresponding to the product models, whose indices will be the same as the ones of the "perfect" product, and whose corresponding values will be the percentage in which the feature is classified as positive by the summarization system. Finally, we compute the similarity between the each of the obtained vectors and the vector corresponding to the “perfect” product using the cosine similarity measure[7].We recommend the top 5 matching products. In order to better understand the process of recommendation, we will consider an example and suppose the user would like to buy a 4 Megapixel camera. There are around 250 available models on the market and for each model one can read an average of 10 customer reviews. Instead of having to read 2500 reviews, employing the presented system, being given the 5 best products, the user will only have to browse through 50 reviews, in case (s)he is not confident in the system classification; when the user is confident, (s)he has to read none.

The list of features for a 4 Megapixel camera is: (price, warranty, size, design, appearance, weight, quality, lens, viewfinder, optical zoom, digital zoom, focus, image resolution, video resolution, memory, flash, battery, battery life, LCD size, LCD resolution, accessories)

The vector associated to the “perfect” 4 Megapixel camera will have as indices the features in the above list and all corresponding values 1: v_perf(price)=1; v_perf(warranty) =1 and so on, in the order given by the list of features. After applying the customer review summarization system on other 4 Megapixel cameras, we obtain among others the vectors v1 and v2, corresponding to Camera1_4MP and Camera2_4MP. The values of v1 are: (0.7,0.5,0.6,0.2,0.3,0.6,0.5, 0.5,0.7,0.8,0.7,0.8, 0.4,0.3,0.3,0.7,0.6,0.3,0.8,0.4,0.4) The values of v2 are: (0.8,1, 0.7,0.2,0.2,0.5, 0.4,0.4,0.8,0.8,0.8,0.8,0.7,0.7,0.3,0.8,0.6, 0.7,0.5,0.3,0.6) Calculating the cosine similarity between v1 and v_perf and v2 and v_perf, respectively, we obtain 0.945 and 0.937. Therefore, we conclude that Camera1_4MP is better than Camera2_4MP, because it is more similar to the “perfect” 4 Megapixel camera model.

9 Evaluation and Discussion

We performed a thorough evaluation the system for customer review summarization and an informal evaluation of the recommender. For the first, we annotated a corpus of 50 customer reviews for each language, collected from the sites containing the “pros and cons”- style reviews: “newgegg.com”, “eopinions.com”, “shopping.com”, “quesabesde.com” and “ciao.es”. The corpus was annotated at the level of feature attributes, by the following scheme: <attribute>[name of attribute] <feature>[feature it determines]</feature><value>[positive / negative]</value></attribute>.

It is difficult to evaluate the performance of such a system, since we must take into consideration both the accuracy in extracting the features that reviews comment on, as well as the correct assignation of identified feature attributes to the positive or negative category. Therefore, we introduced three formulas for computing the system performance: Accuracy (Eq 1), Feature Identification Precision (FIP) (Eq. 2) and Feature Identification Recall (FIR) (Eq. 3).

$$A = \frac{\sum_i^n \left(\frac{\#pos\_id\_features(i)}{\#pos\_features(i)} + \frac{\#neg\_id\_features(i)}{\#neg\_features(i)}\right)}{2n} \qquad (1)$$

[7] www.dcs.shef.ac.uk/ sam/stringmetrics.html#cosine

$$FIP = \frac{\#(correctly\_identified\_features \cap \ identified\_features)}{\#identified\_features} \quad (2)$$

$$FIR = \frac{\#(identified\_features \cap \ correctly\_identified\_features)}{\#correctly\_identified\_features} \quad (3)$$

The results obtained are summarized in Table 1. We show the scores for each of the two languages considered separately and the combined score when using both systems for assigning polarity to feature attributes of a product. In the last column, we present a baseline, calculated as average of using the same formulas, but taking into consideration, for each feature, only the feature attributes we considered as training examples for our method. We can notice how the use of NGD helped the system acquire significant new knowledge about the polarity of feature attributes.

Table 1. System results.

