SOBA: Semi-Automated Ontology Builder for Aspect-Based Sentiment Analysis

Lisa Zhuang, Kim Schouten, Flavius Frasincar*

Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR, Rotterdam, The Netherlands

Abstract

This research explores the possibility of improving knowledge-driven aspect-based sentiment analysis (ABSA) in terms of efficiency and effectiveness. This is done by implementing a Semi-automated Ontology Builder for Aspect-based sentiment analysis (SOBA). Semi-automatization of the ontology building process could produce more extensive ontologies, whilst shortening the building time. Furthermore, SOBA aims to improve the effectiveness of its ontologies in ABSA by attaching to concepts the semantics provided by a semantic lexicon. To evaluate the performance of SOBA, ontologies are created using the ontology builder for the restaurant and laptop domains. The use of these ontologies is then compared with the use of manually constructed ontologies in a state-of-the-art knowledge-driven ABSA model, the Two-Stage Hybrid Model (TSHM). The results show that it is difficult for a machine to beat the quality of a human made ontology, as SOBA does not improve the effectiveness of TSHM, achieving similar results. Including the semantics provided by a semantic lexicon in general increases the performance of TSHM, albeit not significantly. However, SOBA decreases by 50% or more the human time needed to build ontologies, so that it is recommended to use SOBA for knowledge-driven ABSA frameworks, as it leads to greater efficiency.

Keywords: domain ontology, aspect-based sentiment analysis, ontology learning, reviews, semi-automatization

1. Introduction

In the age of digitalization, an increasing amount of people share their opinions on the Internet, for example in online reviews. These reviews are invaluable for companies that provide products or services, as they offer suggestions for improvements. These online reviews are also widely used by customers with 73% to 87% of the online review readers reporting that their choice of purchase significantly dependents on reviews [1].

However, manually analyzing these reviews is time-consuming and it is difficult to summarize and report all the information effectively. Instead, sentiment analysis can be applied. Sentiment analysis aims to extract the writer's sentiment from a text, and to report this in a comprehensible and coherent way [2]. This can either be done on a review-level, where the sentiment is determined for the whole text, or on a sentence-level, so that the sentiment is calculated per sentence.

To gain even more detailed insight into people's opinions expressed in texts, one could use aspect-based sentiment analysis (ABSA), a sub-field of sentiment analysis [3]. Whereas sentiment analysis determines the sentiment for a whole review or sentence, ABSA focuses on detecting the reviewed aspects in a text and computing the sentiment for each aspect that is mentioned. This is useful as people often mention various aspects of a product or service in a single review, with each aspect receiving its own rating. Using ABSA, firms can gain better insight into the specific strengths and weaknesses of their products or services. This is more informative than an overall impression, as more detailed information enables companies to target specific points of improvement.

Numerous studies on sentiment analysis can be found in the literature with many focusing mainly

*Corresponding author; tel: +31 (0)10 408 1340; fax: +31 (0)10 408 9162

Email addresses: lisa.zhuang@hotmail.com (Lisa Zhuang), schouten@ese.eur.nl (Kim Schouten), frasincar@ese.eur.nl (Flavius Frasincar) on machine learning-based sentiment analysis, as various machine learning algorithms have shown to be successful in this domain [4, 5, 6]. However, machine learning-based models also have a downside, as they require a significant amount of training data in order to perform well.

Consequently, other researchers investigated the use of completely knowledge-based sentiment analysis [7], or models that combine knowledge-based and machine learning-based sentiment analysis [8, 9]. For example, [9] presents a Two-Stage Hybrid Model (TSHM), which contains a knowledge repository as well as a machine-learning component. Making use of the complementary characteristics of knowledge-based and machine learning-based methods, the hybrid model shows state-of-the-art results for ABSA.

A major disadvantage of knowledge-driven models is that manually constructing a knowledge repository such as an ontology is a slow and difficult process. To solve this problem, recent research has been conducted on (semi-)automated ontology builders, which decrease the human effort needed to build an ontology, while creating richer knowledge repositories [10, 11, 12].

Another issue with knowledge-based models is that often only the lexical representation of words is taken into account, and the semantic definition of concepts and words is overlooked. As a result, words from texts might be falsely matched with concepts from the knowledge repository when their lexicalizations correspond, despite different underlying semantics. Therefore, [12] stresses the importance of word sense disambiguation (WSD) in building knowledge repositories.

This research presents a Semi-automated Ontology Builder for Aspect-based sentiment analysis (SOBA), of which the source code can be found at `https://github.com/lisazhuang/SOBA`. We examine the implementation of a semi-automated ontology builder, rather than a fully automated one, meaning that human input is required to control for possible mistakes made by the ontology builder. We design our ontology builder so that the semi-automatically built ontologies follow the ontology structure of [9]. This allows us to examine the performance of SOBA ontologies in the knowledge-driven aspect-based sentiment analysis framework of [9], the TSHM, a well-performing ABSA approach. Moreover, our ontology builder includes information on the semantic meaning of concepts, as defined by WordNet [13], in its ontologies. We adjust TSHM accordingly, so that it makes use of the semantic meaning of the concepts when analyzing reviews, and we examine how this affects the performance of the TSHM. Based on this, we define following research question:

How does using the Semi-automated Ontology Builder for Aspect-based sentiment analysis improve the state-of-the-art Two-Stage Hybrid Model in terms of efficiency and effectiveness?

More accurate ABSA will be highly valuable in the field of market research. It can help firms improve their business, as ABSA extracts useful information from already easily accessible data, in the form of online reviews, and gives a clear overview of the company's strengths and weaknesses. Companies can act upon the feedback, and improve their customer experience. Furthermore, sentiment analysis can be of use for customers that use online reviews in their decision of purchasing a product.

The main contribution of this work is the proposal of SOBA, a semi-automatic domain ontology builder for aspect-based sentiment analysis. Previous approaches have mainly focused on building general sentiment lexicons (e.g., SenticNet [14], SentiWordNet [15], MPQA [16], EmoLex [17], etc.), often neglecting the ontological properties that exist between concepts. A notable exception is the common-sense sentiment ontology proposed in [18], but this is also a general representation, which does not consider the domain specificities in relation to sentiment. In contrast to approaches based on word embeddings and attentive Long Short-Term Memory/Convolutional Neural Networks [14, 19], in this work, we focus on word frequencies for ontology building due to the limited amount of available domain data when learning domain ontologies. To the best of our knowledge, SOBA is one of the first methods for developing in a semi-automatic manner domain sentiment ontologies for ABSA. A secondary contribution of this work is related to using word sense disambiguation both in the ontology construction phase and in the sentiment analysis phase, being thus able to deal with the ambiguous semantics of words. For this we have anchored the domain ontology concepts to their semantic counterparts (if these are present) from a semantic lexicon.

2. Related Work

Broadly speaking, sentiment analysis methods can be divided into knowledge-based and machine learning-based approaches [3]. This distinction is based on the fact that machine learning models require a certain amount of training data in order to perform well, while the power of knowledge-based models lies in the quality of the used knowledge repository.

Various machine learning algorithms have shown to be highly successful for sentiment analysis [4, 5, 6]. In recent work, researchers have focused on the application of neural attention models for aspect-based sentiment analysis, continuously improving performance. For example, [20] proposes the use of Long Short-Term Memory (LSTM) based on single-attention, [21] devises an attention based on distances between words in a sentence, [22] develops two types of aspect attention (with and without considering word order) at sentence level, and [23] gives an multi-hop attention model that iterates between aspect and context representations for aspect-based sentiment analysis. Hierarchical attention LSTM models for target and sentence representation have been proposed in [24]. However, these models require a substantial amount of training data in order to reach reasonable performance, which is not always available, for example for certain domains or languages. Therefore, some researchers focus on knowledge-based models for sentiment classification.

Knowledge-based models determine a text's sentiment by means of a knowledge repository. A widely used knowledge repository comes in the form of an ontology, which is defined as a formal, explicit specification of a shared conceptualization [25]. Models that make use of a domain ontology, classify the sentiment of a review as follows [7, 9, 26]: all words in a given text are analyzed by determining their meaning, grammatical function in the sentence, and relation to other words using an ontology. With this information, the model subsequently calculates the sentiment. An alternative to employing domain ontologies is the use of commonsense ontologies, providing cross-domain relations between concepts and sentiment values [18]. In addition, there are approaches that are based solely on the grammatical relations between words in a sentence [27, 28], as well as solutions that exploit the rhetorical structure of text [29, 30] for sentiment computations. In addition to the sentiment (associated to a certain aspect), using argumentation theory additional information can be provided by means of the number of supports or attacks given implicitly (e.g., by means of tags) by the users [31].

Apart from purely machine learning- or knowledge-based approaches, some models combine techniques from both fields. In [32] the author advocates that these hybrid models represent the path forward for improving performance. Inspired by the two-steps hybrid approach proposed in [33], in previous work we have proposed the Two-Stage Hybrid Model (TSHM) [9], a state-of-the-art knowledge-driven ABSA model.

2.1. Ontology learning

A major disadvantage of knowledge-driven models is that building an ontology is a difficult and time-consuming process, often done by manually analyzing text documents. [34] explains the different steps of ontology learning and the related complications, and visualizes these in the Ontology Learning Layer Cake (see Figure 1).

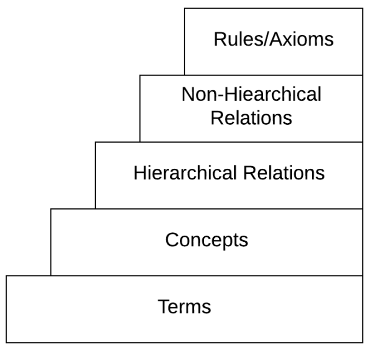


Figure 1: The Ontology Learning Layer Cake

The first step of designing an ontology is term selection, namely identifying domain-specific terms in text documents. At this stage, it is important to recognize words with the same meaning as synonyms. On the other hand, in the case of polysemous words, the different meanings of words should be distinguished, a process known as word sense disambiguation (WSD).

In the next step, concepts are formed by clustering all terms with the same meaning together, so that each concept has (i) a proper definition that is shared by its instances and extensions, and (ii) a set of linguistic realisations [34]. For example, a restaurant ontology could have the concept Drinks, which has lexicalizations ‘drinks’ and ‘beverages’. In addition to lexicalization properties, other properties can also be defined for a concept, depending on the purpose of the ontology. For example, in order to perform aspect-based sentiment analysis, [9] defines an aspect property in their ontologies, which links concepts to specific aspect annotations used in the labeled data set.

The third step of ontology building is to define type-of relations between concepts that establish the hierarchical ordering of concepts. For example, given the concepts Drinks and Soda, Soda should be recognized as a subclass of Drinks. The advantage of hierarchical relations is that properties of superclasses are automatically inherited by subclasses.

Next, non-hierarchical relations between concepts are added, including the domain and range of these relations. To illustrate, for the concepts Person and Drinks, a relation *consumes* can be defined, with Person as domain and Drinks as range. Also, a link between the Drinks concept and a “drinks” aspect in the labeled data set will automatically apply to all subclasses of Drinks like Soda.

Finally, the last step of ontology learning is to define rules. Specific rules can be used in ABSA in order to determine sentiment. For example, a restaurant ontology can store rules that claim that the concept Cold is positive when used to describe the concept Soda, but negative when used for Pizza.

2.2. Automatization of Ontology Learning

Since manually designing an ontology requires a lot of effort, earlier research has been done on (semi-)automated ontology or taxonomy building [10, 11, 12]. This significantly decreases the time and human effort needed to set up a knowledge repository, while increasing the overall size, so that more knowledge is included.

In the following sections, we discuss some approaches for several steps of ontology learning as found in existing literature. However, we do not discuss algorithms for adding non-hierarchical relations and rules, as these are to be added according to the ontology structure of [9], as shown in Section 4.5.

2.2.1. Term Selection

For term selection, [11] and [12] make use of a lexical approach proposed by [35]. This solution is based on calculating the relative importance of a word in domain texts compared to its importance in general English language. Furthermore, they also consider the consensus on terms across the used domain texts [36]. In other words, they check whether terms appear often in all domain-related texts and not just in a few of them. After calculating a score that takes into account both the relative importance and consensus across documents, terms are selected to be included in the ontology when the score is above a certain threshold.

2.2.2. Word Sense Disambiguation

[12] highlights the relevance of WSD of terms from the text corpora after term selection, which facilitates the subsequent concept definition step. WSD determines the proper sense of a word, so that also dictionary semantics are considered in the ontology.

A well-known WSD algorithm is the Simplified Lesk Algorithm (SLA) [37]. This approach compares the words that surround the considered term in the text, the context words, with the words of the example sentences that are provided in a dictionary per sense of this term. The sense whose words in the example sentences show the most overlap with the context words, is eventually picked to be the sense of the word. As a backup, when no overlap is found for any sense, the most frequently used sense in the English language for the considered term is set to be the term’s sense.

