WEB HARVESTING AND SENTIMENT ANALYSIS OF CONSUMER FEEDBACK

Emil Ş. CHIFU , Tiberiu Ş. LEŢIA , Bogdan BUDIŞAN , Viorica R. CHIFU
Faculty of Automation and Computer Science, Technical University of Cluj-Napoca
Bariţiu 28, RO-400027 Cluj-Napoca, Romania, Emil.Chifu@cs.utcluj.ro

Abstract: This paper presents a technique for detecting, extracting and evaluating customer reviews given to online products or services. The technique proposed consists of a web harvesting algorithm and a sentiment analysis method. The web harvesting algorithm extracts customer reviews from Web pages. It extracts reviews given to online products or services, while the sentiment analysis method uses the data returned by the web harvesting method in order to evaluate the opinion polarity of the customer reviews. The technique proposed is intended to work generically, for the reviews and feedback of any given list of URLs which point to web pages where such data is available.

Keywords: *feedback, web harvesting, sentiment analysis, opinion-bearing, SentiWordNet.*

I. INTRODUCTION

Gathering data from web pages can prove extremely useful for domains such as online commerce, marketing and advertising, and even politics. All these areas rely heavily on statistics and studies involving large scale operations such as gathering data and intelligence from a large number of random people.

Thus, being able to recognize useful data which is already available on the Internet, as well as being able to acquire it in a format which can later be interpreted and processed, can prove extremely useful. These abilities could have commercial and scientific uses in industry, marketing, social networking, human studies, biology, or medicine.

In this paper we have combined a web harvesting algorithm with a sentiment analysis method to aid the task of determining the quality of a certain product or service. The data obtained through the web harvesting algorithm is evaluated using a sentiment analysis approach.

The web harvesting algorithm proposed here handles the detection and extraction of customer reviews given to online products or services. The algorithm is intended to work generically, for any reviews and feedback of a given list of URLs, despite the different ways of structuring and organizing the information, which is specific to each website. The purpose of the web harvesting algorithm is to successfully identify consumer-given reviews and feedback, from text embedded within and interleaved with markup code. Further, the algorithm aims to extract and convert the reviews and feedback into data which can be evaluated from the point of view of the sentiment polarity. In the end, the web harvesting process needs to produce its output in a plain text format, the actual review given by consumers, in natural language, stripped from the other content of the web page and markup code.

For the sentiment analysis part, the opinions expressed by the reviewers in the harvested feedback text are evaluated using the SentiWordNet lexical resource [1, 2]. We have used SentiWordNet for determining the semantic orientation of the opinion bearing linguistic components detected in the texts. Our algorithm for classifying the opinions expressed in the user reviews is inspired from the trigram pattern based method proposed by [3]. Finally, every extracted review is numerically quantified, based on the average obtained for the sentiment polarity value of its opinion-bearing linguistic constructions.

Because of a large range of possible structural variations of web pages, the web harvesting algorithm needs to follow some heuristic strategies. These will probably produce certain errors, and determining when such error cases occur and minimizing them by performing corrections will be required, to reduce the propagation of errors and any other flaws towards the sentiment analysis part. For a correct interpretation of the sentiment analysis results and, as a consequence, for improving – by heuristic means – the final sentiment classification, an additional small analytical study was conducted on the SentiWordNet raw data, separated from the web harvesting and sentiment analysis algorithms. Hence, both algorithms of our method – harvesting and sentiment analysis – are heuristic ones. Generically, as well as in our specific instance, heuristics are designed for solving an initially proposed problem, for finding an approximate solution when it is impossible to determine an exact solution. Completeness together with a slight amount of accuracy are traded and sacrificed to gain better speed and a general solution for any given input.

The paper is structured as follows. Section II presents the related work, section III illustrates the web harvesting algorithm, section IV analyzes the SentiWordNet database and how it will be used in the sentiment analysis process, while section V presents the sentiment analysis method. The experimental results are presented in section VI. We end our paper with conclusions and future work proposals.

Manuscript received May 27, 2015; revised September 7, 2015

II. RELATED WORK

This section presents some state of the art sentiment analysis methods. [3] uses surface patterns in order to identify opinion-bearing phrases (tuples of words) in text. Most of these part-of-speech patterns actually relate an aspect of an opinion target entity (the aspect being usually described by a noun or a noun phrase) to an opinion bearing word (usually described by an adjective). As opposed, [9] rely on a deep syntactic parsing, i.e. a dependency parsing, in order to extract dependency relations between an aspect of a target object and an opinion bearing word. The aspect of the target entity could be the target entity as a whole or some characteristic feature (attribute, component part) of the entity. The target object is a product or a service, for instance a photo camera or a touristic resort. The aspects are the features (attributes) of the objects, for instance the lens, the image quality, the weight (for a camera), or a beach, a ski track, a waterfall (for a resort).

