

Automatically Detecting and Rating Product Aspects from Textual Customer Reviews

Wouter Bancken, Daniele Alfarone and Jesse Davis

Department of Computer Science, KU Leuven
Celestijnenlaan 200A - box 2402, 3001 Leuven, Belgium
wouter.bancken@student.kuleuven.be
daniele.alfarone@cs.kuleuven.be
jesse.davis@cs.kuleuven.be

Abstract. This paper proposes a new approach to aspect-based sentiment analysis. The goal of our algorithm is to obtain a summary of the most positive and the most negative aspects of a specific product, given a collection of free-text customer reviews. Our approach starts by matching handcrafted dependency paths in individual sentences to find opinions expressed towards candidate aspects. Then, it clusters together different mentions of the same aspect by using a WordNet-based similarity measure. Finally, it computes a sentiment score for each aspect, which represents the overall emerging opinion of a group of customers towards a specific aspect of the product. Our approach does not require any seed word or domain-specific knowledge, as it only employs an off-the-shelf sentiment lexicon. We discuss encouraging preliminary results in detecting and rating aspects from on-line reviews of movies and MP3 players.

Keywords: aspect-based sentiment analysis, opinion mining, syntactic dependency paths, text mining

1 Introduction

Sentiment analysis is the task of detecting subjectivity in natural language. Approaches to this task mainly draw from the areas of natural language processing, data mining, and machine learning. In the last decade, the exponential growth of opinionated data on the Web fostered a strong interest in the insights that sentiment analysis could reveal. For example, companies can analyze user reviews on the Web to obtain a good picture of the general public opinion on their products at very little cost.

While the first efforts in sentiment analysis were directed towards determining the general polarity (positive or negative) of a certain sentence or document, the interest has recently shifted towards a more qualitative analysis, that aims to detect the different *aspects* of a topic towards which an opinion is expressed. For example, we may be interested in analyzing a movie review to capture the opinions of the reviewer towards aspects such as the plot, the cinematography,

or the performance of a specific actor. The most challenging part in aspect-based sentiment analysis is that a system needs to detect the relevant aspects before these can be associated with a polarity.

In this paper we introduce ASPECTATOR, a new algorithm for automatically detecting and rating product aspects from customer reviews. ASPECTATOR can discover candidate aspects by simply matching few syntactic dependency paths, while other approaches [6, 14, 16, 21] require seed words in input and use syntactic dependencies or some bootstrapping technique to discover new words and the relations between them. Additionally, it does not require any domain-specific knowledge in input, but only few handcrafted syntactic dependency paths and an off-the-shelf sentiment lexicon. Consequently, the proposed system can detect and rate aspects of products in any domain, while many existing approaches [16, 21, 18] focus on domains for which machine-readable knowledge is available. Concretely, ASPECTATOR combines a first high-recall step where candidate aspects are extracted from individual sentences through syntactic dependency paths, with a second and third high-precision steps, where aspect mentions are clustered and their sentiment scores are aggregated by leveraging an external sentiment lexicon.

In our opinion, the considered setting represents an ideal testbed for investigating interactions between natural language processing and data mining. Indeed, our focus is not on extracting the aspects discussed in a single sentence or document, which could be seen as a problem of deep text understanding, but on crunching hundreds of reviews of a specific product to capture the aspects that best summarize the opinions of a group of customers, which requires linguistic knowledge to extract information from single sentences, along with data mining expertise to make sense of large amounts of data.

2 Related Work

Historically, sentiment analysis has been concerned with assigning a binary classification to sentences or entire documents, that represents the polarity (i.e., the orientation) of the writer towards the discussed contents [13, 19]. Nevertheless, the overall polarity gives no indication about which *aspects* the opinions refer to. For this reason, in 2004 Hu and Liu [6] introduced the more interesting problem of aspect-based sentiment analysis, where polarity is not assigned to sentences or documents, but to single aspects discussed in them. In their approach, given a large number of reviews for a specific product, they first attempt to identify interesting product aspects by using association mining, and then attach a sentiment score to each aspect by exploiting a small seed set of opinion words, along with their synonyms and antonyms present in WordNet. Next, they use newly detected opinion words to extract additional infrequent product aspects. Instead of using association mining, our work will detect aspects through dependency paths, and will use an external sentiment lexicon to rate them. However, their work remains the most similar to ours, as in both cases the goal is to summarize a

collection of reviews for a specific product by detecting the most interesting and discussed aspects, while most approaches focus on analyzing individual reviews.