The performance of SLA is often used as baseline to compare other WSD algorithms with, and despite its simplicity, SLA is considered hard to beat [38].

2.2.3. Determining Hierarchical Relations

[11] evaluates two approaches to establish type-of relations between concepts, namely the subsumption method and hierarchical agglomerative clustering. When comparing the subsumption method and hierarchical clustering, the results suggest that the subsumption method is preferred when the quality of the taxonomy is of main importance. Additionally, the subsumption method performs better in case of shallow taxonomies.

2.3. Hypotheses

In this paper, we present a Semi-automated Ontology Builder for Aspect-based sentiment Analysis (SOBA), which is used to adjust TSHM [9], our base model. Incorporating a semi-automatically built ontology in TSHM could lead to higher efficiency, as the human effort needed to build an ontology is reduced. Moreover, it is hypothesized that a semi-automatically built ontology will include more knowledge, and therefore would show higher performance in ABSA. We have summarized this information in the following hypotheses:

Hypothesis I: Using a semi-automated ontology builder will decrease the human time needed to construct the ontology, therefore making the overall process of ABSA more efficient when using an ontology.

Hypothesis II: A semi-automatically built ontology will lead to higher accuracy of the Two-Stage Hybrid Model.

As the ontologies constructed using SOBA are specifically meant to be used in combination with TSHM, SOBA builds the ontologies following the ontology structure proposed in [9]. However, considering the importance of WSD [12], we add an extension to the original ontology structure. Namely, for each concept, we attach a sense property that defines the specific sense that the term represents (as given by a semantic lexicon). We also incorporate this adjustment in TSHM, so that WSD is used during the analysis of reviews. This way, we prevent that words are falsely linked to a concept according to the concept lexical representation, while the two words do not match semantically. This could then contribute to higher accuracy of TSHM, leading to our third hypothesis:

Hypothesis III: Taking into account dictionary semantics for concepts using word sense disambiguation will further increase the performance of the Two-Stage Hybrid Model.

To test these hypotheses, we create ontologies for ABSA using our semi-automated ontology builder. After this, we evaluate the performance of the resulting ontologies in TSHM against the performance of the manual ontologies of [9]. We apply our semi-automated ontology builder to two domains, namely the restaurant and laptop domain, to show that our ontology builder can be broadly used across domains.

3. Two-Stage Hybrid Model

As our base model, we use the Two-Stage Hybrid Model [9], a hybrid model with state-of-the-art results for ABSA. In the first stage, the model attempts to determine the sentiment per aspect for each sentence, by relating words in the evaluated text to the concepts and rules defined in the ontology. However, if the terms in the sentence are not found as concepts in the ontology, or if the ontology gives contradicting sentiment values, the second stage is used to determine the final sentiment.

The second stage is a bag-of-words (BoW) model that is used in combination with a Support Vector Machine (SVM), to classify the sentiment for the aspects. Alternatively, the second stage can be set to a different approach, majority labelling: when the first stage cannot decide on a sentiment, the second stage simply sets the sentiment to the most commonly expressed sentiment in the dataset. By comparing the performance for the two options for the second stage, the additional value of the BoW model can be measured, which indicates the synergy between knowledge-based and machine learning-based approaches.

Apart from using TSHM, [9] evaluates the BoW-model of the second stage by itself. Furthermore, it introduces an alternative BoW-model that includes ontology features. Therefore, [9] considers four models in total: (i) Ont+BoW, namely the TSHM using the BoW-model in the second stage; (ii) Ont, the TSHM using majority labelling; (iii) BoW, the single-stage model that consists of the BoW-model without ontology features; (iv) BoW+Ont, the single-stage BoW-model with ontology features.

In the next sections, we discuss the ontology structure and the various models of [9] in more detail.

3.1. The Ontology Structure

The structure of the ontology of [9] is illustrated in Figure 2. In this figure, n denotes the number of aspects and categories, which differs per domain. As shown, the ontology contains two main classes, namely *Mention* and *Sentiment*. These classes respectively represent concepts that indicate a reviewed aspect, and concepts that hold sentimental value (positive, negative, or neutral).

The *Mention* class contains three subclasses, *ActionMention*, *EntityMention*, and *PropertyMention*, which respectively represent the group of verbs, nouns, and adjectives. *Action-* and *PropertyMention* have at least the *GenericPositive<Group>* and *GenericNegative<Group>* classes, where <Group> is replaced by Action or Property. These two subclasses miss for *EntityMention*, as [9] assumes that nouns do not hold sentiment.

The *Action-*, *Entity-*, and *Property-Mention* classes all contain an *<Aspect/Category><Group>Mention* subclass for each aspect and category considered in the domain, where <Aspect/Category> is replaced by the relevant aspect or category, and <Group> is replaced by Action, Entity, or Property. All concepts that are related to the same aspect or category, will be grouped together in the corresponding *<Aspect/Category><Group>Mention* class. This way, the *<Aspect/Category><Group>Mention* classes are used to determine to which aspects and/or categories a concept refers to, by retrieving and checking the parent classes of the concept.

The aspects and categories used for the restaurant and laptop domains are listed in Section 5.2. For example, for the restaurant domain we have aspects such as Ambience, Food, and Location. Moreover, aspects can be further split into categories. To illustrate, the food aspect has the categories Prices, Quality, and Style Options. When aspects are not split into multiple categories, they will simply be labelled with the category General.

Furthermore, the classes *<Aspect/Category>ActionMention* and *<Aspect/Category>-PropertyMention* each contain two subclasses, namely *<Aspect/Category>Positive<Group>* and *<Aspect/Category>Negative<Group>*. These *<Aspect/Category>Positive<Group>* and *<Aspect/Category>Negative<Group>* classes are again not added for the classes of *EntityMention*.

However, for generality, Figure 2 implies that the *EntityMention* concepts can carry sentiment like *Action-* and *PropertyMention* concepts. Therefore, *EntityMention* contains a *GenericPositiveEntity* and *GenericNegativeEntity* class, and the *<Aspect/Category>EntityMention* classes contain the *<Aspect/Category>PositiveEntity* and *<Aspect/Category>NegativeEntity* subclasses.

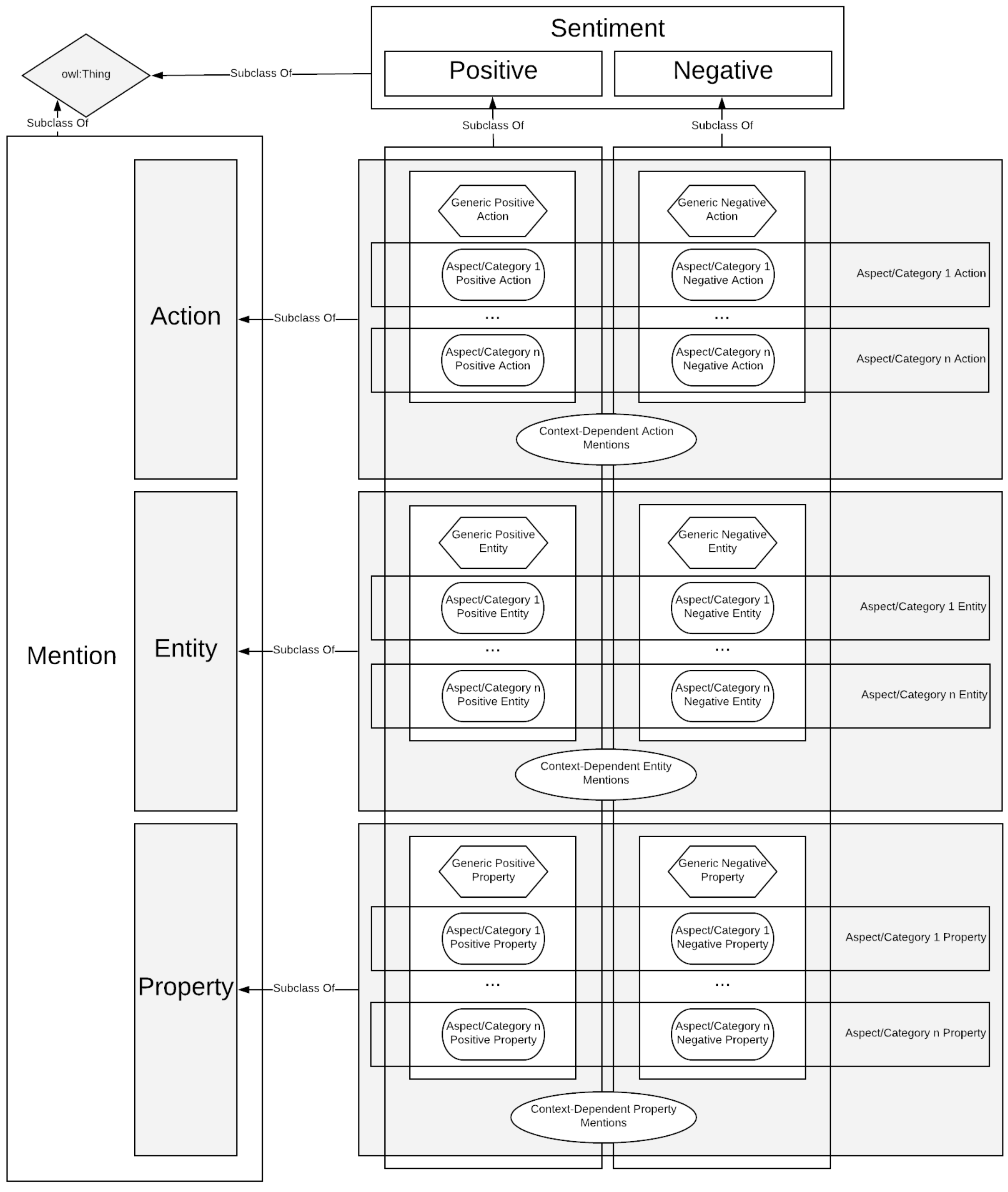
The other main class, *Sentiment*, has two subclasses called *Positive* and *Negative*, which respectively contain classes with positive and negative sentiment. The beforementioned *GenericPositive<Group>* and *<Aspect/Category>Positive<Group>* classes belong to *Positive*, and the *GenericNegative<Group>* and *<Aspect/Category>Negative<Group>* classes belong to *Negative*. Note that not all concepts are a subclass of *Sentiment*, as the ontology also includes concepts that do not express any sentimental value, but are used to determine the mentioned aspects in a sentence.

To determine to which of these two *Sentiment* subclasses a sentiment carrying concept, from now on referred to as a *Sentiment* concept, belongs, is done based on the word meaning and possibly the context, depending on the type of the concept, as explained below. In total, we distinguish three types of *Sentiment* concepts.

Type-1 concepts are the generic concepts that always express positive or negative sentiment regardless of the context, and that can be applied to any aspect. These concepts that express positive/negative sentiment belong to *GenericPositive<Group>*/*GenericNegative<Group>*, which are simultaneously a subclass of *Positive*/*Negative* and the corresponding *<Group>Mention* class. An example of a type-1 concept would be Good, which is a subclass of *GenericPositiveProperty*.

The type-2, aspect-specific, concepts are words that typically express only positive or negative sentiment towards one specific aspect or category. An example for the restaurant domain would be Rude, which only expresses negative sentiment towards the service aspect, but does not carry any sentiment for other aspects or categories. These type-2 concepts are organized in classes that carry a name of the form *<Aspect/Category>Positive<Group>* or *<Aspect/Category>Negative<Group>*, where <Aspect/Category> is replaced by the aspect or category that the word reviews.

Last, the type-3 concepts are context-dependent, namely words that could express sentiment for multiple aspects or categories, but not all aspects and categories like type-1 concepts. It is even possible that the word has ambiguous sentimental value, namely that it would be positive when reviewing one concept, but have negative meaning when it is used for another concept. An example would be the concept Cold, which is positive when used in a phrase as 'cold cola', but negative in 'cold pizza'. For type-3 concepts, the ontology adds a new class that is a subclass of the type-3 *Sentiment* concept and the target aspect concept, namely the intersection of the *Sentiment* concept and target concept. Furthermore, an axiom is added to the type-3 concept, that specifies whether this intersection subclass belongs to *Positive* or *Negative*.