In [5] the authors propose a method that is able to perform sentiment analysis on Twitter. The method collects data from Twitter and builds models for two types of classifications: (*i*) the first one classifies the sentiment into the classes positive and negative, while (*ii*) the second one classifies the sentiment into the classes positive, negative, and neutral. The considered models used in the experiments were a unigram model, a feature based model, a tree kernel based model, and a hybrid model. The latter model combines the principles from the unigram model, the feature based model, and the tree kernel model. Based on the experimental results, the authors concluded that the tree kernel based model and the hybrid model outperform the other models considered.

In [6], the authors illustrate a method that is able to: (*i*) automatically collect a data corpus (i.e. 300000 text posts) from Twitter, (*ii*) perform linguistic analysis on the data collected, *(iii*) build a sentiment classifier that is able to identify positive, negative, and neutral sentiments form the data considered. The method proposed has been tested for the English language, but it can be used for any language.

The paper [7] presents an approach based on machine learning techniques for classifying microblogs in positive and negative opinion classes. The approach proposed consists of three main steps: (*i*) data preprocessing, (*ii*) extracting the features, and (*iii*) training the classifier. In the first step the relevant data is extracted from the corpus considered by using natural language processing techniques (e.g. stop word removal, stemming, spelling correction, mapping emoticons to sentiments, part of speech tagging, filtering). In the second step, the feature sets are extracted by using unigrams, bigrams, trigrams, and a combination of the unigram and bigram approaches. In the third step, the feature sets are used to build feature vectors that will be used to train the classifier. In their experiments, the authors have considered the following classifiers: multinomial Naïve Bayes, logistic regression, random forest, and support vector machines.

In [8] the authors propose a new approach for identifying the product features from Web reviews, which is based on natural language processing and supervised pattern discovery techniques. The approach considers only the reviews in which the Pros and Cons regarding a product are described separately by the reviewer, who also provided a separate detailed review about the product. Additionally, the paper introduces a tool for analyzing and comparing the consumer opinions about the competing products from the market. The tool is able to provide the user – where the user could be either a product manufacturer or a potential customer – with the strengths and weaknesses of each product as it is seen by the (other) customers.

III. WEB HARVESTING ALGORITHM

This section presents the main steps of the web harvesting algorithm.

A. Implementing the HTTP Communication

The designed HTTP implementation should set a maximum timeout value to avoid getting stuck in a waiting state. It also needs to set a maximum number of allowed redirects, to avoid redirect loops or requests that take too long, otherwise these would also critically affect the overall time performance. Moreover, in order to ensure that the web harvesting algorithm works for any server, as we attempt to make it generic, switching between various *user agents* [11, 12] is required, until a successful response is received.

Storing *HTTP cookies* should also be implemented. Some servers might block the incoming request or redirect it to their homepage in case a user or script attempts to directly access an inner page, without passing through the whole path to it.

A successfully implemented HTTP communication should be able to provide the HTML source code of any web page, where the desired reviews will be interleaved with markup code (HTML/ XHTML) and scripts (CSS, JavaScript etc.). Sending the requests and waiting for the response will reflect poorly on the time performance. This depends mostly on external factors, like: the servers, the time required for sending the request, and the time it takes for the server to process the requests and send a response.

B. Response Post-Processing and HTML Sanitation

Before attempting to identify and extract the reviews contained in the downloaded HTML source code, there's a need to minimize the code as much as possible first, to reduce it to the parts which have the potential of holding the customer review texts. Blank lines, anything between script tags, applet code, CSS, canvas, input areas, media-related or browser-specific tags and all their contents should be waived. Also, in the remaining code, any HTML attributes and their values, except “id” and “class”, can also be waived. These two attributes are actually the most important structures in the web page source code, because we will use them to identify the customer reviews, so it is very important that they remain intact during the whole sanitation process.

By applying these filters, up to 80% of the source code can be waived, resulting in a significantly smaller amount of data to work with. This will reflect positively in the overall performance in execution time of the web harvesting algorithm. Working with smaller amounts of data also means that identifying false positives becomes less probable.

C. Review Identification and Extraction

The actual identification of the customer reviews is performed using a combination of the classical HTML parsing and data mining techniques and technologies, more precisely, a *DOM parser* customized with *regular expressions*. We used regular expressions for selecting the desired (i.e. domain specific) set of HTML nodes together with their content. Note that while using regular expressions alone (i.e. without the DOM parser) may seem an easy and

handy solution, it is widely accepted that for HTML parsing it is a bad idea. This holds especially in our case where we want to achieve a universal web harvester, not one fit only for some certain websites with a known and fixed structure matched by a set of given regular expressions. Regular expressions are used to match regular languages, while HTML is a context free language. Thus a DOM parser is nevertheless needed as a syntactic parser to ensure the extraction from generic HTML files of specific information as customized by a set of domain specific regular expressions.

For defining the domain which is to be extracted, a set of relevant and illustrative keywords should be specified to reflect that domain. The words should be few and not loosely chosen, only words whose primary meaning is absolutely relevant to that domain, otherwise the risk of identifying false positives increases greatly. For our research purposes, for identifying and extracting customer reviews and feedback, the following list of keywords was used: *review*, *feedback*, *user*, *customer*, *consumer*, *rating*, *opinion*, *summary*, *revision*, *analysis*, and *report*. In case the data of interest is not successfully extracted, it's more likely that a new keyword should be added to that list rather than conclude the harvesting method is wrong. A new domain specific keyword can be identified by studying the source code of the specific web page which is to be harvested.