Qiu et al. [14] continued to pursue the idea that opinion words can be used to detect product aspects and vice versa, focusing on single reviews. In their approach, a seed set of opinion words is combined with syntactic dependencies to identify product aspects and new opinion words. To detect the polarity of the newly identified opinion words, they consider the given polarities of the seed words and make the assumption that opinion words expressing a sentiment towards the same aspect in the same review share the same polarity. While Qiu et al. use syntactic dependencies solely to capture word sequences that contain aspects or opinion words already observed, our approach uses dependency paths to detect new product aspects, with the potential advantage of achieving higher coverage.

A different line of work on aspect-based sentiment analysis is based on topic models. Brody and Elhadad [3] have tried to use Latent Dirichlet Allocation (LDA) [2] to extract topics as product aspects. To determine the polarity towards each topic/aspect, they start from a set of seed opinion words and propagate their polarities to other adjectives by using a label propagation algorithm. Instead of treating aspect detection and sentiment classification as two separate problems, Lin and He [11] and Jo and Oh [8] directly integrate the sentiment classification in the LDA model, so that it natively captures the sentiment towards the topic/aspect. While these LDA-based approaches provide an elegant model of the problem, they produce topics that are often not directly interpretable as aspects, and thus require manual labelling to achieve a readable output.

The work discussed so far proposes domain-independent solutions for aspect-based sentiment analysis, where also our approach is positioned. However, several works make use of domain-specific knowledge to improve their results. For instance, Thet et al. [16] focus on aspect-based classification of movie reviews, and include as input for their algorithm movie-specific terms such as the name of the movie, the cast and the director. Additionally, they include some domain-specific opinion words as input for their algorithm. As expected, including domain-specific knowledge yields a more accurate sentiment classification. To make an example, the word “*unpredictable*” has a negative polarity in general English, but in the movie domain it is often used to praise the unpredictability of a storyline. Since all relevant aspects are given as input, they exclusively focus on detecting opinions towards the given aspects by (1) capturing new opinion words through syntactic dependencies, and (2) rating the product aspects based on an external sentiment lexicon and some given domain-specific opinion words.

Similarly, Zhu et al. [21] use product aspects and some aspect-related terms as input for their algorithm, but then attempt to discover new aspect-related terms by applying a bootstrapping algorithm based on co-occurrence between seed terms and new candidate terms. A sentiment score is again obtained by accessing an external sentiment lexicon. While our approach retains from these works the usage of an external lexicon, it requires neither labelled examples nor domain-specific knowledge, thus it has wider applicability.

3 Aspectator: a New Approach

ASPECTATOR takes as input a collection of textual customer reviews for one specific product, and automatically extracts the most positive and the most negative aspects of the product, together with all the sentences that contribute to the sentiment polarity of each aspect. More precisely:

Given: a collection of textual reviews of one specific product

Extract:

- The n most *positive* product aspects, along with a list of all sentences containing positive and negative mentions of each aspect.
- The n most *negative* product aspects, along with a list of all sentences containing positive and negative mentions of each aspect.

ASPECTATOR works in three steps, depicted in Fig. 1. First, it detects mentions of aspects and their associated opinion by matching handcrafted paths in dependency trees. Second, it clusters the different mentions of an aspect extracted in the first step by means of a WordNet-based similarity measure. Third, it attaches a sentiment score to each mention, and aggregates the scores from all mentions belonging to the same cluster in order to obtain a final sentiment score for each aspect.

ASPECTATOR does not require labelled examples and it is domain-independent, thus it can run on any collection of reviews for a specific product. The only required external knowledge is in the form of ten handcrafted dependency paths and an English lexicon with a sentiment score for every word.

3.1 Detecting Product Aspects

The objective of the first step is to extract from customer reviews mentions of a product aspect and the words that express the opinion of the writer towards that aspect. For instance, given the sentence:

“The action music used in the movie wasn’t too good.”

ASPECTATOR extracts the following pair:

$$\langle \underbrace{\text{not too good}}_{\substack{\text{Sentiment} \\ \text{modifier}}} ; \underbrace{\text{action music}}_{\substack{\text{Aspect} \\ \text{mention}}} \rangle$$

We call this an **opinion pair**, as the first part is the opinion of a reviewer towards the second part. The first part can optionally be negated, as in the above example, causing an inversion of the polarity expressed by the sentiment modifier.