Sentiment
Positive
Negative
owl:Thing
Subclass Of
Subclass Of
Subclass Of
Subclass Of
Mention
Action
Entity
Property
Subclass Of
Subclass Of
Subclass Of
Generic Positive Action
Generic Negative Action
Aspect/Category 1 Positive Action
Aspect/Category 1 Negative Action
Aspect/Category 1 Action
...
Aspect/Category n Positive Action
Aspect/Category n Negative Action
Aspect/Category n Action
Context-Dependent Action Mentions
Generic Positive Entity
Generic Negative Entity
Aspect/Category 1 Positive Entity
Aspect/Category 1 Negative Entity
Aspect/Category 1 Entity
Aspect/Category n Positive Entity
Aspect/Category n Negative Entity
Aspect/Category n Entity
Context-Dependent Entity Mentions
Generic Positive Property
Generic Negative Property
Aspect/Category 1 Positive Property
Aspect/Category 1 Negative Property
Aspect/Category 1 Property
Aspect/Category n Positive Property
Aspect/Category n Negative Property
Aspect/Category n Property
Context-Dependent Property Mentions

Figure 2: Ontology Structure

Apart from the hierarchical relations between the classes that are shown in Figure 2, the ontology also stores properties of the concepts. All concepts in the ontology can have two properties.

First, the *lex* property links the concept to its various lexicalizations (one property per lexicalization). For example, words such as 'bad' and 'badly' are all linked to the Bad concept through this lexicalization property.

Second, the *<Aspect/Category><Group>-Mention* classes have one or multiple *aspect* properties, which link the concepts to all the aspect categories for which they are applicable. These *aspect* properties are given in the form that matches how the aspect annotations are provided in the employed dataset, namely <ASPECT>#<CATEGORY>, where <ASPECT> is one of the considered aspects of the domain, and <CATEGORY> denotes which category of the aspect is reviewed. For example, the *FoodActionMention* class would be linked to the FOOD#PRICES, FOOD#QUALITY, and FOOD#STYLE_OPTIONS aspects.

Last, the *disjointWith* relation is added for certain classes of the *Sentiment* class. Namely, the *Positive* class has a *disjointWith* relation with *Negative*. Likewise, each *GenericPositive<Group>* has a *disjointWith* relation with its corresponding *GenericNegative<Group>* class, and each *<Aspect>Positive<Group>* with its corresponding *<Aspect>Negative<Group>*.

3.2. Ontology Sentiment Algorithm

The first stage of TSHM uses some basic tools from the Stanford CoreNLP package to pre-process the reviews, such as dependency parsing, tokenization, lemmatization, and part-of-speech tagging. After this, Algorithm 1 is used to compute the sentiment of the reviewed aspects per sentence. It has as input the used ontology *o*, a currently reviewed aspect *a*, and a Boolean *useBOW*, which indicates whether the bag-of-words model is used as backup.

The algorithm first finds, per sentence that reviews the relevant aspect, all words that are related to a concept in the ontology, and evaluates each of these words. The algorithm starts with checking whether the word is negated. Negation is detected by checking for *neg* relations in the dependency graph, which indicate negations, or by looking for a negation word ('not' or 'never') within a window of three words preceding the *Sentiment* term [39].

Next, the sentimental value of the word is determined according to the type of the word. In case the word corresponds to a type-1 *Sentiment* concept, the superclasses of the considered concept are added to a group of superclasses. This group of superclasses is used to determine the sentiment as follows. If only the *Positive* or *Negative* class is found in this group, the first stage predicts a positive or negative sentiment for the considered aspect, respectively. However, if none or both are found, the first stage remains indecisive, and the second stage is used to give a final classification.

Algorithm 1 The Two-Stage Hybrid Model: Ontology Sentiment Algorithm

```
1: function PredictSentiment(Ontology $o$, Aspect $a$, Boolean $uesBOW$)
2:     Set<String> $foundURIs = \perp$
3:     Set<Word> $words$ = GetWordsWithURI(GetSentence(a))
4:     for all Word $w \in words$ do
5:         Boolean $negated$ = isNegated($w$)
6:         String URI = getURI($o, w$)
7:         if isType1($o, URI$) then
8:             $foundURIs = foundURIs \cup$ getSuperclasses($o, URI, negated$)
9:         end if
10:        if isType2($o, URI$) $\wedge$ CategoryMatches($a, URI$) then
11:            $foundURIs = foundURIs \cup$ getSuperclasses($o, URI, negated$)
12:        end if
13:        if isType3($o, URI$) then
14:            for all String $relURI \in$ getRelatedAspectMentions($w$) do
15:                if CategoryMatches($a, URI$) then
16:                    String newURI = addSubclass($o, URI, relURI$)
17:                    $foundURIs = foundURIs \cup$ getSuperclasses($o, newURI, negated$)
18:                end if
19:            end for
20:        end if
21:    end for
22:    Boolean $foundPositive$ = (PositiveURI $\in foundURIs$)
23:    Boolean $foundNegative$ = (NegativeURI $\in foundURIs$)
24:    if $foundPositive \wedge \neg foundNegative$ then
25:        return Positive
26:    else if $\neg foundPositive \wedge foundNegative$ then
27:        return Negative
28:    else if $useBOW$ then
29:        return getBOWPrediction($a$)
30:    else
31:        return getMajorityClass
32:    end if
33: end function
```

For type-2 concepts, the algorithm works similarly as for type-1 concepts, but it performs an extra check for whether the concept is of the same aspect category as the currently considered aspect. This is done by retrieving the superclasses of the type-2 concept and checking the *aspect* property of the *<Aspect/Category><Group>Mention* superclass. If the aspect category does not match, the type-2 *Sentiment* is considered neutral. For example, the sentimental value of 'rude' is only recognized when currently the Service aspect is analyzed, and subsequently all superclasses will be stored. For other aspect categories, 'rude' will be ignored.

Last, when a type-3 *Sentiment* is detected, the algorithm first finds the related aspect concepts, namely the target concepts that are reviewed by the type-3 word in the sentence, using the dependency graph. After this, the algorithm checks for matching aspect categories of the *Sentiment* and target concept, and whether these aspect categories match the currently considered aspect. For each found match, the algorithm creates a new class that is a direct subclass of the intersection of the type-3 *Sentiment* concept and target concept class. Next, all superclasses of this newly created subclass are added to a group of superclasses, together with the superclasses of the *Sentiment* concept and the target concept. If any axiom is available for this intersection subclass, the superclasses will contain either *Positive* or *Negative*. If the combination of *Sentiment* concept and target concept is unknown, the ontology cannot conclude on the sentiment.

For example, suppose that in the ontology design, the following intersection classes and axioms are present:

1. Cold $\sqcap$ Soda $\sqsubseteq$ Positive
2. Cold $\sqcap$ Service $\sqsubseteq$ Negative

If then the currently considered aspect category is DRINKS#GENERAL, and a review contains 'cold cola', the algorithm first creates a new subclass of both the Cold and Cola class:

3. ColdCola $\sqsubseteq$ Cold $\sqcap$ Cola

Since Cold $\sqcap$ Cola is a subclass of Cold $\sqcap$ Soda, the algorithm can then conclude, using (1) from the ontology, that:

4. ColdCola $\sqsubseteq$ Positive

The algorithm uses the antonym superclasses of the *Sentiment* concepts to determine the sentiment in case of negation. The antonym classes are found by retrieving classes with a *disjointWith* relation to the superclasses of the considered *Sentiment* concept.

The next section explains the bag-of-words model that is used as backup to the first stage, when the Boolean *useBOW* is set to 'true'. If *useBOW* is 'false', majority labelling is used instead.

3.3. Bag-Of-Words Sentiment Algorithm

The second stage of TSHM uses the Bag-of-Words (BoW) model, so that each sentence is represented as a vector. This vector contains an entry for each known word in the vocabulary. The entry is 0 if this word is found in the sentence, and 1 if the word is not present. Furthermore, the vector has an entry for all aspect categories, of which the currently reviewed aspect category has a value 1, and all other have value 0. Additionally, the vector has a sentiment value feature, which is a sentiment value calculated based on a score that is determined using the Stanford Sentiment Annotator tool [40], a state-of-the-art sentiment analysis tool for sentences.

This BoW-model is used together with a Support Vector Machine (SVM), which classifies the reviewed aspect as positive, negative or neutral, based on all the features included in the BoW.

3.4. Alternative: Bag-of-Words Model with Ontology Features

[9] also considers an alternative model, namely the hybrid model of [8] called BoW+Ont. This model, a BoW-model with ontology features, simply consists of one stage, but again makes use of both an ontology and a machine learning algorithm. Namely, this BoW includes two additional features, shown as Positive and Negative in Figure 3.

The Positive entry is 1 if positive sentiment is determined using the ontology for the relevant aspect, and 0 else. Similarly, the Negative entry is 1 when the ontology detected negative sentiment, and 0 otherwise. However, when both positive and negative sentiment are found, both entries are 0, as the ontology is considered not to provide any useful information. Again, an SVM is used to classify the sentiment based on features in the BoW-model.

4. SOBA Ontology Builder

This section describes how our Semi-automated Ontology Builder for Aspect-based sentiment analysis (SOBA) is defined. Since our ontology builder is only semi-automated, manual input from the user is required before the concepts are added. This way, the automatization will not be at the cost of the quality of the ontology [10].

The following sections describe the five main tasks of SOBA. First, the ontology builder creates a skeletal ontology based on a given base ontology,

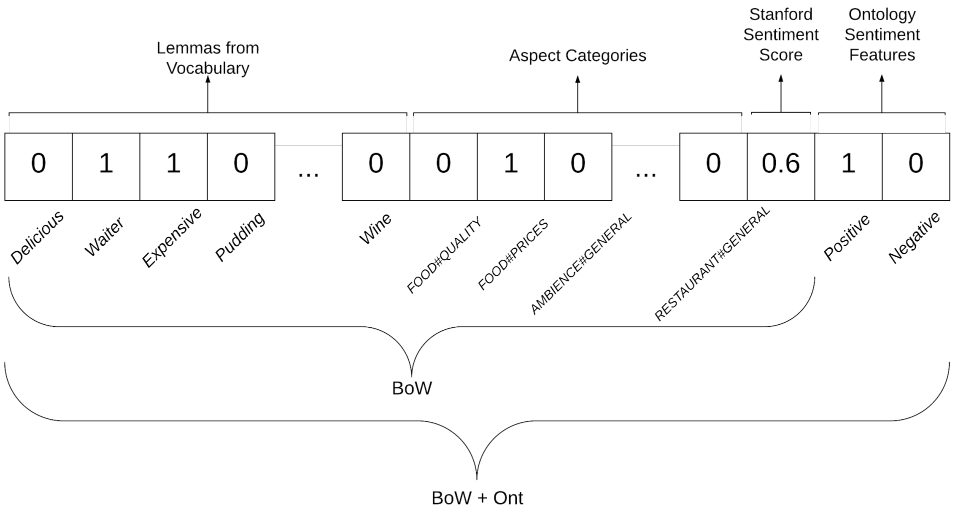


Figure 3: The Bag-of-Words Vector Entries

as explained in Section 4.1. Second, the ontology builder suggests domain-specific terms to the user to add to the skeletal ontology, as described in Section 4.2. Third, after the terms are accepted by the user, SOBA retrieves the specific sense of the words using word sense disambiguation (WSD), in order to form concepts, as shown in Section 4.3. Fourth, Section 4.4 describes how the ontology builder suggests possible parent classes for these accepted concepts, so that hierarchical relations with respect to the existing classes are defined. Last, Section 4.5 describes how the non-hierarchical relations, rules, and properties of a concept are defined.

4.1. Skeletal Ontology

Our base ontology follows the basic structure of the ontology of [9], as shown in Figure 4. This base ontology contains the two main classes *Mention* and *Sentiment*. The *Mention* class contains the *AspectMention*, *EntityMention*, and *PropertMention* classes, and the *Sentiment* class has the subclasses *Positive* and *Negative*.

Moreover, for the three subclasses of Mention, we add the *GenericPositive<Group>* and *GenericNegative<Group>* subclasses, which are subclasses of the corresponding *Positive* or *Negative* class and *<Group>Mention* class. Here, <Group> is replaced by Action, Entity or Property, according to its *<Group>Mention* superclass. A *disjointWith* relation is present between the *Positive* and *Negative* classes, as well as for each of the *GenericPositive<Group>* and *GenericNegative<Group>* classes. We here deviate from the original ontology of [9], as we assume that nouns can also hold sentiment. To illustrate, Disappointment would be a subclass of the *GenericNegativeEntity*.