Once the HTML parsing method is chosen, it should be set to extract all the HTML elements which have the value of the "class" and "id" attributes matching any of the given domain specific keywords. This matching is accomplished with the help of domain specific regular expressions that are customized according to the domain specific keywords. By matching, it is also implied here that attribute values for which the keywords are lexical substrings are also valid patterns. For example, all the following HTML snippets qualify as valid keyword matching patterns.

```
<div class="review">...</div>
<div class="SomeClass"
     id="ProductReview">...</div>
<p class="userRevs">...</p>
```

The usage of a defined list of domain specific keywords which identify the domain of interest from which the related data is to be extracted, and using these keywords in regular expressions within the DOM parser selector is one of the most important touches of originality of our web harvesting algorithm. It is confirmed by the experimental results reported in section VI-A. Nevertheless, this is a heuristic method, and consequently it might produce a fair number of false positives. But capturing other elements from the page (e.g. extracting the user name due to the "user" keyword) or from menu headers is harmless, since it is highly unlikely that such false positives are opinion-bearers. So towards the text polarity evaluation, being detected as sentimentally neutral, they would be irrelevant and discarded anyway, and thus they would not influence the final result.

The review detection method greatly depends on websites using good HTML coding practices, such as having relevant names for their attributes, to ease code readability, and applying CSS and JavaScript selectors, descriptors etc. The method would fail in case of websites which use unusual or random strings as attribute values, and would also fail in case of obfuscated markup code. But such situations appear to be rather rare, and all the sites we've encountered during our research and we've used as input were employing relevant attribute values in their markup code and could successfully be parsed.

D. Review Post-Processing

In the end, there should only remain a list of unique and maximal (outmost) HTML elements relevant to the given keywords (after removing all the duplicates and subsets), i.e. HTML elements whose *class* and *id* attribute values match the given set of domain specific keywords. At this point, the remaining HTML elements contain the customer reviews, but they are still embedded within HTML tags. From these HTML snippet codes, all the markup code can be safely removed, and their text content should be kept.

We have decided to treat all the reviews of a certain product or service as a single one, as opposed to separately identifying them and treating and evaluating each review individually. It would be unfair, for example, to give the same weight and thus, rate as neutral a product or service described by one positive review containing 100 words and one negative review containing only 15 words. Not only that it would be actually quite challenging to manage to successfully identify each review individually, but taking all of them as a whole grants them weights proportional to their length and accuracy. So while it seems fair to treat all the reviews together, it also keeps things simple.

E. Web Harvesting Results and Conclusions

One might think here that an obvious and maybe better alternative to our method would be to define a set of keywords or an ontology containing review and opinion related words. Accordingly, we could propose as a solution the attempt to detect those words inside the content of the HTML elements and identifying and extracting those passages as being the correct reviews. The problem we identified here is that too often the description of the product or service (written by the product seller or service provider) would still match the words of a defined ontology and would usually be too similar to a user or customer given review of product/ service. The purpose of the product/ service description being to sell that certain product or service, it will always be positive and subjective, as being a marketing technique.

Identifying and extracting the product description along with the customer feedback and customer reviews would greatly affect the overall evaluation and classification of that product, and it would pull its value up from its actual one, if it were taken into consideration. This is why we preferred to avoid using this kind of harvesting method, based on semantic context detection. Focusing on a structural identification method like ours (based on the *class* and *id* attributes) is most likely a better solution. It is possible that a combination of our structure-based method with some form of context and meaning detection would yield better results and accuracy than the web harvesting algorithm we present here. But that already exceeds the object of this paper, while it still remains a good area for potential improvement and further research.

Our harvesting algorithm is a heuristic method, so while it does not guarantee absolute correctness, it is proven to work properly in most of the identified situations. This method retains good reliability and versatility for a wide range of different website structures and ways of specifying and delimiting the customer reviews and feedback from the rest of the web page content. We measured a 93.3% rate of

success for a list of 223 different URLs from various websites containing reviews and feedback.

IV. SENTIWORDNET

Before moving on to the sentiment analysis algorithm, a brief presentation of SentiWordNet [1, 2] is required and also presenting the statistics which we have obtained about this lexical database and how our resulting findings reflect in the sentiment analysis process.