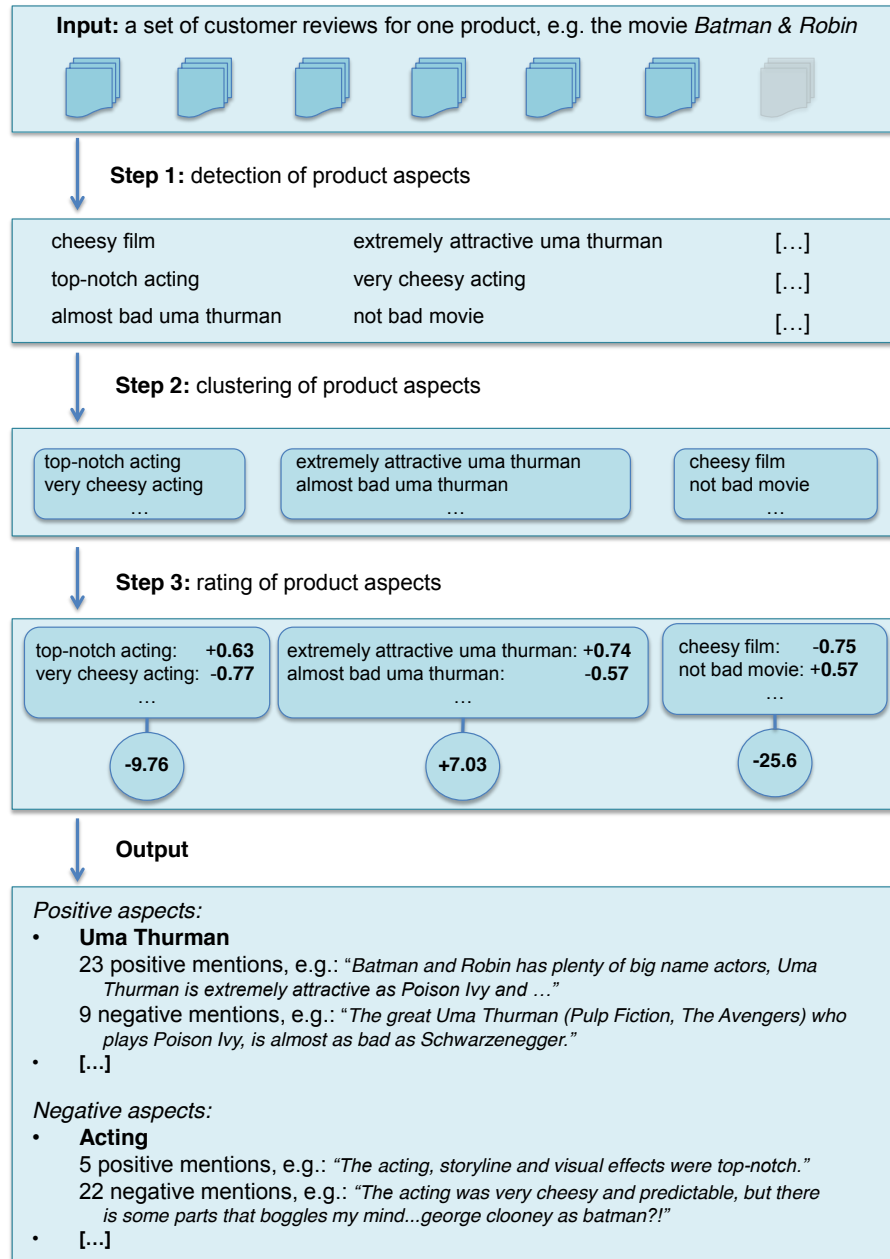


Fig. 1. ASPECTATOR's full pipeline, with example extractions from reviews for the movie *Batman & Robin*. Scores greater or lower than zero represent positive or negative sentiment polarity, respectively.

ASPECTATOR extracts opinion pairs by using ten simple handcrafted dependency paths, in three steps:

1. For each sentence, ASPECTATOR extracts a syntactic dependency tree by using the Stanford dependency parser [4, 10]. Fig. 2 shows the dependencies for the example sentence above.
2. Given a dependency tree, it attempts to extract a basic opinion pair composed by a single-word sentiment modifier and a single-word aspect mention by matching one of the five dependency paths shown in Table 1. For the example sentence, this step extracts the opinion pair $\langle \textit{good} ; \textit{music} \rangle$ through the dependency path $A \xleftarrow{\textit{nsubj}} M \xrightarrow{\textit{cop}} *$.
3. Given a matched opinion pair, it attempts to extend the match to neighbouring words by applying the additional dependency paths shown in Table 2. This allows to (1) capture multi-word expressions, such as “*action music*” and “*too good*” in the running example, and (2) capture negations, such as “*wasn't*” in the example. The final opinion pair for the running example becomes $\langle \textit{not too good} ; \textit{action music} \rangle$.

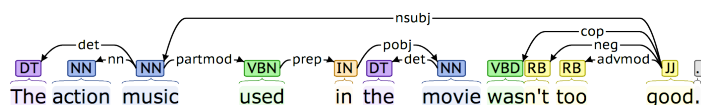


Fig. 2. Example of syntactic dependencies detected by the Stanford dependency parser.