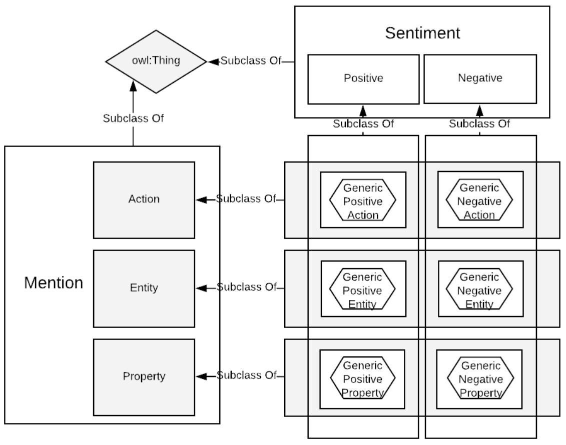


Figure 4: Base Ontology

Our ontology builder uses this base ontology as starting point, and expands it to a domain-specific skeletal ontology by adding some general classes for each considered aspect and category in the domain. For *ActionMention*, *EntityMention*, and *PropertyMention*, we add a subclass for each aspect. Furthermore, following the structure of the ontology of [9], a subclass for each category is added. These subclasses are named *<Aspect/Category><Group>Mention*, where <Aspect/Category> denotes the relevant aspect or category, and <Group> is as defined before. To these subclasses, we also add the relevant annotation properties. The lexical forms of the aspect or category are added to the subclass as a *lex* property, along with the lexicalizations of the synonyms found via WordNet [13]. Besides this, we also attach the corresponding *aspect* properties, in the form of <ASPECT>#<CATEGORY>, following the annotations from our dataset.

After this, we add two subclasses to each *<Aspect/Category><Group>Mention*, and name these classes *<Aspect/Category>Positive<Group>* and *<Aspect/Category>Negative<Group>*. These new subclasses are also a subclass of *Positive* and *Negative*, respectively.

The above specifications are illustrated in the ontology structure of Figure 2. The aspects and categories per domain are to be defined by the user. For example, for the restaurant domain, we use the aspects Service, Restaurant, Location, Food, and Sustenance, and the categories Prices, Style&Options, and Quality, so that n is 8. The laptop domain, which is more complex than the restaurant domain, contains a lot more aspects and categories, so that n is 88.

For categories that consist of multiple words, such as Style&Options, the category is added such that the words are split by the character '&' or '_'. For these multi-word categories, each separate word and its synonyms are also added as a *lex* property to the corresponding *<Aspect/Category><Group>Mention* class in the ontology.

We already explained above how *aspect* and *lex* properties are added in the skeletal ontology. In addition to this, we introduce a new property to our ontology structure, that contains information on the exact semantic meaning of concepts, called *sense*. This *sense* property is a code that is unique for each sense of a term, and is of the form <word>#<pos>#<number>, where <word> is the lemma that the concept represents, <pos> is the part-of-speech of the word, and <number> is the sense number as defined by the semantic lexicon (dictionary) WordNet [13]. We add this *sense* property to all concepts that have lexicalizations. This means for the skeletal ontology that this property is added for all *<Aspect/Category><Group>Mention* classes.

Last, we define the non-hierarchical *disjointWith* relations as in the ontology of [9], between each of the *<Aspect/Category>Positive<Group>* and *<Apect>Negative<Group>* classes.

After the skeletal ontology is created, the ontology builder continues with the next step, term extraction and suggestion.

4.2. Term Suggestion

This section describes how the ontology builder selects domain-specific terms from text documents to suggest to the user to add to the ontology, as discussed in Section 4.2.1. Section 4.2.2 explains how multi-word phrases, also known as quality phrases, are detected.

4.2.1. Term selection

The ontology builder extracts terms from given text documents, based on their relevance for the domain. This is determined using frequency values, namely by determining how often a word appears in a given domain-related document (the domain corpus), relative to how often the words appear broadly in any text, that are not domain-specific (the contrastive domain corpus) [11].

First, we calculate the domain pertinence (DP) for each term t, which measures the relevance of the term for the considered domain [35]:

$$DP_D(t) = \frac{freq(t,D)}{max_j(freq(t,C_j))}, \tag{1}$$

where D refers to the domain corpus and C denotes the contrastive corpus. Index j refers to a specific document within the contrastive corpus.

We then calculate the domain consensus (DC), which measures the consensus on a term across the documents from the domain corpus [36], and is calculated as:

$$\begin{aligned} DC_D(t) = -\textstyle\sum_{d\in D} norm\_freq(t,d) \\ *log(norm\_freq(t,d)), \end{aligned} \tag{2}$$

where $norm\_freq(t,d)$ is defined as:

$$norm\_freq(t,d) = \frac{freq(t,d)}{max_{d\in D}(freq(t,d))}. \tag{3}$$

Last, we create a final criterium that takes into account both the DP and the DC, namely the term score:

$$\begin{aligned} term\_score(t,D) = \alpha\frac{DP_D(t)}{max_{t\in D}(DP_D(t))} \\ +\beta\frac{DC_D(t)}{max_{t\in D}(DC_D(t))}, \end{aligned} \tag{4}$$

where α and β are parameters on the range (0.0, 1.0], and for which we determine the optimal values using a grid search with step size 0.1 for α and β. In this grid search, we aim to maximise the probability that the ontology suggests terms that would be accepted by the user, so we find a combination of α and β that results in the highest fraction of terms accepted by the user. Note that for Equation (4), only the ratio of α/β is important, as multiplying α and β by the same number would not lead to any difference in the ranking of the terms according to this score. Therefore, we add the restriction $\alpha + \beta = 1$ when optimizing these parameters [12].

After the term scores are calculated for each term, a fraction of μ_n, μ_v, and μ_a of the terms with highest term scores are suggested to the user to add to the ontology. Here, the values of μ_n, μ_v, and μ_a are respectively the fraction of suggested nouns, verbs, and adjectives, and the optimal values again need to be determined based on some criterium.

Since the values of the fractions need to ensure quality of the suggested terms, while also keeping the number of suggested terms at a reasonable level, we use the harmonic mean of the number of accepted terms and the ratio of accepted terms as criterium to optimize the above mentioned fraction values [12]:

$$fraction\_score(\mu_{pos}) = \frac{2}{(AC_{pos}(\mu_{pos}))^{-1} + (\frac{AC_{pos}(\mu_{pos})}{SUG_{pos}(\mu_{pos})})^{-1}}. \tag{5}$$

In this equation, AC_{pos} is the number of accepted terms by the user, and *pos* indicates the considered type of part-of-speech of the terms (noun, verb, or adjective). Furthermore, SUG_{pos} is the total number of suggested terms of the type *pos*. We find the optimal values for each μ_{pos} in a grid search with step size 0.01, on a range of [0.10, 0.20], corresponding to the top 10% to 20% of terms with highest fraction score. This range is chosen as such to ensure that the number of suggested terms will not be too low or too high, so that quality is maintained.

4.2.2. Finding Quality Phrases

For the term extraction step, it is important that the ontology builder can extract relevant phrases that consist of multiple words, also called quality phrases [41]. We detect these quality phrases using an approach that is similar to how [9] detects quality phrases in texts. Namely, we go over the text of the domain corpus, and we check for each sequence of words whether they are defined in WordNet [13]. This ensures that only phrases that have a sensible meaning are suggested to the user, and not also sequences of words that simply often appear together but do not form an expression with a specific meaning.

When looking for quality phrases, we only consider sequences of up to four words, as a window of four words more or less covers all multi-word phrases of the English language [10]. Furthermore, longer matches superseed shorter matches, meaning that if a multi-word phrase is found, the shorter phrases of which the multi-word phrase consists are discarded.

When a quality phrase is detected, we calculate its term score as described in Section 4.2.1, and suggest it to the user when it belongs to the top fraction of the corresponding part-of-speech terms. The part-of-speech of the quality phrase is as defined by WordNet.

4.3. Concepts: Senses and Synonyms

When creating concepts from the accepted terms, it is important to define the exact sense of the term [12]. This is especially true for polysemous words. Therefore, we apply word sense disambiguation (WSD) to the selected terms from the domain corpus, using the Simplified Lesk Algorithm (SLA). We use SLA because of its efficiency and relatively high performance [38].

Algorithm 2 The Simplified Lesk Algorithm

```
1: function SimplifiedLesk(Word $w$, Set<Word> $context$)
2:     Synset $best\_synset$ = most frequently used synset of $w$
3:     int $max\_overlap = 0$
4:     for all Synset $s \in$ getSynsets($w$) do
5:         Set $gloss$ = set of words in example sentences of $s$
6:         int $overlap$ = computeOverlap($gloss$, $context$)
7:         if $overlap > max\_overlap$ then
8:             $max\_overlap = overlap$
9:             $best\_synset$ = s
10:        end if
11:    end for
12:    return $best\_synset$
13: end function
```

Algorithm 2 shows the pseudocode of SLA. For each word found in the domain corpus, we store all the context words of this term, namely we store all nouns, verbs, and adjectives that appear in a window that spans the whole review, as we assume that words have only one meaning per review [12].

Next, we retrieve all synsets that the word is part of, as defined by WordNet [13], along with the example sentences that are provided by WordNet for each synset. For each of the found synsets, the algorithm counts the number of words that appear in the example sentences as well as in the context words. In the end, the synset that has the most overlapping words between the example sentences and the context, is set as the sense of the word as encountered in the domain corpus.

When the exact sense of a concept is found, we add the sense code as a *sense* property, again represented in the form of <word>#<pos>#<sense>, as described before. Furthermore, we also retrieve all synonyms of the synset to which the sense belongs, and add them to the concept as a *lex* property. As a result, each concept represents one synset, and has a set of lexicalizations that consists of all terms that share the semantic meaning of that synset.

For this *sense* property to be useful in detecting sentiment, we also incorporate WSD in TSHM. Whenever a word of a review is analyzed, we first apply SLA to find the synset of the word. After this, the algorithm recalls all concepts from the ontology with a *sense* property. For each concept, it translates the *sense* code to a synset, and checks if it corresponds with the considered word. If this is the case, the URI of that concept is retrieved and used in the algorithm. However, if no concept is found that shares the same meaning with the currently analyzed word, the algorithm still checks whether the word appears as lexicalization of a concept in the ontology, as done in the original TSHM.

4.4. Hierarchical Relations

After a concept is formed, the ontology suggests parent-child relations to determine where the concept should be added in the ontology. This process of suggesting parent classes is different depending on whether the child concept carries sentiment or not. Therefore, we distinguish between sentiment concepts, concepts that have sentimental meaning, while possibly referring to a specific aspect, and aspect concepts, which are concepts that refer to a specific aspect or category, but do not express any sentiment.

This section describes the several steps of finding the parent-classes for concepts. First, we discuss how we make a selection of potential parent-classes, as explained in Section 4.4.1. Next, we determine which of these potential parent classes are suggested to the user, and in what order. This process is different for the aspect concepts and the sentiment concepts. For both kinds of concepts, we use the subsumption method [42], in order to determine to which aspects or categories the concept is related to. This is described in Section 4.4.2. For the sentiment concepts, an additional step is required, that determines the sentimental value of the concepts, as shown in Section 4.4.3.

4.4.1. Potential Parent Classes

To make the parent selection more efficient, we make a pre-selection of potential parent classes that are possibly suggested to the user. In doing this, we distinguish between nouns, verbs, and adjectives. This means that we only consider subclasses of *ActionMention* as parents for verb concepts, *EntityMention* subclasses for noun concepts, and only *PropertyMention* subclasses for adjective concepts.

Furthermore, we make use of the division between sentiment and aspect concepts. Namely, we only consider classes that are a subclass of *Sentiment* as possible parent classes for sentiment concepts, and *Mention* classes that are not a subclass of *Sentiment* for aspect classes. This means that for the aspect concepts, the potential parent classes are the *<Aspect/Category><Group>Mention* classes, and the potential parents of sentiment concepts are the *<Aspect/Category>Positive<Group>* and *<Aspect/Category>Negative<Group>* classes.

4.4.2. Aspect or Category: Subsumption Method

To determine to which aspect- or category-related classes a concept is to be added, we use the subsumption method [42], which has shown to perform well for ontologies that do not have a lot of layers [11], as is the case with the structure of our ontology (see Section 4.1).

The subsumption method is based on the degree of co-occurrences between the lexicalizations of the potential parent and child classes found in the text documents of the domain corpus. Let x denote the potential parent concept and y the considered child concept, then a parent-child relation is suggested by the ontology-builder whenever the following inequalities hold:

$$P(x|y) \geq t, P(y|x) < t, \tag{6}$$

where t denotes a certain threshold value.

For our research, we replace the t of the second equation by 1, as suggested by [11], who found that replacing the threshold value by 1 for the second inequality leads to more accurate parent-child suggestions. Equation (6) can then be interpreted as follows: given the child y, the parent x needs to appear at least t percentage of the time. However, given parent x, child y does not always appear with the parent.

The optimal value of t is still left to be determined, and we do this again using a grid search with step size 0.1, on a range of [0.0, 1.0], according to a criterium that takes into account the acceptance ratio of all terms and parent relations, and the number of all accepted terms and parent relations [12]:

$$threshold\_score(t) = \frac{2}{(AC_{all}(t))^{-1} + (\frac{AC_{all}(t)}{SUG_{all}(t)})^{-1}}. \tag{7}$$

This equation is the harmonic mean of $AC_{all}(t)$, the overall number of accepted terms and parent-child relations, and $SUG_{all}(t)$, the overall number of suggested terms and parent-child relations for a given t. We include the number of accepted terms, and not just the number of accepted parent-child relations, since a concept is rejected entirely if no correct parent-child relation is suggested to the user.

