Production-Capable Extraction of Pros and Cons from Product Reviews

Todd Sullivan
Department of Computer Science
Stanford University
todd.sullivan@cs.stanford.edu

1 Introduction

Product reviews are now a key part of the purchasing process for many consumers. With the widespread availability of the internet, consumers are increasingly using product reviews to guide their purchasing decisions. Aside from checking reviews at home before purchasing online or going to a brick-and-mortar store, smartphones such as the iPhone and Android-powered devices are enabling consumers to lookup product reviews on the spot while in a store.

While reading reviews can certainly be helpful, few consumers have time to sift through all of the reviews available for a given product. Aggregate information is far more useful in this context than plain text reviews. While the average rating is somewhat useful, pros and cons for each review and the aggregate counts of each pro and con on the product level would provide consumers with an instant snapshot of the product's performance and value.

In this paper, we describe a system for extracting pros and cons from review text that achieves high precision for extracted tags and successfully extracts tags from a reasonable percentage of reviews. Our system uses part of speech tags and dependency trees, as well as information such as the companies and products in the category, which can be automatically collected from Amazon and other websites. In the next section we briefly describe our previous work in this area and other related works. Section 3 describes our problem definition and goals. Our data sources and the pre-processing of these sources are covered in Section 4. We then move to annotating review text with entities, relationships between words, and other important information in Section 5. Section 6 details the core tag extraction algorithm with its many heuristics for eliminating erroneous extractions. We evaluate our system's performance in several product categories in Section 7, and finish with a brief discussion of potential future additions to the system.

2 Background

In our previous study [1], we explored several methods for automatically tagging GPS device reviews with pros and cons. Our product review data was provided by PowerReviews, which collects product reviews from consumers and explicitly asks for pros and cons. The user is given a selection of suggested pros and cons for the category and also has the option of typing in his or her own free-text tags. Our previous study focused on predicting whether the reviewer tagged the review with a particular pro or con based on the review's rating, reviewer's location, date/time of review submission, and the review's comment text (i.e. we built classifiers for each tag). While our techniques showed promise, the dataset was ultimately inadequate for the study because PowerReviews' collection method caused reviewers to write about different topics in the comment text than the pros and cons that they had tagged before writing the comment.

There have been several somewhat successful studies in the realm of predicting ratings, or sentiment, expressed in review [2, 3]. Other work in extracting pros and cons found that pros and cons occur in both factual and opinion sentences [4].

3 Problem Definition

Our system is different from previous studies in that we use dependency trees to extract specific pros and cons from reviews. We do not use any machine learning techniques to predict if sentences or phrases are pros or cons. Instead, we use the structure of the sentences, along with a list of tags to be extracted and information about the product category such as product names, company names, generic terms for products in the category, etc., to extract pros and cons from reviews.

As in our previous study, our data is provided by PowerReviews. The data consists of reviews from the Automotive GPS and Dress Shoes categories. For each category, we have a list of companies and products in the category. Each review is matched to a product and contains a title and comment text. The dataset contains reviews collected via the PowerReviews system (which we will reference as “PR Reviews”) and reviews collected via other systems (referenced as “Imported Reviews”). The imported reviews come from multiple sources such as Overstock.com, Best Buy, and websites that use Bazaarvoice’s review collection system. The collection systems for imported reviews do not ask the user for pros and cons, and as a result tend to have longer comments with more pro/con information to extract.

The PR reviews also have pros and cons as free-text tags. We use these tags as a jumpstart for defining the tags that the system will extract from each category. This process is described in detail in Section 4.4.

The primary goal of our system is to extract pros and cons from a reasonable percentage of reviews in any given category (at least 50% of reviews with extracted tags) with an accuracy of at least 90% on extracted tags. Hence, we deem high precision on extracted tags to be much more important than high recall.

4 Knowledge Sources

In this section we describe our knowledge sources and the preprocessing of these sources. We use the Stanford Parser [5] on product names, pro/con tags, and product reviews to obtain part of speech tags, phrase structure trees, and dependency trees [6]. We also use the Stanford Parser’s stemming functionality, which removes inflections but does not remove derivational morphology.