Note that our approach leverages syntactic dependency paths for two purposes: (1) detecting aspect mentions and sentiment modifiers, and (2) discovering relations between them. This is a significant difference with other approaches that are based on syntactic dependencies. For example, Qiu et al. [14] only use syntactic dependencies to identify relations between word sequences that contain an aspect or an opinion word that has been detected before.

While our technique for extracting aspect mentions and sentiment modifiers yields high recall, its precision is low, since several irrelevant word sequences are captured. Nevertheless, the following steps allow our system to assign lower confidence to incorrect extractions, thus ultimately yielding accurate top-ranked extractions.

Table 1. Main dependency paths used by ASPECTATOR to detect an aspect (A) and a sentiment modifier (M) that form an opinion pair $\langle M; A \rangle$. Asterisks (*) are wildcards that can match any word.

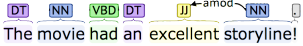
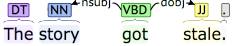
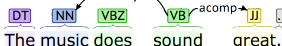
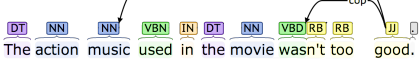
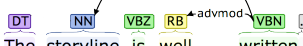
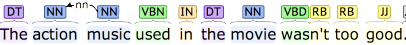
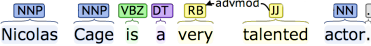
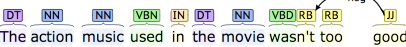
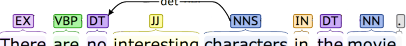
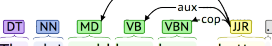
Type	Dependency path	Example
Adjectival modifier	$M \xleftarrow{amod} A$	
Direct object	$A \xleftarrow{nsubj} * \xrightarrow{dobj} M$	
Adjectival complement	$A \xleftarrow{nsubj} * \xrightarrow{acomp} M$	
Complement of a copular verb	$A \xleftarrow{nsubj} M \xrightarrow{cop} *$	
Adverbial modifier to a passive verb	$A \xleftarrow{nsubjpass} * \xrightarrow{advmod} M$	

Table 2. Extensions to the dependency paths of Table 1 to deal with multi-word aspects (A) and multi-word sentiment modifiers (M), and to capture negations. Note that the fourth extension is the only one that imposes a lexical constraint, as it only triggers if the determiner is the word “no”.

Type of extension	Dependency path	Example
Compound noun	$A' \xleftarrow{nn} A$	
Adverbial modifier	$M' \xleftarrow{advmod} M$	
Simple negation	$* \xleftarrow{neg} M$	
Negation through “no” determiner	“no” $\xleftarrow{det} A$	
Negation through hypothetical phrase	$* \xleftarrow{aux} M \xrightarrow{cop} *$	

3.2 Clustering Product Aspects

The goal of this step is to cluster the previously-extracted opinion pairs by searching for all semantically similar aspect mentions, independently from their sentiment modifier. For example, in the context of movie reviews, we would like to cluster together the opinion pairs $\langle \textit{very bad}; \textit{music} \rangle$ and $\langle \textit{awesome}; \textit{soundtrack} \rangle$, as they both express opinions towards the same aspect of a movie.

To identify semantically similar aspect mentions, ASPECTATOR uses a WordNet-based similarity metric called *Jcn* [7]. Zhai et al. [20] experimented with several WordNet-based similarity metrics in the context of clustering for aspect-based sentiment analysis, concluding that *Jcn* delivers the best results.

Jcn is based on the principle that two terms are similar if their least common subsumer (LCS) in the WordNet taxonomy has high information content (IC). For instance, the terms $(\textit{car}, \textit{bicycle})$, having LCS “*vehicle*”, are more similar than $(\textit{car}, \textit{fork})$, having LCS “*artifact*”, because “*vehicle*” is a more informative term than “*artifact*”. Formally, the *Jcn* similarity between two terms t_1 and t_2 is defined as:

$$Jcn(t_1, t_2) = \frac{1}{IC(t_1) + IC(t_2) - 2 \cdot IC(LCS(t_1, t_2))} \quad (1)$$

where $LCS(t_1, t_2)$ is the least common subsumer of t_1 and t_2 in WordNet, and the information content of a term is equivalent to:

$$IC(t) = -\log P(t) \quad (2)$$

where $P(t)$ is the probability of observing, in a large English corpus, the term t or any term subsumed by t in the WordNet hierarchy. The higher the probability of observing a term t or any of its subsumed terms, the lower the information content of t .