Finally, when the optimal value of t is found, it is still possible that multiple parent classes satisfy the inequalities of Equation (6). Therefore, we define a parent score for each potential parent class, calculated as follows:

$$parent\_score(x, y) = P(x|y), \tag{8}$$

where again x is the set of lexicalizations for a potential parent class, and y is the concept that needs to be added. We then rank the potential parents according to their parent scores, therefore suggesting them to the user in the order where parents with higher scores are suggested first.

As explained in the previous section, the potential parents x are the *<Aspect/Category><Group>Mention* classes for the aspect concepts, where <Group> is as corresponding to the type (Action, Entity, or Property) of concept y. For the sentiment concepts, the potential parents are the *<Aspect/Category>Positive<Group>* and *<Aspect/Category>Negative<Group>* classes. Note that for *<Aspect/Category>Positive<Group>* and *<Aspect/Category>Negative<Group>* classes referring to the same <Aspect/Category>, the parent scores will be equal, as the parent score does not take into account sentiment. Therefore, we use an alternative way to determine the sentiment of the concept, as explained in the next section.

4.4.3. Sentimental Value: Sentiment Score

As done for the aspect concepts, we determine the parent scores for the potential parents of sentiment concepts as described in the previous section. However, we also determine the sentiment (positive or negative) that the sentiment concept expresses towards aspects or categories, so that the corresponding *<Aspect/Category>Positive<Group>* or *<Aspect/Category>Negative<Group>* class will be suggested to the user first.

Therefore, we calculate the sentiment score for sentiment concepts as defined by [43]:

$$sentiment\_score(t) = \frac{\sum_{d \in D}(rating(d) * \frac{n(t,d)}{\sum_{w \in sentiments(D)} n(w,d)})}{\sum_{d \in D} rating(d)}, \tag{9}$$

where t is the lexical representation of the sentiment concept, D is the set of reviews d of the domain corpus, and $rating(d)$ is the star rating for review d, rescaled to a score between 0 and 1 using Min-Max normalisation. Moreover, $n(t, d)$ denotes the number of times that t is used in review d, and $sentiments(D)$ is the set of sentiment concepts found in the domain corpus.

Using this sentiment score, the order in which the potential parent classes for sentiment-concepts are suggested to the user is determined as follows: the parent-classes are ranked according to their parent score of Equation (8), which depends only on the aspect or category they represent, and not whether the parent class represents positive or negative sentiment. Of the potential parent classes that are related to the same aspect or category, the positive variant is suggested first if the sentiment score indicates positive sentiment. If negative sentiment is determined by the sentiment score, the negative variant is suggested first.

Last, it is important to note that some *Sentiment* concepts are not necessarily related to an aspect or category, and that there are two potential parent classes for *Sentiment* concepts for which we cannot calculate the parent scores, namely the *GenericPositive<Group>* and *GenericNegative<Group>* classes, which do not have any lexicalizations. Therefore, when suggesting parent classes for *Sentiment* concepts, we first prompt the *GenericPositive<Group>* and *GenericNegative<Group>* classes. Again, which of these two generic *Sentiment* classes is suggested first, is determined based on the sentiment score. When both generic *Sentiment* classes are rejected, the ontology builder then suggest the other potential parent classes, which are related to a specific aspect or category, in the order given by the parent and sentiment scores.

The following example serves as an illustration of how the parent classes are suggested to a user in the case of a sentiment concept. Suppose the user accepts the term 'delicious' as a sentiment carrying adjective for the restaurant domain. The algorithm then computes the parent score for all *<Aspect/Category>PositiveProperty* and *<Aspect/Category>NegativeProperty* classes. Next to the parent score, also the sentiment score for 'delicious' will be computed. Suppose that this indicates positive sentiment, then the order in which the potential parent classes are suggested will be as follows: the *GenericPositiveProperty* class, the *GenericNegativeProperty* class, the *<Aspect/Category>PositiveProperty* class with highest parent score, the *<Aspect/Category>NegativeProperty* class with highest parent score, the *<Aspect/Category>PositiveProperty* class with second highest parent score, the *<Aspect/Category>NegativeProperty* class with second highest parent score, et cetera.

Using the above procedure, all potential parent classes of which the parent score exceeds the threshold t will be suggested, so that the user can accept multiple parents, as done in the case of type-3 concepts. However, when the positive variant of an aspect- or category-related class is already accepted, the negative variant will not be suggested anymore. Furthermore, if the user does not accept any parent class for a concept, the concept will not be added to the ontology.

4.5. Non-Hierarchical Relations, Rules and Properties

In this section we discuss the non-hierarchical relations, rules, and properties that are used in the ontologies for TSHM. First, the ontology structure makes use of the non-hierarchical *disjointWith* relation to find antonym classes in case of negation. As these have already been defined for the classes in the skeletal ontology, these *disjointWith* relations are not considered anymore when adding concepts.

Second, we use a set of rules for type-3 concepts in order to determine their sentimental value per concept that they review. Namely, whenever multiple parent classes are accepted for a *Sentiment* concept, a rule will be defined that determines the sentimental value of this *Sentiment* concept in each combination with a reviewed target concept. These rules are added together with the *Sentiment* concept to the ontology.

Last, we add two properties to the concepts, as already described in Section 4.3. First, we use word sense disambiguation to obtain and add the sense of the concept as a *sense* property, which specifies the meaning of the concept as defined by WordNet [13]. As described before, this *sense* property is a code that is unique for each possible sense of a concept, of the form <word>#<pos>#<number>. Second, we add all lexicalizations of a concept and all lexicalizations of the synonyms that are defined for the concept sense by WordNet as a *lex* property.

5. Data

This section describes the data that is used for the ontology learning by the Semi-automated Ontology Builder for Aspect-based sentiment analysis (SOBA) (Section 5.1), and for evaluating our SOBA ontologies in the Two-Stage Hybrid Model (TSHM) (Section 5.2). To test whether our ontology builder is fit for use across multiple domains, we test its performance on the restaurant and laptop domain.

5.1. Dataset for Ontology Learning

In order to determine the relevance of words for a specific domain for term selection, we evaluate the frequency with which words appear in domain-related texts, the Domain Corpus, and compare this to the frequency of the words with which they appear in the Contrastive Corpus, which represents the standard English language (see Section 4.2).

This section describes the datasets that we use as Domain and Contrastive Corpora.

As Domain Corpora, we use external restaurant reviews from the Yelp Dataset Challenge [44] for the restaurant ontology. For the laptop ontology, we use data from Amazon [45]. Both datasets were pre-processed, so that they only contain restaurant and laptop reviews, respectively, and so that they both contain 5001 reviews each. Equal sizes of the datasets with external reviews for the two domains allows for fair comparison of the performance of our ontology builder on the different domains.

Furthermore, we also use the SemEval 2016 [46] reviews for restaurants and laptops from the training sets as domain corpora for the ontology learning of the semi-automatically built ontologies. This is so that we can fairly compare the performance of the semi-automatically built ontologies and the manually built ontologies of [9], since the manual ontologies are also built based on the concepts found in the training data of the SemEval datasets.

As Contrastive Corpus, we use several non-copyrighted novels from the English literature that are provided as text files by Project Gutenberg[1]. The used texts are Alice's Adventures in Wonderland by Lewis Carrol, Pride and Prejudice by Jane Austen, The Adventures of Sherlock Holmes by Arthur Conan Doyle, The Adventures of Tom Sawyer by Mark Twain and Great Expectations by Charles Dickens.

5.2. *Dataset for Two-Stage Hybrid Model*

The SemEval datasets are made available to enable researchers of aspect-based sentiment analysis (ABSA) to easily compare the performance of their model to that of others, and are therefore widely used in earlier research on ABSA. As [9] also use these datasets for TSHM, this enables us to evaluate the effect of our extensions relative to our base model.

For each domain, the datasets contain a training set and a test set. Each review is provided with annotations so that the text is readily split into sentences and the mentioned aspect categories per sentence are given. When aspects are explicitly mentioned, the target expression is also provided, together with the indices of the target expression. When aspects are implicitly evaluated, the target is given as $NULL$ in the annotations. Furthermore, the true polarity (positive, negative, or neutral) for each mentioned aspect per sentence is reported.

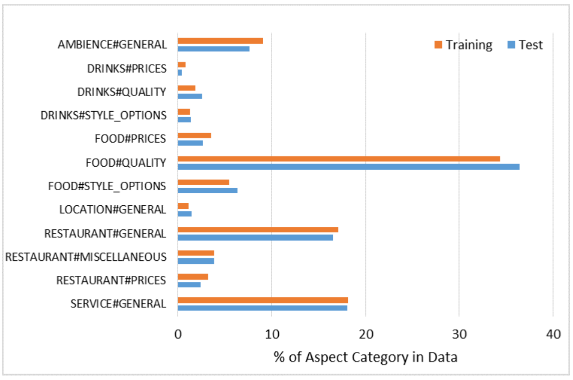


Figure 5: Relative frequencies of aspect categories in SemEval 2016 Restaurant dataset

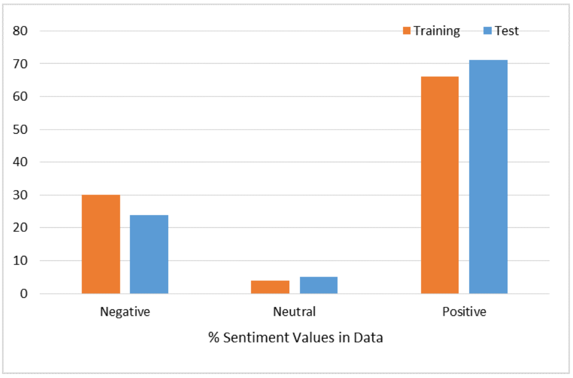


Figure 6: Relative frequencies of sentiment values in SemEval 2016 Restaurant dataset

For the restaurant domain, the SemEval 2016 training dataset has 350 reviews whereas the test dataset has 90 reviews. Figure 5 shows how often the different aspect categories of the restaurant domain are reviewed in the training and test datasets. The relative frequencies show that the aspect categories are more or less mentioned equally often in the training and test datasets. The most frequent aspect category is RESTAURANT#QUALITY.

Furthermore, Figure 6 shows that generally, people are more likely to express positive sentiment in the SemEval 2016 Restaurant dataset. This also implicates that the majority labelling approach in TSHM will predict positive for all reviews for which the ontology could not determine the sentiment.

[1]`http://www.gutenberg.org/wiki/MainPage`

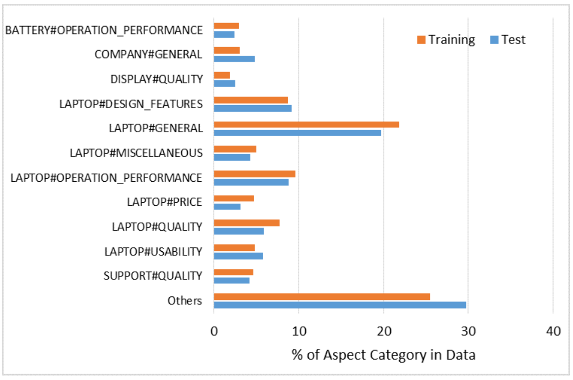


Figure 7: Relative frequencies of eleven most used aspect categories in SemEval 2016 Laptop dataset (all other categories grouped together under 'Others')

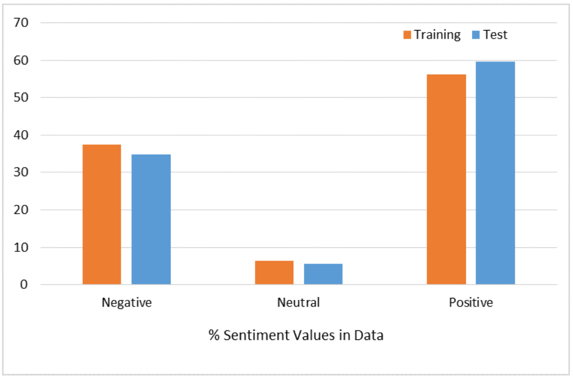


Figure 8: Relative frequencies of sentiment values in SemEval 2016 Laptop dataset

For the laptop domain, again a training dataset is provided, which contains 395 reviews, and a test dataset, consisting of 80 reviews. Figure 7 shows the relative frequencies for how often each aspect category of the laptop domain appears in the training and test datasets. Only the eleven most often reviewed categories are shown, and all other categories are grouped together under 'Others'. In total, the laptop domain considers 88 categories, of which many are only mentioned a few times throughout the datasets. Noteworthy is that some categories are not present in the training dataset, but only appear in the test data. The most frequent aspect category is LAPTOP#GENERAL.