4.1 Companies and Product Names

Our company data consists of a company identifier along with a list of synonyms for the company. The synonyms are the names that each merchant using the PowerReviews collection system uses to refer to the company. For example, if Store1.com and Store2.com both use PowerReviews’ collection system and Store1.com refers to the company Garmin as “Garmin” while Store2.com uses “Garmin International”, then both of these names exist in the company’s synonym list. We use this data without any pre-processing to map words and phrases in the review text to their respective company entities.

Similarly, our product data consists of a product identifier along with a list of product names for the company where each product name is the name used by some merchant in the PowerReviews system. The product data requires light preprocessing before being used in the system. Particularly, many products have names that include additional information such as screen size or color. These additional bits of information often occur after a comma, colon, dash, parenthesis, or a preposition such as “for”. Thus for each product name that contains one of the previously mentioned items, we keep only the words occurring before the item.

Realizing that most consumers do not refer to products by their full name, we extract all possible n-grams from each product name and use them as possible references to the product. We use all n-grams with a length greater than one, and only use unigrams that do not occur in one of the following cases:

- a company name
- one or two digit number
- punctuation
- the word occurs in more than one WordNet synset among noun, verb, adverb, adjective, and adjective satellite synset types

This method eliminates many problem words that occur in multiple product names such as “navigation”, “system”, and “device”. It can also be used to find unigrams that are generic terms for products in the category. The method could also be taken further to identify brands such as “Streetpilot” in the GPS category, although we do not pursue this route.

4.2 Word Synonyms

Word synonyms are simply lists of words that for our intents and purposes have the same meaning. Word synonym lists also include common misspellings for important terms. Table 4.2 lists our system’s word synonyms and a sampling of their lists. Synonyms can be single exact-match words or more complex regular expressions. We opted for hand-written lists over using WordNet or similar automatic synonym dictionaries because we aim for high accuracy and we wish to include many words in the lists that are not necessarily exact synonyms with each other. Words that are in the “unimportant adverbs” list are treated as if they do exist in the review text.

4.3 Generic Entity Resolvers

As part of the review annotation process discussed in Section 5, we attempt to label words by their entity types. To do this we use simple word lists for each entity type. Table 4.3 details the generic entity resolvers for the Dress Shoes category. The current and competing indicators are used to determine, for example, if a match for a generic product noun is referring to the current product or a competing product.

Table 4.2: Word Synonym Lists

Meaning	Example Words
good	good, great, wonderful, dandy, delicious, …
bad	bad, horrible, woeful, poor, torturous, unpleasant, …
comfortable	comfortable, comfy, …
operate	use, operate, operation
easy	easy, elementary, effortless, …
difficult	difficult, impossible, baffling, …
fast	fast, speedy, quick, …
slow	slow, sluggish, …
small	small, tiny, miniature, …
large	large, huge, gigantic, …
versatile	various misspellings of versatile
unimportant adverbs	very, rather, quite, still, already, normally, honestly, highly, …

Table 4.3: Generic Entity Resolvers for the Shoes Category

Entity Type	Example Words
Reviewer Pronouns	I, me, my, …
Reader Pronouns	you, your, …
Other Person Pronouns	he, she, him, …
Other Person Nouns	friend, relative, husband, …
It Pronouns	it, itself, its, thing
Generic Product Nouns	shoe, slipper, loafer, …
Generic Company Nouns	company, brand, marque
Competing Indicators	other, another, older, …
Current Indicators	this, the

4.4 Tags

We use the pro/con tags in PR reviews as suggestions for tags to extract for each category. For a given category, we count the frequency of each tag, sort the tags by frequency, and use the frequent tags as hints for the tags to extract. We do not, for example, attempt to extract all tags that occur at least ten times in the category because many users (and the PowerReviews system) use tags such as “Heel”, which means “good heel” or “sturdy heel” when in the pro section and “bad heel” or “wobbly heel” when in the con section. The word “heel” by itself is obviously not enough information for our system to do anything with.

Thus, for each category, we define a list of pro and con tags to extract. Each tag has a list of synonyms and optionally a link to an opposite tag. The opposite tag is used in the case that the current tag is extracted but it is determined to be negated.