A. The SentiWordNet Lexical Resource

SentiWordNet (*SWN*) [1, 2] is a lexical resource based on the WordNet [4] lexical database, containing a large amount of data, specifically words and associated values. It is designed to aid in the process of sentiment analysis. In *SWN* each synset from *WordNet* is associated with two absolute numerical values which represent the *positive-negative polarity* of the words in the synset. A third value, *objectivity*, can be deduced from the positive and negative scores (see equation 1). All these three values range within the [0, 1.0] interval. *SWN* is constructed in such a manner that their sum is exactly 1.0 for each word (see equation 1).

$$obj_{score} = 1 - pos_{score} + neg_{score} \quad (1)$$

SWN provides already calculated positivity and negativity scores for each synset. But a certain word might have multiple appearances (based on how many different meanings it has) in *SWN*, each in a different synset and with different polarity values. The *SWN* database file needs to be parsed and converted to a temporary hash structure, with each word used uniquely as key, and its values represented as an inner hash inside the initial hash, with each of its positive-negative values for the different meanings of the word. For any given word, the meanings that appear first in the file are the ones more frequent for that word. From this first created hash of hashes structure, a new hash structure needs to be constructed, using the word as unique key, and the total calculated score of all its meanings as a single value. We will use these latter values in order to determine the polarity of the opinion-bearing tuples (i.e. syntactic patterns) found in the customer reviews, during the sentiment analysis algorithm.

Since our sentiment analysis algorithm does not differentiate between the different meanings of a word depending on the context it fits in, we need to obtain a single value for every word in *SWN*. Such an overall score of the word is calculated as a weighted average (according to [1, 2]), which weighs stronger the value of the most likely and often used meaning of the word in English texts, and then gradually, each of the less frequent meanings of the word has a lower weighted score.

B. SentiWordNet Statistics and Analysis

We decided to calculate the overall average of all the non-null synsets in *SWN* and also to count how many of the word scores fit within certain ranges. This will help to properly be able to find a representative mathematical formula for quantifying the identified opinion-bearing tuples (i.e. *n*-grams), and after that, to use fuzzy classification for giving the final rating.

Thus, by calculating the overall average of all the words whose polarity value was different from null, we determined that there were 39963 such unique words (each with multiple possible meanings and scores), with a slightly negative (−0.046222) total average value. The number of unique words in *SWN* was counted to 155287, which means that 75% of the *SWN* words (mostly the nouns) have a null polarity score, and thus do not qualify as opinion-bearers. By counting the words which fit in certain ranges, we concluded that 66% of the non-null *SWN* words are within either the [−0.25, 0) or the (0, 0.25] intervals, so most of the opinion-bearing words have quite low values to begin with.

These results and observations are reflected in the way we further measured and quantified the overall review scores and the opinion-bearing tuple scores. Setting the value for the neutrality threshold (which is used for determining whether an apparent opinion-bearing tuple is subjective enough to be taken into account for evaluation) was also based on these findings.

V. METHOD FOR SENTIMENT ANALYSIS FROM CUSTOMER FEEDBACK

Sentiment analysis will be used to determine and quantify the overall opinion customers expressed about a certain product or service. We use here *SWN* as a lexical resource, and start from the 3-gram pattern based sentiment analysis algorithm proposed in [3]. Each review will obtain partial scores, based on its opinion-bearing linguistic components, and the average value of these scores will be calculated and rated using a fuzzy classifier.

A. Part of Speech Tagging

The first step for determining the overall opinion of a text is to determine the polarity of its linguistic components. The text is split into separate sentences and then each sentence is split into words and evaluated. In order to determine the opinion-bearing linguistic constructions (e.g. “great product”), *part of speech (POS) tagging* of each word in the sentence is required. POS tagging the words of a sentence could lead to bad results in cases like punctuation marks or capital letters at the start of each sentence missing. Such small errors might be caused by the user who wrote the review or by the heuristic web harvesting algorithm. POS tagging tools might get confused by the lack of a capital letter after a full stop, by considering the full stop as a marker for an abbreviation and considering the next sentence as part of the previous one. So before applying POS tagging (we have used the Lingua tagger [10] for POS tagging) to the sentences of a review, attempts to correct the text should be made.

Applying POS tagging to a sentence will output its words with POS tags attached to them, usually separated by a slash character (e.g. “great/JJ product/NN”). After POS tagging, a new correction should be attempted, since a POS tagger might mislabel certain words. For example it might tag all capital letter words as proper nouns (NNP), instead of their true POS. Correcting this can be achieved by looking up all the words detected as NNP into the *SWN* database and, if found, correct the mislabeled NNP with the POS found in *SWN*.

B. Detecting the Opinion-Bearing Tuples

For determining the opinion-bearing tuples (or synonymously, n-grams, phrases, syntactic patterns), in each sentence of a text we search for sequences of three neighboring words whose part of speech conforms to certain lexico-syntactic patterns. So we actually identify 3-grams, like in [3]. Out of each trigram, we only keep the first two

words encountered. Thus, what we call generically an opinion-bearing tuple is rather a syntactic phrase consisting of a simple pair of words in our approach. To identify an opinion-bearing tuple we have used the English language trigram syntactic patterns proposed in [3] and illustrated in Table 1. So, despite using three consecutive words for matching a pattern (i.e. a trigram), only the first two are kept to build an opinion-bearing tuple, as being the only actually opinion-bearing words in the trigram. Each of the two words in the opinionated tuple (pair) will receive its corresponding sentiment polarity value from the SWN database. The third word in the original trigram is only needed for identifying grammatically the proper opinion bearing tuple in the text (see Table 1).