Figure 8 shows how often the different sentiment values are used in the reviews of the laptop training and test data. Again, positive sentiment is expressed the most, so that the majority labelling approach will predict positive for laptop reviews.

6. Evaluation

To evaluate the effect of SOBA on the Two-Stage Hybrid Model (TSHM), we created a restaurant ontology and a laptop ontology with our ontology builder. We then used these semi-automatically created ontologies in TSHM to test their performance. In this section, we first present our evaluation framework, after which we discuss our results.

6.1. Evaluation framework

We evaluate our semi-automatically built ontologies against our benchmark ontologies, the manually built ontologies of [9], in terms of efficiency and effectiveness.

Efficiency is measured by comparing the amount of time required for human input to build the ontologies semi-automatically, with the time used to build our benchmark ontologies. Apart from the building time, we also take into account the sizes of the ontologies.

We evaluate the effectiveness of our ontologies for sentiment classification by using our semi-automatically built ontologies in the four models from [9], namely the two TSHM variants (Ont and Ont+BoW), and the two SVM algorithms (BoW and BoW+Ont). For each model, we measure the in-sample and out-of-sample accuracy in predicting the sentiment, using the SemEval 2016 restaurant and laptop reviews. Furthermore, we also apply a ten-fold cross-validation, and measure the average accuracy to perform a two-sided t-test, which is used to determine whether one model performs significantly different than another.

We then compare the accuracy of these four models when using the SOBA ontologies with the performance of the models when using the benchmark ontologies, to see whether the semi-automatically built ontologies outperform the manual ontologies in terms of effectiveness.

For both domains we run the SOBA ontology twice: once without and once with the *sense* property incorporated in TSHM. This way, we can measure whether using the dictionary meaning of concepts will lead to a significant difference in performance of the TSHM compared to not including dictionary semantics.

6.2. Evaluation Results

In this section, we present the results of the optimization processes for the parameters used in the Semi-automated Ontology Builder for Aspect-based sentiment analysis (SOBA), as described in Section 6.2.1.

Furthermore, we compare our semi-automatically built ontologies for the restaurant and laptop domains to the benchmark ontologies. We evaluate the ontologies in terms of efficiency and effectiveness for aspect-based sentiment analysis, as explained in Section 6.2.2 and Section 6.2.3, respectively.

6.2.1. Parameter Optimization

For the highest performance of SOBA, we optimized several parameters, namely $\alpha, \beta, \mu_n, \mu_v$, and μ_a, which are used in the term selection process, and for the parameter t, which is used for finding parent classes. For this, we performed the term and parent selection steps using the restaurant reviews of the Yelp Dataset Challenge [44]. Figure 9 shows that α and β are optimized for the values 0.3 and 0.7, respectively, which corresponds to an acceptance ratio of terms of 0.272.

After optimizing α and β, we optimize the fractions of verbs, nouns, and adjectives to suggest (μ_v, μ_n, and μ_a), the results being shown in Figures 10, 11, and 12. The left graphs show that the number of accepted words strictly increases with the fraction of suggested terms. This is logical, as the more terms are suggested, the more words there are to accept. However, this trend does not hold for the acceptance ratio of terms.

Moreover, the right graphs show the harmonic means for the different values of the fractions of suggested verbs, nouns, and adjectives, respectively. The maximum value of this mean is attained at a fraction of 0.16 for both verbs and nouns, which corresponds to suggesting the top 16% of nouns and verbs with highest fraction scores. For adjectives,

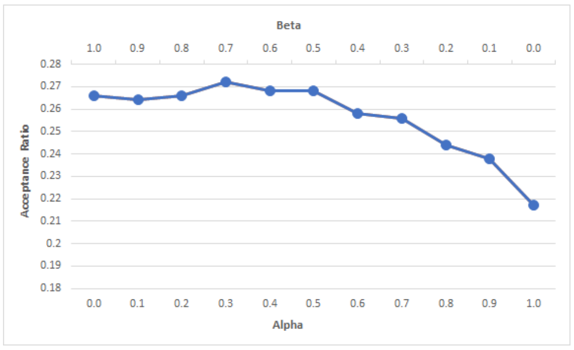


Figure 9: Acceptance ratio of terms for different values of α and β

the mean is optimal at $\mu_a = 0.20$, the maximum value of the considered range.

Last, we optimize the value of threshold t. Figure 13 shows the overall acceptances and acceptance ratio for different values of t, which seem to indicate that there is a trade-off for the two variables: the overall number of acceptances is monotonously decreasing for increasing t, whereas the acceptance ratio mostly increases with t. Note that for a threshold of 0.7 or higher, the number of acceptances and the acceptance ratio do not change, since none of the parent scores reach 0.7. Furthermore, the right figure shows that the maximum is attained for the harmonic mean at $t = 0.5$.

Setting the above mentioned parameters to the optimal values, we then run the ontology builder for the restaurant and laptop domains.

6.2.2. Building Time and Ontology Sizes

The manual restaurant and laptop ontologies of [9] were built in four and five hours, respectively. However, building the restaurant ontology using SOBA only took less than three hours, of which one-and-a-half hours of user input was required. The laptop ontology on the other hand, was built in about six hours, of which two-and-a-half hours of user input. Therefore, SOBA decreases the amount of human time needed to build the ontologies considerably, by 50% or more for both domains.

The next two tables show the sizes of the manual and SOBA ontologies for the restaurant and laptop domain. We compare the size of the ontologies based on the number of classes, the number of lexicalizations, and the number of axioms for type-3 concepts. Specifically, we also count the number of synonyms and the number of quality phrases found in the ontologies, to evaluate the relevance of these incorporated components in SOBA.

Table 1 shows that the SOBA restaurant ontology contains almost 30% more classes than the manual restaurant ontology, and the number of lexicalizations and the number of axioms tripled. Furthermore, 81.2% of the lexicalizations of the SOBA ontology are synonyms, while this is only 56.3% for the manual ontology. It is noteworthy that there are less quality phrases in the SOBA ontology than in the manual ontology, despite the larger total size of the SOBA ontology. This could possibly be explained by the relatively low frequency values of quality phrases, compared to terms that only consist of one word, so that quality phrases are less

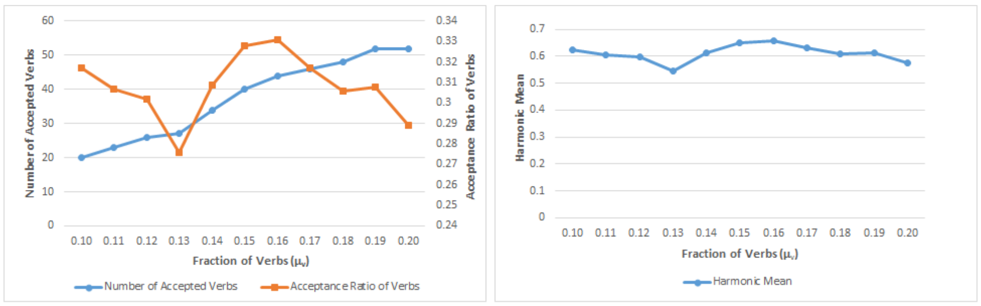


Figure 10: Number of accepted verbs and acceptance ratio of verbs (left), and the harmonic mean of the number of accepted verbs and acceptance ratio of verbs (right), for different values of the fraction of verbs suggested

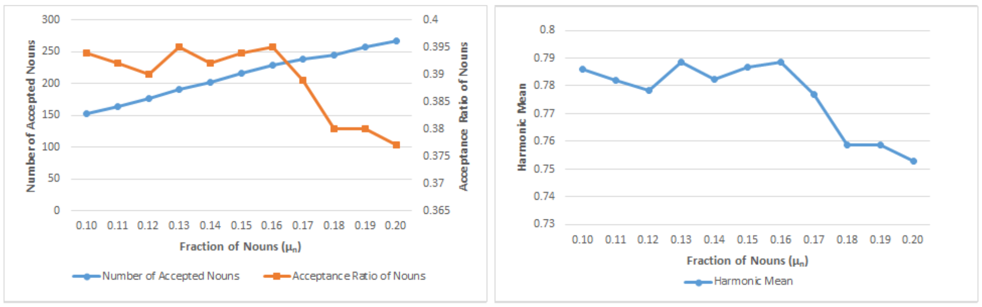


Figure 11: Number of accepted nouns and acceptance ratio of nouns (left), and the harmonic mean of the number of accepted nouns and acceptance ratio of nouns (right), for different values of the fraction of nouns suggested

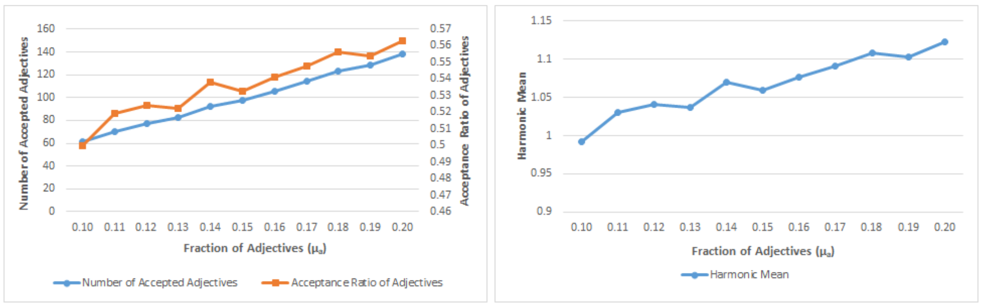


Figure 12: Number of accepted adjectives and acceptance ratio of adjectives (left), and the harmonic mean of the number of accepted adjectives and acceptance ratio of adjectives (right), for different values of the fraction of adjectives suggested

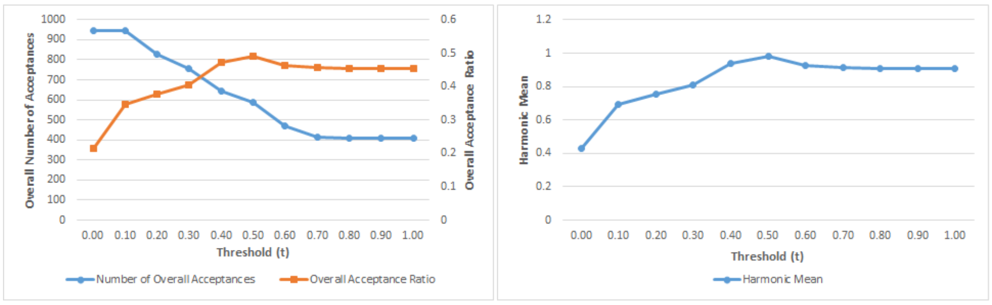


Figure 13: Number of overall accepted terms and parent-child relations and overall acceptance ratio (left), and the harmonic mean of the overall number of acceptances and overall acceptance ratio (right), for different values of the threshold

likely to be selected by SOBA during the term suggestion step.

Table 2 shows the sizes of the laptop ontologies. The manual laptop ontology is considerably smaller than the manual restaurant ontology and contains no axioms, while the SOBA laptop ontology is larger than the SOBA restaurant ontology. Furthermore, the number of classes of the SOBA laptop ontology is more than three times larger than that of the manual ontology, while the number of lexicalizations is more than six times larger. Relatively less quality phrases but more axioms are found in the SOBA laptop ontology, as compared to the SOBA restaurant ontology.

Table 1: Size of the manual and SOBA restaurant ontologies

	Manual Ont.	SOBA Ont.
Classes	365	470
Lexicalizations	750	2175
Axioms	21	63
Synonyms	422	1766
Quality Phrases	16	6

Table 2: Size of the manual and SOBA laptop ontologies

	Manual Ont.	SOBA Ont.
Classes	200	622
Lexicalizations	357	2263
Axioms	0	161
Synonyms	235	1823
Quality Phrases	0	2

6.2.3. Performance Ontologies in Two-Stage Hybrid Model

The performances of the different ontologies for the different domains are given in the next tables. Table 3 and Table 4 show the accuracy of the four models of [9] for the restaurant and laptop domain, respectively, when using the manual ontologies. For the restaurant domain, we observe that the Two-Stage Hybrid Model with SVM as backup stage, the Ont+BoW model, performs best, which agrees with the findings of [9].

For the laptop domain, the BoW, Ont+BoW and BoW+Ont model do not significantly differ in accuracy. Furthermore, the Ont model performs especially bad for the laptop reviews. The in-sample accuracy of 56.2% and out-of-sample accuracy of 60.0% are close to the fractions of positive reviews of the laptop training and test datasets, which are 56.1% and 59.6%, respectively (see Figure 8). Therefore, the Ont model barely performs better than always using majority labelling.