Each tag synonym can be thought of as another way to state the tag, but it is ultimately a list of words that together define a class of phrases that potentially map to the tag. The erroneous phrases that are mapped by the list of words are eliminated by the heuristics described in the tag extraction process of Section 6. The words in each tag synonym are processed with the Word Synonyms to, for example, convert occurrences of “great” and “awesome” to the word “good”.

5 Review Summarization and Annotation

The review summarization and annotation phase prepares each sentence in each review for tag extraction. In order, we chunk nouns, attach modifiers (adverbs, adjectives, negations, and quantities) to words, find subjects/objects, resolve entities, and connect each word in the sentence to an entity.

5.1 Noun Chunking

We chunk all nouns that are identified as compound nouns in the sentence’s dependency tree. We also include numbers in the chunk if they occur immediately to the right of any proper noun. This step is fairly standard procedure.

5.2 Attaching Modifiers

We attach all adverbs, adjectives, negations, and quantities to the words that they are affecting. The majority of these modifiers are identified simply by looking at the dependency tree. Due to the relatively poor quality of sentences that most users generate while typing their comments, we include as negations additional words that are not identified as negations in the dependency tree.

For each word, we take all advmod, dep, det, and quantmod relations in the dependency tree where the current word occurs as the parent, as well as all advmod and dep relations where the current word occurs as the child. If the child word in the relation is “not” or “no” and the child word comes before the parent word in the sentence, then we attach the child word as a negation modifier to the parent word. Similarly, if the parent word is “not” or “no” and the parent word comes before the child word in the sentence, then we attach the parent word as a negation modifier to the child word. These simple additional modifiers correctly attach negations that the dependency tree does not identify due to fragmented sentences and other writing issues.

5.3 Finding Subjects and Objects

Similar to the modifier attachment phase, we identify most subjects and objects simply by looking at the various subject and object relations in the dependency tree. In fact, for direct and indirect objects we solely include the words occurring in the dobj and iobj relations of the dependency tree.

For subjects, we include all subject relations in the dependency tree as well as others. If the current word is a verb that is not involved in any subject relation but is the child in a cop relation, then the word’s subject is set to the cop relations’ parent’s subject.

Adjectives that do not occur in a subject relation or an amod relation are also marked with a subject if they are the child in a dep relation with a verb, noun, or adjective. If this is the case, then the adjective’s subject is set to the relation’s parent’s subject.

5.4 Resolving Entities

Resolving entities is a multistep process where we use the company names, product names, and generic entity resolvers to mark nouns as the current product, competing product, current company, competing company, reviewer, reader, other person, or unknown.

First, we attempt to mark all nouns by searching for matches in the various company/brand/generic lists. Some matches, such as a match in the generic product noun list, require additional checks. For example, when coming across the noun “product”, if the noun is involved in a dependency with any word in the competing indicators list (such as “other”), then the noun is marked as a competing product.

After marking all nouns that match one of our lists, we mark each sentence with a head entity. While parsing the sentences, we use the Stanford Parser’s head word marking functionality to mark the head word of each phrase. Each sentence’s head entity is the word that has been marked as a current / competing company or product that is closest to the root of the sentence as dictated by the phrase structure tree. If no words in the sentence were marked as current/competing company or product, then the sentence’s head entity is assumed to be the same as the previous sentence’s head entity. If no words in the sentence were marked and the sentence is the first sentence in the review, then the head entity is assumed to be the current product.

After marking each sentence with a head entity, we resolve “it” and other similar words. To resolve these words, we compare their position with the sentence’s head entity’s position. If the “it” word occurs before the sentence’s head entity, then we assign the “it” word to the previous sentence’s head entity type. If the “it” word occurs after the sentence’s head entity, then we assign the “it” word to the current sentence’s head entity type.

5.5 Connecting Words to Entities

After resolving all of the entities, we determine if each word in each sentence is connected to a competing company/product or the current company/product.

5.5.1 Connecting to Competing Entities

A word is locally connected to a competing entity if one of the following cases is true:

- The word itself is marked as a competing entity.
- The word has a subject, and the subject is marked as a competing entity.