Concretely, in order to cluster opinion pairs, ASPECTATOR first computes the *Jcn* similarity for every possible pair of aspect mentions, by using an implementation available in the WS4J library [15]. Next, it normalizes all mentions by stemming them, in order to increase data density. When two terms map to the same root, for instance “*act*” and “*acting*”, a comparison with another term is made by picking the stem that maximizes the *Jcn* similarity. Finally, ASPECTATOR uses the pairwise similarity values as input for the K-Medoids clustering algorithm [9], which will return clusters of opinion pairs, with each cluster representing a collection of opinions towards a single aspect. K-Medoids is preferred over K-Means because it can compute the centroid of a cluster without the need of defining a mean.

3.3 Rating Product Aspects

In the final stage of our approach, each cluster receives a sentiment score, which represents the overall emerging opinion of a group of customers towards a specific aspect of a product. Concretely, ASPECTATOR undertakes three sub-steps for each cluster:

1. For each opinion pair in the cluster, it assigns an individual sentiment score to each word that composes the sentiment modifier. For instance, given the opinion pair $\langle \textit{just plain stupid}; \textit{action music} \rangle$, it attaches three individual scores to “*just*”, “*plain*” and “*stupid*”.
2. It combines the scores for the individual words into a single score for the entire sentiment modifier, e.g., “*just plain stupid*”.
3. It extracts a final sentiment score for the entire cluster by aggregating the scores of all sentiment modifiers.

Step 1. In order to obtain a sentiment score for individual words, ASPECTATOR uses the external sentiment lexicon SentiWordNet [1]. SentiWordNet extends WordNet by attaching three scores to each *synset*:¹ a positive sentiment score, a negative sentiment score and a neutrality score. These three scores always sum to 1. For example, the word “*mediocre*”, in the sense of “lacking exceptional quality or ability” has the scores 0.25, 0.125 and 0.625 as its positive, neutral and negative score, respectively.

For simplicity, our approach does not use three different sentiment scores, but combines them in one score in the range $[-1,1]$ by subtracting the negative score from the positive score. The neutrality score is thus ignored, as “almost neutral” opinions will have a score close to zero, and consequently will have no significant impact in the following aggregation steps. Instead of performing word sense disambiguation, ASPECTATOR simply aggregates the sentiment scores of all the synsets in which a word w appears, as follows:

$$\textit{score}(w) = \frac{\sum_{i=1}^n \textit{score}(\textit{synset}_i)/i}{\sum_{i=1}^n 1/i} \quad (3)$$

where $i \in \mathbb{N}$ is the rank of a synset in WordNet based on the synset’s frequency in general English, and \textit{synset}_i is the i^{th} synset of w in the ranking. Intuitively, dividing a synset’s score by i allows our approach to give higher weight to synsets that are more likely to represent the right sense of the word w in a certain context, given their overall higher popularity in English.

Step 2. The word-level scores obtained in the previous step are then combined into a single score for the entire sentiment modifier by adopting an approach based on the work of Thet et al. [16]. Specifically, ASPECTATOR takes as initial score the sentiment score of the rightmost (i.e., most specific) word in the sentiment modifier. Then, it iteratively uses the score of each preceding word to either intensify or attenuate the current score depending on the polarity of the considered words, remaining in the range $[-1,1]$. Concretely, the score for a sentiment modifier composed by words $w_n w_{n-1} \dots w_1 w_0$ is computed as:

¹ A synset is a group of synonymous words, corresponding to a node in the WordNet hierarchy.

$$\begin{aligned} \text{score}(w_i \dots w_0) &= \text{score}(w_{i-1} \dots w_0) - (\text{score}(w_{i-1} \dots w_0) \cdot |\text{score}(w_i)|) \\ &\quad \text{if } \text{score}(w_{i-1} \dots w_0) > 0 \text{ and } \text{score}(w_i) < 0 \end{aligned} \quad (4a)$$

$$\begin{aligned} \text{score}(w_i \dots w_0) &= \sigma \cdot (|\text{score}(w_{i-1} \dots w_0)| + (1 - |\text{score}(w_{i-1} \dots w_0)|) \cdot |\text{score}(w_i)|) \\ &\quad \text{with } \sigma = \text{sign}(\text{score}(w_{i-1} \dots w_0)) \\ &\quad \text{otherwise} \end{aligned} \quad (4b)$$

In case the sentiment modifier is negated, the resulting sentiment score is multiplied by -1 to obtain the opposite polarity.

Equation (4b) models the general case, where the next word w_i in the iterative procedure intensifies the current score functionally to $|\text{score}(w_i)|$. This follows Thet et al.’s observation that (1) words with the same polarity tend to intensify each other (e.g., “*super nice*”, “*terribly stupid*”), and (2) a negative current score becomes more negative when the next word has positive score (e.g., “*super bad*”). Equation (4a) is introduced to handle the particular case in which the current score is positive and the next word to be processed is negative (e.g., “*hardly interesting*”). In this case, applying (4b) would make the final score more positive, while a negative modifier should make the score less positive.