It generally holds that the accuracies for the laptop reviews are lower than the accuracies for restaurant reviews. This can be explained by the large number of categories that are considered in the laptop domain, which makes aspect-based sentiment analysis much more challenging. Moreover, the combination of many aspect categories and limited number of reviews in the training dataset results in relatively little or even no training data for some aspect categories. This could contribute to the lower performance on the laptop domain by the models that use machine learning techniques.

Table 5 and Table 6 show the performance of the four models of [9] with the SOBA ontology for the restaurant domain, with and without the dictionary meaning of the concepts incorporated in the TSHM, respectively, so that we can evaluate the additional value of considering dictionary semantics. We notice that the performances of the two Two-Stage Hybrid Model alternatives, Ont and Ont+BoW, are lower when using the SOBA ontology, as compared to when using the manual ontologies (see Table 3). Furthermore, comparing Table 5 with Table 6 suggests that taking into account dictionary semantics generally increases the performance of the models, for both in-sample, out-of-sample and cross-validation accuracies.

Moreover, Tables 7 and 8 give the accuracies of TSHM with the SOBA ontology for the laptop domain, with and without dictionary semantics incorporated, respectively. We notice that the Ont and Ont+BoW models perform slightly better with the SOBA ontology than with the manual ontology (see Table 4) based on the cross-validation accuracies. Moreover, for both models, the in-sample and out-of-sample performances slightly increase when dictionary semantics is included. This also holds for the cross-validation accuracy of the Ont+BoW model, but not for the Ont model, where including dictionary semantics results in a very slight decrease of performance.

We perform a two-sided t-test to test for significant differences between the performance of the various ontologies in the two variants of the TSHM (Ont and Ont+BoW). We opt for Welch's t-test since we have two independent samples with unequal variances. The samples are independent as the training set is split into ten subsamples randomly for each ontology for the ten-fold cross-validation, and therefore not the same subsamples

Table 3: Performance of the four models using the manual benchmark restaurant ontology

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	75.3%	73.3%	73.7%	0.0725			
BoW	73.2%	90.2%	77.5%	0.0613	0.0160*		
Ont+BoW	82.0%	89.3%	82.1%	0.0467	2.25E-5***	3.15E-4***	
BoW+Ont	79.7%	90.3%	78.3%	0.0636	0.0157*	0.129	0.00776**

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 4: Performance of the four models using the manual benchmark laptop ontology

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	60.0%	56.2%	56.2%	0.0701			
BoW	72.2%	86.5%	78.2%	0.0411	5.79E-6***		
Ont+BoW	72.0%	86.5%	78.2%	0.0429	5.15E-6***	0.969	
BoW+Ont	72.3%	86.8%	77.5%	0.0479	8.95E-6***	0.182	0.155

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 5: Performance of the four models using the semi-automatically built restaurant ontology without *sense* feature

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	72.8%	70.8%	70.9%	0.0504			
BoW	73.7%	86.6%	77.8%	0.0380	0.00132**		
Ont+BoW	79.0%	84.0%	79.7%	0.0295	3.65E-5***	0.0418*	
BoW+Ont	79.0%	86.6%	77.4%	0.0502	0.0202*	0.776	0.172

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 6: Performance of the four models using the semi-automatically built restaurant ontology with *sense* feature

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	72.8%	70.9%	71.1%	0.0534			
BoW	78.9%	86.9%	79.0%	0.0366	8.30E-4***		
Ont+BoW	80.7%	84.6%	80.0%	0.0268	9.35E-5***	0.0966	
BoW+Ont	79.7%	87.2%	79.3%	0.0376	5.51E-4***	0.339	0.216

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 7: Performance of the four models using the semi-automatically built laptop ontology without *sense* feature

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	61.9%	61.6%	61.7%	0.0475			
BoW	71.0%	84.0%	78.4%	0.0352	1.67E-5***		
Ont+BoW	70.3%	79.7%	76.5%	0.0401	6.87E-5***	0.00178**	
BoW+Ont	73.3%	85.0%	79.1%	0.0299	1.97E-5***	0.395	0.0275*

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 8: Performance of the four models using the semi-automatically built laptop ontology with *sense* feature

	out-of-sample accuracy	in-sample accuracy	cross-validation accuracy	cross-validation st. dev.	paired two-sided t-test (p-values) Ont	BoW	Ont+BoW
Ont	61.9%	61.6%	61.3%	0.0319			
BoW	71.9%	84.2%	78.2%	0.0579	4.66E-5***		
Ont+BoW	70.5%	80.1%	78.6%	0.0277	1.59E-5***	0.163	
BoW+Ont	74.4%	84.2%	76.8%	0.0295	2.11E-5***	0.558	0.0590

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 9: Welch's two-sided t-tests to test the difference between the performance of the various ontologies when used in the Ont model for the restaurant and laptop domain

		out-of-sample accuracy	in-sample accuracy	cross-validation accuracy	cross-validation st. dev.	Welch's two-sided t-test (p-values) MO	SO
Restaurant	MO	75.3%	73.3%	73.7%	0.0725		
	SO	72.8%	70.8%	70.9%	0.0504	0.329	
	SO + sense	72.8%	70.9%	71.1%	0.0534	0.373	0.932
Laptop	MO	60.0%	56.2%	56.2%	0.0701		
	SO	61.9%	61.6%	61.7%	0.0475	0.0548	
	SO + sense	61.9%	61.6%	61.3%	0.0319	0.0506	0.828

Notes:
1. ***: p<0.001, **: p< 0.01, *: p<0.05
2. MO: manual ontology, SO: SOBA ontology without semantics in TSHM, SO+sense: SOBA ontology with semantics in TSHM

Table 10: Welch's two-sided t-tests to test the difference between the performance of the various ontologies when used in the Ont+BoW model for the restaurant and laptop domain

		out-of-sample accuracy	in-sample accuracy	cross-validation accuracy	cross-validation st. dev.	Welch's two-sided t-test (p-values) MO	SO
Restaurant	MO	82.0%	89.3%	82.1%	0.0467		
	SO	79.0%	84.0%	79.7%	0.0295	0.186	
	SO + sense	80.7%	84.6%	80.0%	0.0268	0.233	0.815
Laptop	MO	72.0%	86.5%	78.2%	0.0429		
	SO	70.3%	79.7%	76.5%	0.0401	0.372	
	SO + sense	70.5%	80.1%	78.6%	0.0277	0.807	0.190

Notes:
1. ***: p<0.001, **: p< 0.01, *: p<0.05
2. MO: manual ontology, SO: SOBA ontology without semantics in TSHM, SO+sense: SOBA ontology with semantics in TSHM

are used for the different ontologies. However, note that an unpaired t-test will generally have lower power than the paired t-test [47], and that the latter is used to compare the performance of the four models of [9] per ontology.

Table 9 and Table 10 show the p-values for the two-sided t-tests with different ontologies, respectively for the Ont and Ont+BoW model. For both models, the results suggest no significant difference in accuracy between when the manual ontologies (MO) are used or the SOBA ontologies (SO and SO+sense).

Furthermore, the tables also indicate that incorporating dictionary semantics does not have a significant effect on the accuracy of TSHM. It should, however, be noted that the positive effect that we observe is insignificant possibly due to the inaccuracy of the word sense disambiguation phase [48].

Table 11: Performance of the four models using the manual benchmark restaurant ontology (2015 data)

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	75.3%	75.3%	73.8%	0.0583			
BoW	79.3%	78.3%	78.7%	0.0598	0.00641**		
Ont+BoW	84.5%	83.1%	82.9%	0.0431	4.47E-5***	2.32E-4***	
BoW+Ont	79.0%	78.9%	78.3%	0.0642	0.0140*	0.643	0.00176**

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 12: Performance of the four models using the semi-automatically built restaurant ontology without *sense* feature (2015 data)

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	59.7%	76.3%	76.3%	0.0494			
BoW	76.7%	88.2%	78.3%	0.0371	0.283		
Ont+BoW	77.3%	85.6%	80.3%	0.0374	0.0223*	0.0710	
BoW+Ont	73.5%	88.7%	78.6%	0.0382	0.194	0.617	0.0354*

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 13: Performance of the four models using the semi-automatically built restaurant ontology with *sense* feature (2015 data)

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	72.9%	72.9%	71.4%	0.0714			
BoW	79.6%	79.2%	79.2%	0.0457	0.00717**		
Ont+BoW	81.1%	81.0%	80.7%	0.0363	5.32E-4***	0.0555	
BoW+Ont	79.3%	78.8%	79.2%	0.0457	4.81E-4***	0.964	0.0168*

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

In order to check the validity of our results on a different data set, we also analyze the performance of SOBA using SemEval 2015 restaurant data (254 reviews and 1315 sentences training data, 96 reviews, and 685 sentences test data), and SemEval 2015 laptop data (277 reviews and 1739 sentences training data, 173 reviews, and 761 sentences test data) [49].

Tables 11 to 13 show the performance of the four models of [9] using the manual ontology, the SOBA ontology without *sense* feature, and the SOBA ontology with *sense* feature for the restaurant domain. The results for the laptop domain are presented in Tables 14 to 16, which respectively show the performance of the manual ontology and the SOBA ontology without and with *sense* feature.

For the restaurant domain, we see that the Ont+BoW method consistently performs best (Tables 11 to 13), which is in line with 2016 results. The same, however, does not hold for the laptop domain based on 2015 results, which show that the BoW and BoW+Ont models perform equally well (Table 14) or better (Tables 15 and 16) than the Ont+BoW model.

Moreover, it is noteworthy for the laptop domain that the low quality of the manual ontology results in the Ont model always opting for the backup method of predicting positive; the out-of-sample and in-sample accuracies are exactly equal to the percentage of positive reviews in the test (57.0%) and training (55.9%) data [49]. Consequently, the Ont+BoW and BoW+Ont models completely rely on the BoW model, resulting in the same performance of the three models.

Again, we apply a two-sided Welch's t-test to test for significant difference between the different approaches (see Table 17 and 18). We see that similar to the SemEval 2016 data set, the performance

Table 14: Performance of the four models using the manual laptop ontology (2015 data)

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	57.0%	55.9%	55.8%	0.0700			
BoW	74.7%	98.7%	81.1%	0.0286			
Ont+BoW	74.7%	98.7%	81.1%	0.0286			
BoW+Ont	74.7%	98.7%	81.1%	0.0286			

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 15: Performance of the four models using the semi-automatically built laptop ontology without *sense* feature (2015 data)

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	63.5%	60.8%	61.3%	0.0897			
BoW	74.7%	90.1%	81.8%	0.0400	3.73e-4***		
Ont+BoW	74.2%	82.6%	77.6%	0.0314	6.09E-4***	3.49E-4***	
BoW+Ont	75.0%	90.8%	82.6%	0.0397	2.91E-4***	0.349	0.00226**

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 16: Performance of the four models using the semi-automatically built laptop ontology with *sense* feature (2015 data)

	out-of-sample	in-sample	cross-validation		paired two-sided t-test (p-values)		
	accuracy	accuracy	accuracy	st. dev.	Ont	BoW	Ont+BoW
Ont	64.3%	61.2%	61.3%	0.0657			
BoW	73.6%	90.4%	81.3%	0.0452	4.18E-6***		
Ont+BoW	72.8%	98.9%	77.8%	0.0323	4.91E-6***	0.0036**	
BoW+Ont	75.4%	91.3%	82.5%	0.0361	1.10E-6***	0.120	2.66E-5***

Notes:
***: p<0.001, **: p< 0.01, *: p<0.05

Table 17: Welch's two-sided t-tests to test the difference between the performance of the various ontologies when used in the Ont model for the restaurant and laptop domain (2015 data)

		out-of-sample	in-sample	cross-validation		Welch's two-sided t-test (p-values)	
		accuracy	accuracy	accuracy	st. dev.	MO	SO
Restaurant	MO	75.3%	75.3%	73.8%	0.0583		
	SO	59.7%	76.3%	76.3%	0.0494	0.315	
	SO + sense	72.9%	72.9%	71.4%	0.0714	0.421	0.0912
Laptop	MO	57.0%	55.9%	55.8%	0.0700		
	SO	63.5%	60.8%	61.3%	0.0897	0.144	
	SO + sense	64.3%	61.2%	61.3%	0.0657	0.0868	1.00

Notes:
1. ***: p<0.001, **: p< 0.01, *: p<0.05
2. MO: manual ontology, SO: SOBA ontology without semantics in TSHM, SO+sense: SOBA ontology with semantics in TSHM

of the models using the manual ontology generally does not significantly differ from the models when using SOBA ontologies. The only exception is for the laptop domain for the model Ont+BoW, but this can be explained by the exceptional high performance when the manual ontology is used, as it then completely relies on the BoW algorithm. Moreover, the t-tests show that incorporating dic-