To determine if a word is connected to a competing entity, we first check to see if it is locally connected to a competing entity. If it is not locally connected, then we recursively follow prepositions checking each word for a local connection. If the current word is not connected through prepositions to a competing entity then we lastly perform the local check on all words that are in a relation with the current word in the dependency tree. If none of these checks turns up a word that is locally connected to a competing entity, then the current word is not connected to a competing entity.

5.5.2 Connecting to Current Entities

The check for whether or not a word is connected to a current entity is identical to the competing check, except that we do not perform the recursive preposition check.

6 Tag Extraction

After summarizing and annotating each sentence, we proceed with extracting tags. For each tag we cycle through the tag synonyms trying to find a match in the sentence for the synonym. For a given tag synonym, such as “Easy Controls”, we first check our Word Synonyms for a synonym list for each word. Thus we have two matchers: the Word Synonym for “easy” and the exact string matcher for “controls” (which is actually reduced to “control” during the stemming process). We take each matcher and find all of the words in the sentence that match.

For each matcher in the tag synonym, at least one word from the sentence must be a match. Thus if the sentence contains a match for “easy” but not for “controls” then the tag synonym does not exist in the sentence. The matching process does not require the same part of speech tags, so the matcher for “easy” will match adverbs such as “easily” or even nouns if the word “easy” is parsed as part of a compound noun.

After finding all of the words that match each matcher, the system performs various heuristic checks on each combination of the words to determine if the set of words should be extracted as the tag. For example, in the sentence “I needed something easy, and this has really simple controls”, both “easy” and “simple” would match to the “easy” matcher while “controls” would match with the “controls” matcher. Thus the system would perform the heuristic checks (that we are about to describe) on the pair (“easy”, “controls”) and then on the pair (“simple”, “controls”).

Our heuristics are designed to eliminate as many sets of matched words as possible that should not be extracted as the given tag while still accepting most correct sets. In the following sections, “matched word” refers to a word in the sentence that is a match for one of the matchers.

6.1 Matched Words Must Be Related

All matched words in the sentence must form a connected graph in the dependency tree. There are a few exceptions such as ignoring ccomp, none, and prep_for relations in the dependency tree and no two matched words can exist in a conj_but relation. This obvious requirement successfully eliminates matches such as our previous example with the pair (“easy”, “controls”) in the sentence “I needed something easy, and this has really simple controls”. In this example, “easy” and “controls” do not share a dependency relation with each other. Thus they do not form a connected graph in the dependency tree and the pair of words is rejected as a tag match.

6.2 No Connections to a Competing Company or Product

None of the matched words can be connected to a competing company or product within the category (as determined by Section 5.5.1).

6.3 At Least One Connection to the Current Company / Product

At least one of the matched words must be connected to the current company or product (as determined by Section 5.5.2). If none of the matched words are connected to a current entity, then the sentence’s head entity must be marked as the current company or product. If neither condition is satisfied, then the set of matched words is rejected as a tag match.

6.4 Compound Nouns

If a matched word is part of a compound noun and the entire compound noun is not in the matched word set, then the tag match is rejected. This solves issues, for example, where the reviewer says that the GPS device has “good sound quality” and you are trying to extract the tag “High Quality”. While having good sound quality increases the probability of the product being of high quality, other features of the product may be extremely poor. Thus it is not safe to extract “High Quality” from such a sentence. It is important to note that often “sound” in “sound quality” will be parsed as an adjective modifying the noun “quality”, but in many cases due to capitalization and non-standard sentence structures, it will be parsed as a compound noun as presented here. Potential tag matches that are rejected solely due to this heuristic can be aggregated and presented to the system’s operator as suggestions for new tags to extract, but we do not detail the process here.

6.5 Modifying Adjectives

If a matched word is a noun and it has a modifying adjective that is not a matched word, then the tag match is rejected. This eliminates potential problems where an adjective that is not a matched word could completely change the meaning of the phrase. It also solves similar problems as in the compound noun section such as trying to extract “High Quality” when the text includes “good sound quality”.