As a full example, we show how our iterative procedure computes a sentiment score for the opinion pair $\langle \textit{just plain stupid} ; \textit{action music} \rangle$:

Example opinion pair: $\langle \textit{just plain stupid} ; \textit{action music} \rangle$

Individual scores: $\begin{matrix} w_2 & w_1 & w_0 \\ 0.07 & 0.12 & -0.51 \end{matrix}$

$$\text{score}(\textit{plain stupid}) = (-1) \cdot (0.51 + (1 - 0.51) \cdot 0.12) = -0.57$$

$$\text{score}(\textit{just plain stupid}) = (-1) \cdot (0.57 + (1 - 0.57) \cdot 0.07) = -0.60$$

Thus, the resulting sentiment score for the aspect mention “*action music*” in this example is -0.60 .

Step 3. Lastly, ASPECTATOR computes a final sentiment score for each aspect, by summing the scores computed in the previous step for all sentiment modifiers belonging to the aspect’s cluster. A simple algebraic summation supports the intuition that few strongly positive/negative opinions should result in a sentiment score comparable to the one of many weakly positive/negative opinions. We refer back to Fig. 1 for a complete example.

In order to produce the final output, ASPECTATOR ranks the aspects by their sentiment score, and returns only the n most positive and the n most negative aspects, where n is specified by the user. This ranking places at the top the most interesting aspects, i.e., the ones that (1) are frequently mentioned in the reviews, and (2) are subjected to strong positive or negative opinions of the reviewers.

This has also the advantage that many incorrect opinion pairs extracted in the first step of the pipeline (Sect. 3.1) will be excluded from the final output, as they typically have very few mentions and are not associated with strong opinions.

4 Experiments

In this section, we present a preliminary evaluation of ASPECTATOR. The objective of our experiments is to address the following questions:

1. Can our approach detect interesting and relevant product aspects?
2. Can our approach provide meaningful evidence that supports the sentiment score assigned to each aspect?

Additionally, we discuss the main sources of error of our approach.

4.1 Methodology

ASPECTATOR’s output was manually evaluated on a portion of two public datasets from different domains by two annotators, out of which only one was a co-author.

The first dataset is a collection of movie reviews taken from Amazon,² published by McAuley and Leskovec [12]. Since manual evaluation is required, we sampled ten movies to create a validation set and a test set, in the following way. First, we considered only the 50 movies with the highest number of reviews, as we want to test the ability of our algorithm to summarize a large amount of data for a single movie. Since most of these movies have a majority of positive reviews, in order to obtain a more balanced dataset we first took the five movies with the highest number of negative reviews, and then randomly sampled five other movies from the remaining set. This resulted in a collection of 700 to 850 reviews for each movie.

The second dataset consists of reviews of MP3 players taken from Amazon,³ published by Wang, Lu and Zhai [17]. From this dataset we selected the five products with the highest number of reviews in the dataset, obtaining a collection of 500 to 770 reviews for each MP3 player.

From these samples, we used eight movies and three MP3 players as our validation set, and the remaining two movies and two MP3 players as our test set. We used the validation set to determine the optimal k for the K-Medoids clustering applied in Sect. 3.2, which should ideally be equal to the total number of unique aspects appearing in a set of reviews. We found that the optimal k is 0.9 times the number of aspect mentions to be clustered. For instance, if 1700 aspect mentions have been identified for a certain product, we set $k = 1530$.

We used the test set consisting of two movies and two MP3 players to manually evaluate our algorithm. For each product, two annotators were given a form with the ten most positive and the ten most negative product aspects, along

² <http://snap.stanford.edu/data/web-Movies.html>

³ <http://sifaka.cs.uiuc.edu/~wang296/Data/LARA/Amazon/mp3/>

with the six sentences containing the three most positive and three most negative mentions of each aspect. The annotators were asked to mark each product aspect and each sentence mentioning the aspect as either correct or incorrect. For simplicity, in the form given to the annotators each aspect was only represented by the aspect mention appearing most frequently in the reviews. An aspect is considered correct if it is an interesting and relevant aspect for the considered product, such as “*battery life*” and “*display*” for an MP3 player, or “*storyline*” and the name of an actor for a movie. A sentence listed by our algorithm for a certain aspect is considered correct only if (1) the sentence mentions the considered aspect, and (2) the sentence expresses an opinion towards the considered aspect that matches the polarity extracted by our algorithm for that specific opinion pair.