Table 1. Potential opinion-bearing POS Patterns [3]

1st Word	2nd Word	3rd Word
JJ or JJR or JJS	NN or NNS	Anything
RB or RBR or RBS	JJ or JJR or JJS	not NN nor NNS
JJ or JJR or JJS	JJ or JJR or JJS	not NN nor NNS
NN or NNS	JJ or JJR or JJS	not NN nor NNS
RB or RBR or RBS	VB or VBD or VBN or VBG	Anything

C. Evaluating the Detected Tuples

Once the potential opinion-bearing tuples have been determined, the opinion polarity value of each word in the tuple needs to be extracted from the hash structure built out of the SentiWordNet lexical resource (*SWN*). The research papers that define *SWN* [1, 2] reflect how its stored sentiment polarity values were calculated. Yet, the authors do not impose a way of how the values in *SWN* should be used for sentiment analysis, so we based our decisions on the *SWN* analysis we performed, reported in section IV-B.

According to our analysis of SentiWordNet, most of the nouns will contribute with a null value to the total tuple evaluation. Moreover, most of the non-null (i.e. subjective) words have really low values to begin with; 66% of these words have polarity values in the [−0.25, 0.25] interval. Consequently, we decided to use simple addition to obtain an overall tuple value to characterize the pair of words composing the tuple. The arithmetic average of the two words in the tuple would yield a value which would be too low, leading to a reduced discrimination power among the reviews from the point of view of the opinion polarity value. A very positive review would become almost indistinguishable from a slightly positive one, the difference between them being an insignificant amount. By using addition, null words in opinion-bearing tuples are ignored, instead of dragging the whole average value down. We are well aware that through addition, an opinion-bearing tuple can, in theory, score higher than 1 or lower than −1, but slightly going over the limit with a few tuples is easily compensated by the inevitable tuples which are detected as false positives and will obviously receive a polarity value close to zero (i.e. neutral polarity, as being false positives, so not opinion bearing). However, in practice, we observed that a tuple scoring over +1 or below −1 occurs extremely rare and it is a negligible amount in the scope of the whole algorithm.

A final evaluation of a review will be calculated as the average of the scores of all the opinion-bearing tuples detected in the text of the review. In order to improve the discrimination power among the sentiment polarities of the reviews, we define a threshold value, which has been heuristically established as a result of our statistical analysis of the *SWN* lexical resource (see section IV-B). More exactly, we use a threshold over which an apparent opinion-bearing tuple found through POS positional 3-gram pattern matching is considered as truly expressing a strong, subjective opinion. We talk here about a *Subjective-Objective polarity* axis, besides the *Positive-Negative polarity* axis associated to the subjectivity values, i.e. the opinion polarities. Thus, we define a *neutrality threshold* value, to eliminate from the final evaluation those tuples which express a weak, diluted opinion as having a reduced polarity value. Such weakly polarized linguistic constructions occur quite often, their concentration increasing directly proportional to the quantity of words in the review. Taking them into account would seriously flatten out the final average value for the review, causing any final evaluation to tend strongly towards a null average value, despite the reviews being actually strongly positive or strongly negative. Based on our statistics calculated on *SWN*, which shows that two thirds of the non-null words have their values within [−0.25, 0.25], and that three quarters of the words in *SWN* have null values, we have set this neutrality threshold to 0.2 for positive tuples and −0.2 for negative tuples. Thus, tuples expressing opinions polarized on values situated between the two positive and negative thresholds – revolving around zero – are considered more likely to be neutral rather than be polarized towards a direction, so they should be waived. The *subjective-objective polarity* value for a tuple will be calculated by adding the objectivity polarity value (see equation 1) of the two words composing the tuple.

This subjective-objective value obtained for a tuple is to be compared with the heuristically defined neutrality threshold *N*. We are interested in strongly subjective tuples (see equation 2), as these are the true carriers of sentiment. Weakly subjective tuples (see equation 3) will be discarded.

$$tuple_{strong} \in (-\infty,-N) \cup (N,+\infty) \quad (2)$$

$$tuple_{weak} \in [-N,N] \quad (3)$$

At this point, we should have the text of the customer review broken down into a list of strongly polarized opinion-bearing tuples. Each tuple has its own, already determined sentiment polarity score, calculated by adding the individual score of each of the two words in the tuple.

D. Final Classification

For each review, the average value of its identified opinion-bearing (i.e. subjective) tuples can now be calculated. The final average value will most likely be within the range of [−1.0, 1.0]. To give it a human understandable format, the final score is to be converted from its numeric form by using a fuzzy classification. If the final score is greater than zero, the review is overall positive; otherwise it is classified as negative. By judging in which subinterval of the range of values [−1.0, 1.0] the final score fits in, this final score is translated using its overall *Positive/ Negative* polarity label and an *adverb of degree* (i.e. *Strongly, Very, Moderately*) (see Table 2).