6.6 Matched Noun Dependent of Non-Matched Adjective

If a matched word is a noun and it is in a dep relation with an adjective that is not a matched word, then the tag match is rejected. This heuristic is similar to the modifying adjective heuristic of Section 6.5. It basically applies the same restriction for relations involving adjectives that the parser could only say was some kind of dependency, but the parser could not definitively say that it was a modifying adjective of the word.

6.7 Single Noun in prep_for

If the matched word set consists of a single noun that is in a prep_for relation, then the tag match is rejected. This heuristic solves, for example, the problem of extracting the tag synonym “comfort” from the sentence “Bad for comfort” since the matched word “comfort” is involved in a prep_for with another word. Unfortunately, this also rejects many perfectly reasonable tag matches such as with the sentence “Great for comfort”, but many of these lost tag matches can be easily recovered by including a tag synonym such as “good comfort”.

6.8 Matched Adjectives Modifying Non-matched Words

If a matched word is an adjective and it is modifying a non-matched word or its subject is a non-matched word then the non-matched word’s entity type must be the current company/product or unknown. Otherwise the tag match is rejected.

6.9 Appear, Suppose, Should, Could, and Would

Appear, suppose, should, could, and would are difficult words to process. Each can be used to describe how the product performed or how the product should have performed. For example, a reviewer might say that the product was “supposed to be easy to operate”. This could be followed by both “but I found it challenging”, or “and it definitely was”. One might think that a good heuristic for this particular example is the use of “but” versus “and”, but empirically this potential heuristic does not succeed for our dataset. Determining computationally whether the person meant “easy to operate” or its negation requires additional semantic computation of the sentence that is more complex that it is worth. Instead, we simply throw out all sentences where a matched word is directly connected in the dependency tree to appear, suppose, should, could, or would.

6.10 Negations

Identifying negations accurately is essential for our system. If a set of matched words have passed all other heuristics, the particular tag or its opposite will be extracted from the sentence. If the system does not catch a negation or identifies a negation when one does not exist then the exact opposite of the reviewer’s true meaning is extracted, which is arguably worse than not extracting anything at all.

For each matched word we count the number of negations that apply to it. First, we count the number of local negations. A local negation exists if the matched word is the parent word in a neg relation in the dependency tree, or if the matched word is in a relation with words such as “no”, “not”, or “never” (which are not always classed as a neg relation in the dependency tree due to parsing issues and incomplete sentences). A word is also locally negated if it is in a mark relation with the word “if” in the dependency tree. This covers statements such as, “Would be great if it fit the child's head" while extracting the tag "Good Fit".

After counting local negations we add any local negations of adjective or adverbial modifiers to the count. We also follow all acomp, xcomp, ccomp, and prep_for relations with the matched word as a child and include the relation's parent's local negation count in the current matched word's total negation count. For example, following the acomp relation successfully applies the "not" negation to "comfortable" in the sentence "It does not feel comfortable." In this sentence "not" applies to the word "feel", which is the parent of an acomp relation with "comfortable".

Similarly, following xcomp relations successfully includes the negation in the sentence "I would not call these shoes comfortable" where "call" is in an xcomp relation with "comfortable". Following ccomp relations catches sentences such as "I do not think that it is comfortable", but a few extra restrictions must be imposed for ccomp relations with parent words such as "believe" in the sentence "I cannot believe how comfortable this is".

After counting all negations that apply to the matched word, the word is determined to be negated if the negation count is an odd number. If any of the words is negated and the tag synonym itself is not negated then the opposite of the tag is extracted. Similarly, if none of the matched words are negated but the tag synonym is negated then the opposite tag is extracted. Otherwise, the current tag is extracted.

6.11 Tags Inside Other Tags

After attempting to extract each tag from a review, we perform one final tag elimination heuristic. The system up to this point is capable of extracting multiple tags from the same set of words. For example, the Puzzle category might have the tags "Easy to Assemble" and "Simple". If a review contains the sentence "This is easy to assemble", the system will extract both tags, which for most purposes is undesirable. To solve this issue we throw out any tag whose matched words are a strict subset of another tag's matched words.