4.2 Results

The accuracy of the top- n aspects is shown in Fig. 3. On average, the 10 most positive and the 10 most negative aspects were considered to be correct in 72.5% of the cases. The sentences mentioning an aspect were only considered correct in 59.8% of the cases. However, this last result can be studied more closely. Table 3 shows the accuracy of these sentences in function of the polarity of both aspects and sentences. Clearly, the detected sentences are generally more accurate when the aspect and the corresponding sentence have the same polarity. This is due to the fact that for an aspect there are typically many more sentences with a matching polarity than sentences with the opposite polarity, so when the top-3 sentences are drawn from a larger number of sentences, these tend to have higher accuracy.

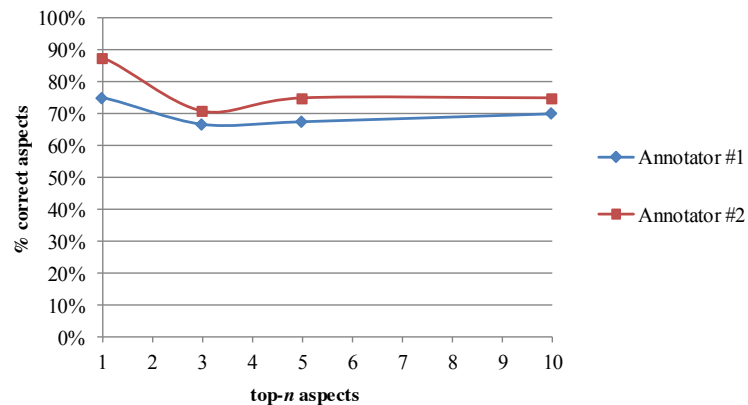


Fig. 3. Percentage of top-1, top-3, top-5, top-10 aspects marked as correct by two annotators.

Table 3. Percentages of correct sentences split accordingly to the polarity of the sentence and the product aspect.

	Pos. sentence pos. aspect	Neg. sentence pos. aspect	Neg. sentence neg. aspect	Pos. sentence neg. aspect
Annotator #1	86.9%	47.8%	66.7%	39.1%
Annotator #2	85.0%	53.0%	58.3%	41.3%
Average	85.9%	50.4%	62.5%	40.2%

Table 4. Positive and negative aspects for the movie *Batman & Robin*. Each aspect is represented as a cluster of its different mentions. Aspects shown in italics were marked as incorrect by the two annotators, while † indicates a disagreement between annotators.

Batman & Robin		
	Positive aspects	Negative aspects
1	<i>book</i>	movie, film, ...
2	Uma Thurman	dialogue, dialog
3	<i>job, occupation, ...</i>	† line
4	actor, thespian	acting, act, ...
5	Alicium Silverstone	† review, reviewer
6	performance, perform, ...	plot
7	Bruce Wayne	<i>guy</i>
8	effect, consequence, ...	script
9	costume	<i>thing</i>
10	<i>way</i>	character, depict, ...

When evaluating the product aspects, the annotators agreed in 88% of the cases, with an inter-annotator agreement of $\kappa = 0.69$ according to Cohen’s kappa score. When evaluating the sentences containing mentions, the annotators agreed in 89% of the cases, with $\kappa = 0.785$. Table 4 shows an example of the aspects manually evaluated for the movie “*Batman & Robin*”. A sentence marked as correct positive mention for the aspect “*performance*” in the same movie is:

“Though not classic villains, Arnold Schwarzenegger as Mr. Freeze and Uma Thurman as Poison Ivy give engaging performances.”

while for the same aspect a negative mention is:

“The performances were horrendous, roll call: George Clooney, Chris O’Donnel, Alicia Silverstone and Arnold.”

4.3 Error Analysis

We conclude the experimental section by reporting the main sources of errors for our approach.

Ambiguity of word polarity. The polarity of an opinion word can vary according to its context. In some cases, SentiWordNet does not cover all possible senses of a word. For instance, SentiWordNet only contains positive sentiment scores for the word “*laughable*”, while in a sentence such as “*The acting is laughable.*” the intended sentiment is clearly negative.

In some other cases, even though SentiWordNet covers also the correct sense of a word, ASPECTATOR picks the wrong polarity. This is due to the fact that, for simplicity, our algorithm does not perform word sense disambiguation, but instead computes a sentiment score for a term as a weighted sum of the scores of all possible senses of the term. For example, SentiWordNet contains several positive sentiment scores for different senses of the word “*joke*”, and only one negative score. By using a weighted sum, the overall sentiment score is positive, while in a sentence such as “*The dialogue was just a joke.*” the word is used with a negative connotation.

Inclusion of objective statements. The presence of opinion words in a review does not necessarily imply that the reviewer is expressing an opinion. For instance, when describing the antagonist of a movie, reviewers often use words with a negative connotation without the intention of expressing any judgement. This is the case in sentences like “*Javier Bardem is an extremely creepy bad guy.*”, where ASPECTATOR incorrectly concludes that a strongly negative opinion is expressed towards Javier Bardem.