Table 2. Fuzzy Classification

Range	Classification
[0.5, +∞)	Very Positive
[0.2, 0.5)	Strongly Positive
[0, 0.5)	Moderately Positive
[−0.2, 0)	Moderately Negative
[−0.5, −0.2)	Strongly Negative
(−∞, −0.5)	Very Negative

VI. EXPERIMENTAL RESULTS

This section presents the experimental results obtained for the web harvesting algorithm as well as for the sentiment analysis method.

A. Web Harvesting Results

A set of 223 different URLs was used as input for the web harvesting part. The URLs were valid inputs, pointing towards web pages containing consumer feedback data from multiple different servers. Pages from important online commerce and online service providers were used, such as Amazon.com, Epinions.com, RottenTomatoes.com, iTunes.Aplpe.com, cNet.com, BarnesandNoble.com, Fishpond.com.au, Play.com, Kmart.com and others.

The obtained results represent (see Figure 1) the average value obtained through multiple runs on the same set of input data. Each different run achieved different results in terms of both time performance and accuracy, because the availability of the web sites and network connections may vary from time to time, not predictably, or too many requests to one and the same address may lead to blocking further requests to that address. The arithmetic average of these multiple runs is more relevant than considering isolated runs which might fall into particulars of exceptional cases. Presenting an average of multiple runs performed on a large input data set should provide a relevant measure of the results and thus the solution.

- Fetching the HTML sources of the given web pages URLs worked in 97.3% of the cases, meaning that an average amount of 217 HTML sources were successfully received out of the 223 given URLs.
- Identifying and extracting reviews from the fetched HTML source files was successful in 95.9% of the cases, meaning 208 reviews.
- These amount that out of the 223 input URLs, the reviews for 208 of them were successfully extracted, meaning 93.3%.

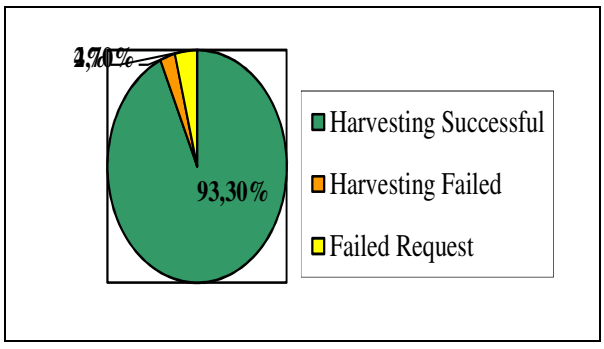

Figure 1. Web Harvesting Results.

Some reviews are missed (not identified) because of the need to add new keywords to the domain specific set of keywords for the web harvesting algorithm, as mentioned in section III-C. The same section III-C also enumerates part of the other reasons why some reviews are not identified and extracted from the fetched HTML sources: irrelevant names for the attribute values (such as non-English or random words), obfuscated markup code, and, very often, reviews displayed dynamically with AJAX (i.e. not explicitly stored in the HTML source). Section III-A mentions other cases when the desired reviews are missed, specifically when the home page is rather sent by the site server instead of the requested inner page containing reviews, or similarly when the 404 error page is sent successfully (i.e. with success code returned).

An average of 1204 seconds was required to fully process the 223 URLs, from requesting their source code to evaluating the reviews they contained. This yields an average of 5.39 seconds for all the operations per web-page. Most of the web harvesting running time was consumed by the HTTP communication part (average duration of 1040 seconds, meaning 86.36% of the total running time).

F-measure (see formula 4) was computed on a number of 20 randomly chosen reviews out of the 208 successfully extracted reviews, in order to measure the relevance of the web harvesting solution.

$$F - measure = 2 * \frac{prec * rec}{prec + rec} \tag{4}$$

where:

- *prec* is the precision and represents the ratio of retrieved instances that are relevant;
- *rec* represents the recall and is computed as the fraction of relevant instances that are retrieved.

In our case, for any given review, the relevant instances in formula 4 are the true words extracted from the review, and the retrieved instances are all the words extracted from the review. For each of the 20 reviews, we selectively counted the true words extracted from the review in relation to the total number of words extracted from the review. By true words we understand here the words actually extracted from the pure natural language portions of the review and not from the HTML markup code of the review, i.e. tag names, attribute names, and attribute values. What was noticed is that in almost all the cases, there were no missed review fragments from the original web pages source code. Consequently, almost all of the errors were false positives – additional information which was captured from the HTML source code besides the actual review, i.e. the words that are not true words, as mentioned above. This results from the overall heuristics of the method, which strives for generality instead of being restrictive, by favoring the recall over the precision. Being rather permissive, the method extracts false positives that proved to be sentimentally neutral, and thus not affecting the polarity evaluation of the review. Capturing false positives is much more desirable than having misses (i.e. false negatives), since the latter would mean missing sentimentally relevant portions of the reviews.

Thus, in the *F-measure* equation, for most of the cases the *recall* was 1.00. By calculating the *F-measure* for the 20 reviews, the obtained values ranged from 0.25, for the worst harvested data, to almost maximum, 0.98, for which the harvesting method worked exceptionally. The average *F-measure* calculated for the 20 reviews amounted to 0.73,

which is a good score. A sample set for 5 randomly chosen reviews is illustrated in Table 3.