	English	Spanish	Combined	Baseline English	Baseline Spanish
A	0.82	0.80	0.81	0.21	0.19
FIP	0.80	0.78	0.79	0.20	0.20
FIR	0.79	0.79	0.79	0.40	0.40

In respect to the tools and resources we used for the system, the ones performing sentence chunking and parsing are vital to applying the method described and they cannot be removed from the system. As far as the anaphora resolution component is concerned, in the considered English corpus, an average of 61.3 of the sentences contained a reference to the product in question or its features. In the case of the Spanish corpus, the average was 42.1. Among these, the error rate of anaphora resolution for JavaRAP was 38 percent. It was however interesting to notice the fact that in the case of wrong classification, the output was not negatively influenced in 20.4 percent of the cases, when the resolution was done to a term identifying the name of another product, since by unification the product name was replaced with the product category name. In the case of anaphora resolution for Spanish, the error rate was 21 percent. An interesting observation we must make, however, that the use of these natural language processing tools is highly dependent both on the length of the reviews, as well as the collecting site, since the use of rather informal styles of writing results in low success rates in anaphora resolution.

Regarding the evaluation of the recommender system, we asked 10 persons to rank the 5 recommendations they were given by the system. All found the product they were interested to buy within the 5 suggested alternatives.

10 Conclusions and Future Work

In this paper we presented a method to extract, for a given product, the features and feature attributes that could be commented upon in a customer review. Moreover, we presented a method to extract and assign polarity to feature attributes according to "pros and cons"-style reviews. We showed the manner in which a system that statistically summarizes the polarity of feature attributes in review texts in English and Spanish can be built and finally the method to compute similarity between the product considered as "perfect" and products of the same category that are available for purchase. The main advantage obtained by using this method is that one is able to extract and correctly classify the polarity of feature attributes, in a product and community dependent manner. Not lastly, we employ a measure of word similarity that is in itself based on the "word-of-mouth" on the web. SVM learning and clasification is dependent on the NGD scores obtained with a set of anchors and terms that are previously established depending on the classification of the same users as those whose reviews are used to classify products. Future work includes a finer-grained classification depending on smaller communities and a more thorough evaluation of the differences in using a general classification system as opposed to a community dependent one. Related to the technical side of the implementation, future work includes the use of alternative natural language processing tools, whose improved performance could also help achieve better results in the feature-driven opinion summarization system.

References

1. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines. Microsoft Research Technical Report MSR-TR-98-14 (1998)
2. Cilibrasi, D., Vitanyi, P.: Automatic Meaning Discovery Using Google. IEEE Journal of Transactions on Knowledge and Data Engineering (2006)
3. Liu, B.: Web Data Mining. Exploring Hyperlinks, Contents and Usage Data. First edn. Springer (2007)
4. Hu, M., Liu, B.: Mining Opinion Features in Customer Reviews. In: Proceedings of Nineteenth National Conference on Artificial Intellgience AAAI-2004. (2004)
5. Dave, K., Lawrence, S., Pennock, D.: Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In: Proceedings of WWW-03. (2003)
6. Fellbaum(ed.), C.: WordNet: An Electronic Lexical Database. First edn. MIT Press (1999)
7. Vázquez, S., Montoyo, A., Rigau, G.: Using relevant domains resource for word sense disambiguation. In: Proceedings of the ICAI 2004. (2004)
8. Liu, H., Singh, P.: ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal **22** (2004)
9. Ferrández, A., Palomar, M., Moreno, L.: An Empirical Approach to Spanish Anaphora Resolution. Machine Translation. Special Issue on Anaphora Resolution In Machine Translation. Special Issue on Anaphora Resolution In Machine Translation (1999) 191–216
10. Lin, D.: Dependency-based Evaluation of MINIPAR. In: In Workshop on the Evaluation of Parsing Systems. (1998)
11. Polanyi, L., Zaenen, A.: Exploring attitude and affect in text: Theories and applications. Technical Report SS-04-07 (2004)