7 System Performance

To evaluate the performance of our system we extracted tags from all reviews in the Automotive GPS and Dress Shoes categories of Buzzillions.com. Table 7a summarizes the data from both categories while Table 7b/c summarizes the results. We used 88 tags for extraction in the GPS category and 133 in the shoe category. As shown in Table 7a, the GPS category's PR reviews contain twice as many user-generated tags as the shoe category's PR reviews. The vast majority of PR reviews (around 98%) contain user-generated tags.

Table 7a: Dataset Description

	GPS	Shoes
# Imported Reviews	1,039	9,860
# PowerReviews Reviews	3,278	7,666
Average number of user-generated tags in a PR review	6.19	3.12
Percent of PR reviews with user-generated tags	98.78%	97.93%
# Pros used for extraction	43	60
# Cons used for extraction	45	73

Table 7b shows metrics most closely related to the traditional recall metric in natural language processing. We see that in both categories, more tags are extracted from imported reviews than PR reviews. This is most likely because the imported reviews were generally longer since the user did not have to input user-generated tags before writing the review.

Our system was able to extract more than twice as many tags per review from the shoe category, but the shoe category also had twice as many extracted tags that already existed as user-generated tags. This was in part due to using more tags for extraction in the shoe category and because the GPS category's tags were often more complex such as "Acquires Satellites Quickly" and "Complicated Controls" while most of the shoe tags were straightforward such as “Cute”, “Too Tight”, and “Fashionable”.

An important note in Table 7b is that for both categories less than 30% of the tags extracted from PR reviews already existed as user-generated tags. This shows that the system can produce substantial positive results even when processing reviews where the reviewer is explicitly asked to input pro and con tags during the review writing process.

Table 7b: Recall Results

	Imported		PowerReviews	
	GPS	Shoes	GPS	Shoes
Average number of extracted tags per review	0.69	2.13	0.68	1.46
Percent of reviews with extracted tags	47.5%	89.9%	44.0%	77.6%
Percent of extracted tags that already existed as user-generated tags	0%	0%	15.9%	28.0%

Table 7c shows the precision of the extracted tags for each category. Due to resource constraints, we were unable to hand-evaluate every extracted tag. Instead, we evaluated a subset of each category by randomly sampling the extracted tags. Our system performed well on both categories, achieving around 95 ± 2%.

Table 7c: Precision Results

	GPS	Shoes
Precision	94.77%	95.45%
Margin of error at 95% confidence	2.13%	2.05%

8 Conclusion

In this paper we have described a production-capable system for extracting pros and cons from product reviews. The system draws upon many sources of data for product/company names, word synonyms, and tags to extract. Many of these data sources, such as product names, can be automatically compiled from websites such as Amazon.com. Other data sources, such as the tags to be extracted from each category, require hand tailoring. Combining these data sources, the parsed dependency trees of the review text, and our tag extraction heuristics, our system is capable of automatically extracting highly accurate (~95%) pros and cons with a reasonable recall rate (up to 90% in some categories). Our system is also capable of extracting additional tags from reviews where the user explicitly chose pro/con tags during the review writing process, with 70% or more of tags extracted from the review text of such reviews being new tags.

9 Acknowledgements

We thank Robert Chea and the rest of the engineering team at PowerReviews for the product review data and computational resources for running our experiments. Todd Sullivan is supported by an NDSEG Fellowship sponsored by the DOD and AFOSR.

10 References

[1] Todd Sullivan, “Pro, Con, and Affinity Tagging of Product Reviews,” Stanford CS 224n. http://www.daysignmedia.com/research/graduate/nlp/product-reviews

[2] Bo Pang , Lillian Lee and Shivakumar Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," Proceedings of EMNLP 2002.

[3] Bo Pang and Lillian Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts," In Proceedings of ACL 2004.

[4] Kim, S-M. and E. H. Hovy, “Automatic Identification of Pro and Con Reasons in Online Reviews,” Poster. Companion Proceedings of the Conference of the ACL, Sydney, Australia, 2006.

[5] Dan Klein and Christopher D. Manning, “Fast Exact Inference with a Factored Model for Natural Language Parsing,” In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10.

[6] Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning, “Generating Typed Dependency Parses from Phrase Structure Parses,” In LREC 2006.