B. Sentiment Analysis Results

The data set used in our experiments consists in a number of 208 reviews posted as feedback by the customers/ consumers and successfully extracted by the web harvesting algorithm. The format of the data is plain text in natural language, in English.

First of all, we noticed that, consistent with the experimental results from [3], we also recorded a lower accuracy for movies and books reviews. In these reviews, the context tends to have a stronger influence on the polarity of the text. Meanwhile, reviews which fall into a somewhat more technical and less abstract domain benefited of a more accurate final classification.

The sentiment analysis algorithm managed to successfully detect opinion-bearing tuples and evaluate 96.5% of the 208 reviews analyzed. Opinion-bearing tuples could not be extracted from the identified reviews in some cases when the reviews are irrelevant or have a bad format, since their authors wrote them badly. This is the case, for instance, when the reviews are too short (e.g. "cool product!", which is not a valid sentence as having no predicate) or when they have grammatical errors. In all such cases, the part of speech (POS) tagging process is unable to work properly and does not give correct results, as already mentioned in section V-A. Likewise, the lack of punctuation marks could lead to bad POS tagging and as such to errors in extracting the correct opinion-bearing tuples. Correct POS tagging is required by our method for extracting opinion bearing tuples, since the method relies on matching POS 3-gram patterns in the text of the reviews. Moreover, since the POS tagging works only for English, reviews written in any other language cannot be POS tagged and consequently the opinion-bearing tuples could not be identified by our POS 3-gram pattern based extraction method.

Time measuring was used to be able to provide the following statistics:

- The sentiment analysis operations (considering the reviews already given) performed in approximately 164 seconds for the average number of 208 reviews that qualified to this stage out of the 223 given URLs;
- The average counted total number of processed words for each run was 313932. This is the number of words which were processed overall. These results amount to the conclusion that for a review with an average number of 1555 words, the average evaluation time was 1.227 seconds. These time measurements might not be very relevant because of their hardware dependency.

To measure the quality of the automated sentiment analysis method, we performed a comparison of the obtained results versus a human evaluation. The texts of the 208 captured reviews were given to human experts who were instructed to grade them as correctly as possible. They were asked to use in their evaluation a term from our list of possible fuzzy classifications (from very negative up to strongly negative, moderately negative, moderately positive, strongly positive, very positive). Then, our sentiment analysis algorithm was used to rate the same review text files. The comparison was made based on a graded similarity, by retaining the difference between the human and the automated method. Thus, a difference between a human detected very positive review and an automated detected very negative review will lower the overall quality of the similarity score more than the difference between a very positive and a moderately positive review. By comparing the two sets of results (human detected and automated detected), an overall accuracy of 67% was measured (see Figure 2). Out of the total 208 reviews, 21 were movie and book reviews, for which we already determined that our sentiment analysis method has a lower accuracy. This happens because the reviewers tend to mix-up the review with the summary of the book or movie, which affects both the subjective-objective and positive-negative polarity in unpredictable ways. In these cases, it might be considered that it's the human reviews which are inaccurate, thus making bad reference systems.

Judging the results only for the books and movies reviews, the accuracy decreased significantly, for obvious reasons, to only 53%. If these books and movies reviews were to be isolated, the accuracy measured for the remaining reviews was quite good, 69%. It is to be noted here that an accuracy of 70% can be considered human-like.

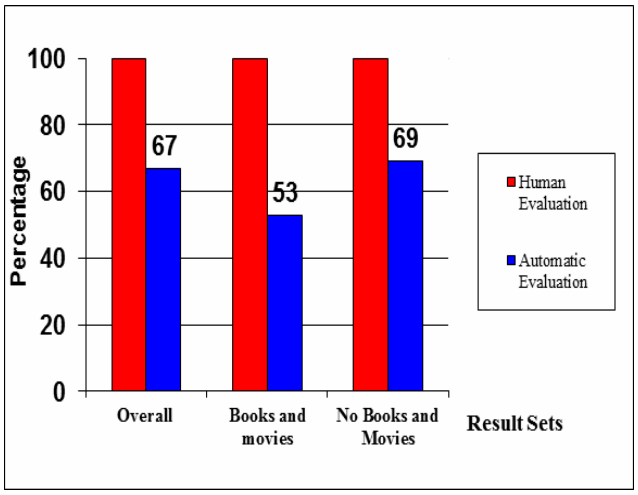

Figure 2. Sentiment Analysis Results.