Limitations of dependency paths. The ten handcrafted dependency paths sometimes fail to capture the full sentiment of a sentence. To make a concrete example, consider the sentence “*Uma Thurman was really sexy as Poison Ivy... and that’s about it.*”. If the first part of the sentence was considered in isolation, a human reader would interpret it as a positive opinion about Uma Thurman, and ASPECTATOR does the same. Nevertheless, the second part of the sentence reveals a negative attitude of the reviewer, which our simple dependency paths cannot capture.

Incorrect dependency parsing. A last major source of error is introduced by the Stanford dependency parser. Some of the errors are caused by the inability of the Stanford parser to deal with the imprecise, colloquial language typically adopted in on-line product reviews. To make an example, from the sentence “*Man, this film is bad.*” the parser indicates that the opinion word “*bad*” refers to “*man*”, and not “*film*”.

Additionally, the Stanford parser is not always able to detect compound nouns, as terms like “*hard drive*” are considered to be adjective-noun pairs. This makes ASPECTATOR interpret the adjective as an opinion expressed towards the

noun, while the compound noun simply represents an aspect mention with no associated opinion.

5 Conclusions

We presented ASPECTATOR, a novel algorithm for aspect-based sentiment analysis that takes in input a collection of customer reviews for a specific product, and automatically extracts the most positive and the most negative aspects, together with evidence that supports the extractions. ASPECTATOR first harvests candidate aspects and the associated opinions by matching ten simple hand-crafted dependency paths, then clusters together mentions of the same aspect, and finally computes a sentiment score that expresses the overall orientation towards each aspect. Our approach is domain-independent and does not require any labelled example, thus it can be adopted to analyze customer reviews for products in unseen domains.

In a preliminary evaluation, we show that on average the 72.5% of the extracted aspects are relevant, and that sentences that adhere to the overall polarity of each aspect are correct in 74.2% of the cases. This percentage drops to 45.3% when the sentence polarity does not match the overall aspect polarity. Furthermore, we found that most errors in our pipeline are caused by the ambiguity and the complexity of the colloquial language adopted in the reviews.

For future work, we are interested in verifying whether, starting from few example opinion pairs, we can learn the dependency paths that we now hand-craft, and discover additional ones that generalize well across multiple domains. Additionally, an extended (and comparative) evaluation is required.

Acknowledgements

Daniele Alfarone and Jesse Davis acknowledge the generous support of the Research Fund K.U. Leuven (CREA/11/015 and OT/11/051), EU FP7 Marie Curie Career Integration Grant (#294068), and FWO-Vlaanderen (G.0356.12).

References

1. Baccianella S., Esuli A., Sebastiani F., Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10, pages 2200–2204, 2010.
2. Blei D. M., Ng A. Y., Jordan M. I., Latent dirichlet allocation. *Journal of Machine Learning research*, 3:993–1022, 2003.
3. Brody S., Elhadad N., An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics, 2010.
4. De Marneffe M.-C., MacCartney B., Manning C. D., Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

5. Ganu G., Elhadad N., Marian A., Beyond the stars: Improving rating predictions using review text content. In *Proceedings of the 12th International Workshop on the Web and Databases*, 2009.
6. Hu M., Liu B., Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–177. ACM, 2004.
7. Jiang J. J., Conrath D. W., Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, 1997.
8. Jo Y., Oh A. H., Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM, 2011.
9. Kaufman L., Rousseeuw P. J., Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, 2009.
10. Klein D., Manning C. D., Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, 2003.
11. Lin C., He Y., Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 375–384, 2009.
12. McAuley J. J., Leskovec J., From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 897–908, 2013.
13. Paltoglou G., Thelwall M., A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395, 2010.
14. Qiu G., Liu B., Bu J., Chen C., Expanding domain sentiment lexicon through double propagation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, volume 9, pages 1199–1204, 2009.
15. Shima H., WS4J WordNet Similarity for Java. <https://code.google.com/p/ws4j/>, 2014.
16. Thet T. T., Na J.-C., Khoo C. S., Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 36(6):823–848, 2010.
17. Wang H., Lu Y., Zhai C., Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 618–626. ACM, 2011.
18. Wang H., Lu Y., Zhai C., Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2010.
19. Yessenalina A., Yue Y., Cardie C., Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing*, pages 1046–1056, 2010.
20. Zhai Z., Liu B., Xu H., Jia P., Clustering product features for opinion mining. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 347–354. ACM, 2011.
21. Zhu J., Wang H., Zhu M., Tsou B. K., Ma M., Aspect-based opinion polling from customer reviews. *IEEE Transactions on Affective Computing*, 2(1):37–49, 2011.
22. Zhu X., Ghahramani Z., Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.