Table 3: F-Measure for Web Harvesting

Total Words	Relevant Words	False Positives	Misses	Precision	Recall	F-measure
877	548	329	0	0.63	1.00	0.77
1809	1701	108	0	0.94	1.00	0.97
454	405	49	0	0.89	1.00	0.94
3203	1853	1350	0	0.58	1.00	0.74
523	493	30	0	0.94	1.00	0.97

Table 4: F-Measure for Sentiment Analysis

Review Words	Actual Relevant Tuples	Detected Tuples	False Positives	Misses	Precision	Recall	F-measure
475	7	5	0	2	1.00	0.72	0.84
865	18	16	2	4	0.88	0.78	0.83
246	9	7	1	3	0.86	0.67	0.75
467	14	12	1	3	0.92	0.79	0.85
1903	56	54	5	7	0.91	0.88	0.90

F-measure was calculated for a set of randomly chosen 20 reviews, by counting the relevant opinion-bearing tuples which were detected and those which were missed by the method. An average value of 0.80 was obtained, which is a pretty good result. A sample set for 5 randomly chosen reviews is illustrated in Table 4.

VII. CONCLUSIONS AND FURTHER WORK

In this paper we have presented an automated technique that combines a web harvesting algorithm with a sentiment analysis method to determine the quality of a certain product or service. The data obtained through the web harvesting algorithm is evaluated using the sentiment analysis approach.

Besides the overall proposed solution, we consider important to underline certain unique particularities which we have used in our heuristics for the sentiment analysis and which do not occur in the related approaches [3, 1, 2]. Introducing a neutrality threshold for waiving weakly polarized tuples is an addition that improved the power to discriminate among the sentiment polarities of different reviews. Another difference from the sentiment analysis method based on trigram patterns in [3] is the use of the SentiWordNet lexical resource database for evaluating the polarities of the opinions. Instead, [3] uses the method of pointwise mutual information between two words or phrases, i.e. a statistical measure of word association based on an unsupervised machine learning algorithm. Taking into consideration the average values of the polarity scores of SentiWordNet synsets and their statistical repartition, we concluded that the addition operation is the proper way of computing the polarity value for a tuple. The mentioned method of correcting the part-of-speech tagging output is also something we came up with, putting the SentiWordNet database to a use which it was not designed for.

Finally, the heuristic method defined and employed for the web harvesting system, which exploits the way the data is formatted and structured rather than using the context, is a method we envisioned in order to avoid capturing unwanted and irrelevant data. For our overall method – with both harvesting and sentiment analysis – we attempted to achieve a generic solution. Because of the impossibility of covering all the possible particular cases, we had to employ heuristics in the attempt to obtain good computational performances (i.e. short execution time) at the cost of solution completeness.

The current results and quality measurements are satisfying and the whole method seems to be a good tradeoff between accuracy and completeness. The overall good and accurate sentiment analysis results, combined with the very good results achieved by the web harvesting algorithm, qualify the overall presented solution as a good, reliable and accurate means of generically harvesting and evaluating user-given feedback and reviews.

As already mentioned in section III-E, as future work, in order to improve the identification and extraction of the data of interest we will combine our current web harvesting method based on structure with a semantic context detection method in order to yield better results and accuracy for the web harvesting algorithm. Constructing a domain specific ontology for the domain of interest and using it to determine where relevant data is found will be the solution to follow.

ACKNOWLEDGMENTS

This paper was supported by the Post-Doctoral Programme POSDRU/159/1.5/S/137516, project co-funded from European Social Fund through the Human Resources Sectorial Operational Program 2007-2013.

REFERENCES

[1] A. Esuli and F. Sebastiani, "SentiWordNet: A publicly available lexical resource for opinion mining", *5th Conference on Language Resources and Evaluation (LREC '06)*, pp. 417-422, Genova, IT, 2006.
[2] S. Baccianella, A. Esuli, and F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining", *7th Conference on language Resources and Evaluation (LREC 2010)*, pp. 2200-2204, 2010.
[3] P.D. Turney, "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews", *the 40th Annual Meeting of the Association for Computational Linguistics (ACL '02)*, pp. 417-424, 2002.
[4] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "Introduction to WordNet: an on-line lexical database", *International Journal of Lexicography*, vol. 3, no. 4, pp. 235-244, 1990.
[5] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of Twitter data", *Workshop on Languages in Social Media*, pp. 30-38, 2011.
[6] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining", *7th International Conference on Language Resources and Evaluation (LREC2010)*, pp. 1320-1326, 2010.
[7] K. Shah, N. Munshi, and P. Reddy, "Sentiment analysis and opinion mining of microblogs", *University of Illinois at Chicago, Course CS 583 - Data Mining and Text Mining*, 2013.
[8] B. Liu, M. Hu, and J. Cheng, "Opinion Observer: analyzing and comparing opinions on the Web", *14th international conference on World Wide Web*, pp. 342-351, 2005.
[9] G. Qiu, B. Liu, J. Bu, and C. Chen, "Opinion word expansion and target extraction through double propagation", *Computational Linguistics*, vol. 37, no. 1, pp. 9-27, 2011.
[10] Lingua-EN-Tagger *http://search.cpan.org/~acoburn/Lingua-EN-Tagger-0.24/Tagger.pm*, 2014.
[11] User Agent *http://tools.ietf.org/html/rfc3261*, 2014.
[12] HTTP Header Fields *http://tools.ietf.org/html/rfc4229*, 2014.