Table 18: Welch's two-sided t-tests to test the difference between the performance of the various ontologies when used in the Ont+BoW model for the restaurant and laptop domain (2015 data)

		out-of-sample accuracy	in-sample accuracy	cross-validation accuracy	st. dev.	Welch's two-sided t-test (p-values) MO	SO
Restaurant	MO	84.5%	83.1%	82.9%	0.0431		
	SO	77.3%	85.6%	80.3%	0.0374	0.167	
	SO + sense	81.1%	81.0%	80.7%	0.0363	0.233	0.811
Laptop	MO	74.7%	98.7%	81.1%	0.0286		
	SO	74.2%	82.6%	77.6%	0.0314	0.0179*	
	SO + sense	72.8%	98.9%	77.8%	0.0323	0.0264*	0.890

Notes:
1. ***: p<0.001, **: p< 0.01, *: p<0.05
2. MO: manual ontology, SO: SOBA ontology without semantics in TSHM, SO+sense: SOBA ontology with semantics in TSHM

Table 19: Ranks of TSHM (Ont+BoW model) with SOBA ontology in top of SemEval 2015 and SemEval 2016 ranking (restaurant domain)

2015 Team	Accuracy	2016 Team	Accuracy
MO	**84.5**	XRCE	88.1
SO+sense	**81.1**	IIT-T	86.7
sentiue	78.7	NileT	85.5
ECNU	78.1	IHS-R	83.9
SO	**77.3**	ECNU	83.6
Isislif	75.5	AUEB	83.2
LT3	75.0	**MO**	**82.0**
UFRGS	71.7	**SO+sense**	**80.7**
wnlp	71.4	**SO**	**79.0**

Table 20: Ranks of TSHM (Ont+BoW model) with SOBA ontology in top of SemEval 2015 and SemEval 2016 ranking (laptop domain)

2015 Team	Accuracy	2016 Team	Accuracy
Sentiue	79.3	IIT-T	82.8
ECNU	78.3	INSIG	78.4
Lsislif	77.9	ECNU	78.2
MO	**74.7**	IHS-R	77.9
ECNU	74.5	NileT	77.4
SO	**74.2**	AUEB-	76.9
LT3	73.8	**MO**	**72.0**
TJUdeM	73.2	**SO+sense**	**70.5**
SO+sense	**72.8**	**SO**	**70.3**

tionary semantics again does not have a significant effect on the accuracy of TSHM.

Last, Table 19 and Table 20 show the performance of the TSHM (Ont+BoW model) using SOBA ontologies in comparison to the top 6 best performing models amongst the SemEval 2015 and SemEval 2016 submissions. While our goal is not to compete in terms of accuracy with the top 6 best performing models, as we focus on the semi-automatic building of the sentiment domain ontology, we have provided these accuracies as a reference (emphasizing the TSHM accuracies).

7. Conclusion

This research examined the possibility to improve the performance of knowledge-driven aspect-based sentiment analysis (ABSA) models by making the knowledge repository, the employed ontology, more extensive, while reducing the amount of human effort needed to build the ontology. We do this by introducing a Semi-automated Ontology Builder for Aspect-based sentiment analysis (SOBA), which semi-automates the ontology building process, requiring minimal user input to control for possible mistakes of the ontology builder thus ensuring the quality of the resulting ontology. Furthermore, SOBA incorporates information from a semantic lexicon in its ontologies. We then created ontologies using SOBA for two domains, the restaurant and laptop domains, and evaluated the performance of the SOBA ontologies in the knowledge-driven Two-Stage Hybrid Model (TSHM) [9], a state-of-the-art ABSA framework. Based on these results, we present the following three conclusions.

First, we conclude that using a semi-automated ontology builder substantially decreases the human time needed to construct the ontology, therefore making ABSA frameworks that use an ontology more efficient. The results show that SOBA reduces the human time needed to build the ontologies by 50% or more for both domains, when comparing the building time of the SOBA ontologies with the manual benchmark ontologies of [9].

Second, SOBA does not lead to more effective ABSA. Comparing the performance of the SOBA ontologies in TSHM with the performance of the benchmark ontologies, we show that the SOBA ontologies do not perform significantly differently than the benchmark ontologies.

Last, we conclude that considering a semantic lexicon in ontologies generally increases the performance of the TSHM. However, the observed differences are not significant.

Overall, SOBA semi-automatically creates ontologies of comparable quality as that of manual ontologies, thereby making the overall process of using the Two-Stage Hybrid Model of [9] more efficient. It is therefore advisable to use SOBA, as it considerably decreases the human time and effort required to construct ontologies.

As future research, we suggest to investigate ways to improve the quality of the constructed ontologies. For example, one could adapt the term and parent suggestion steps by incorporating term semantics from a semantic lexicon, which could help improve the measurement of the relative importance of words for a domain and the degree of co-occurrence between concepts [12].

Another point of improvement would be to also consider terms and phrases that are not defined in WordNet during the term suggestion step of SOBA, as those terms could be highly informative for determining the expressed sentiment. Unofficial words, such as slang, but also urban expressions, deliberate misspellings and similar phenomena that are ubiquitous in online reviews, can be used to enrich the SOBA ontologies, leading to more accurate sentiment classifications.

Our last suggestion is to examine the influence of the accuracy of the word sense disambiguation (WSD) phase on the effect of including dictionary semantics. Future research could consider using more advanced WSD algorithms with higher accuracy, to analyze if this could lead to a significant positive effect in ABSA.

References

[1] B. Pang, L. Lee, et al., Opinion Mining and Sentiment Analysis, Foundations and Trends in Information Retrieval 2 (1–2) (2008) 1–135.

[2] B. Liu, Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, Cambridge University Press, 2015.

[3] K. Schouten, F. Frasincar, Survey on Aspect-Level Sentiment Analysis, IEEE Transactions on Knowledge and Data Engineering 28 (3) (2016) 813–830.

[4] Q. Liu, H. Zhang, Y. Zeng, Z. Huang, Z. Wu, Content Attention Model for Aspect Based Sentiment Analysis, in: 27th International Conference on World Wide Web (WWW 2018), International World Wide Web Conferences Steering Committee, 2018, pp. 1023–1032.

[5] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, Learning Word Vectors for Sentiment Analysis, in: 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011), Vol. 1, ACL, 2011, pp. 142–150.

[6] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, B. Qin, Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification, in: 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Vol. 1, 2014, pp. 1555–1565.

[7] B. Schuller, T. Knaup, Learning and Knowledge-based Sentiment Analysis in Movie Review Key Excerpts, in: Toward Autonomous, Adaptive, and Context-Aware Multimodal Interfaces. Theoretical and Practical Issues, Springer, 2011, pp. 448–472.

[8] K. Schouten, F. Frasincar, F. de Jong, Ontology-Enhanced Aspect-Based Sentiment Analysis, in: 17th International Conference on Web Engineering (ICWE 2017), Springer, 2017, pp. 302–320.

[9] K. Schouten, F. Frasincar, Ontology-Driven Sentiment Analysis of Product and Service Aspects, in: 15th European Semantic Web Conference (ESWC 2018), Springer, 2018, pp. 608–623.

[10] S. Bloehdorn, P. Cimiano, S. Staab, An Ontology-Based Framework for Text Mining, LDV Forum 20 (1) (2005) 87–112.

[11] J. De Knijff, F. Frasincar, F. Hogenboom, Domain Taxonomy Learning from Text: The Subsumption Method versus Hierarchical Clustering, Data & Knowledge Engineering 83 (2013) 54–69.

[12] K. Meijer, F. Frasincar, F. Hogenboom, A Semantic Approach for Extracting Domain Taxonomies from Text, Decision Support Systems 62 (2014) 78–93.

[13] C. Fellbaum, WordNet: An Electronic Lexical Database, 1998.

[14] E. Cambria, S. Poria, D. Hazarika, K. Kwok, SenticNet 5: Discovering Conceptual Primitives for Sentiment Analysis by Means of Context Embeddings, in: Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI 2018), AAAI Press, 2018, pp. 1795–1802.

[15] S. Baccianella, A. Esuli, F. Sebastiani, SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining, in: 7th International Conference on Language Resources and Evaluation (LREC 2010), ELRA, 2010.

[16] T. Wilson, J. Wiebe, P. Hoffmann, Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis, Computational Linguistics 35 (3) (2009) 399–433.

[17] S. Mohammad, P. D. Turney, Crowdsourcing a Word-Emotion Association Lexicon, Computational Intelligence 29 (3) (2013) 436–465.

[18] M. Dragoni, S. Poria, E. Cambria, OntoSenticNet: A

Commonsense Ontology for Sentiment Analysis, IEEE Intelligent Systems 33 (3) (2018) 77–85.

[19] S. Poria, E. Cambria, A. F. Gelbukh, Aspect Extraction for Opinion Mining with a Deep Convolutional Neural Network, Knowledge Based Systems 108 (2016) 42–49.

[20] Y. Wang, M. Huang, X. Zhu, L. Zhao, Attention-based LSTM for Aspect-level Sentiment Classification, in: 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), ACL, 2016, pp. 606–615.

[21] D. Tang, B. Qin, T. Liu, Aspect Level Sentiment Classification with Deep Memory Network, in: 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016), ACL, 2016, pp. 214–224.

[22] Q. Liu, H. Zhang, Y. Zeng, Z. Huang, Z. Wu, Content Attention Model for Aspect Based Sentiment Analysis, in: 27th World Wide Web Conference on World Wide Web (WWW 2018), ACM, 2018, pp. 1023–1032.

[23] S. Zheng, R. Xia, Left-Center-Right Separated Neural Network for Aspect-based Sentiment Analysis with Rotatory Attention, CoRR abs/1802.00892.

[24] Y. Ma, H. Peng, E. Cambria, Targeted Aspect-Based Sentiment Analysis via Embedding Commonsense Knowledge into an Attentive LSTM, in: Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018), AAAI Press, 2018, pp. 5876–5883.

[25] T. R. Gruber, A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition 5 (2) (1993) 199–220.

[26] L. Zhao, C. Li, Ontology Based Opinion Mining for Movie Reviews, in: 3rd International Conference on Knowledge Science, Engineering and Management (KSEM 2009), Springer, 2009, pp. 204–214.

[27] M. Federici, M. Dragoni, A Knowledge-Based Approach for Aspect-Based Opinion Mining, in: Third Semantic Web Evaluation Challenge (SemWebEval 2018) at ESWC 2016, Vol. 641 of Communications in Computer and Information Science, Springer, 2016, pp. 141–152.

[28] K. Schouten, F. Frasincar, The Benefit of Concept-Based Features for Sentiment Analysis, in: Second Semantic Web Evaluation Challenge Challenge at ESWC 2015, Vol. 548 of Communications in Computer and Information Science, Springer, 2015, pp. 223–233.

[29] B. Heerschop, F. Goossen, A. Hogenboom, F. Frasincar, U. Kaymak, F. de Jong, Polarity analysis of texts using discourse structure, in: 20th ACM Conference on Information and Knowledge Management (CIKM 2011), ACM, 2011, pp. 1061–1070.

[30] M. Taboada, K. Voll, J. Brooke, Extracting Sentiment as a Function of Discourse Structure and Topicality, Tech. Rep. 20, Simon Fraser University (2008).

[31] M. Dragoni, C. da Costa Pereira, A. G. B. Tettamanzi, S. Villata, Combining Argumentation and Aspect-Based Opinion Mining: The SMACk System, AI Communications 31 (1) (2018) 75–95.

[32] E. Cambria, Affective Computing and Sentiment Analysis, IEEE Intelligent Systems 31 (2) (2016) 102–107.

[33] E. Cambria, A. Hussain, Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis, Springer, 2015.

[34] P. Buitelaar, P. Cimiano, B. Magnini, Ontology Learning from Text: An Overview, Ontology Learning from text: Methods, Evaluation and Applications 123 (2005) 3–12.

[35] Y. Park, R. J. Byrd, B. K. Boguraev, Automatic Glossary Extraction: Beyond Terminology Identification, in: 19th International Conference on Computational Linguistics (COLING 2002), Vol. 1, ACL, 2002, pp. 1–7.

[36] R. Navigli, P. Velardi, Semantic Interpretation of Terminological Strings, in: 6th International Conference on Terminology and Knowledge Engineering (TKE 2002), 2002, pp. 95–100.

[37] F. Vasilescu, P. Langlais, G. Lapalme, Evaluating Variants of the Lesk Approach for Disambiguating Words, in: 4th International Conference on Language Resources and Evaluation (LREC 2004), ACL, 2004.

[38] A. Kilgarriff, J. Rosenzweig, English Senseval: Report and Results, in: 2nd International Conference on Language Resources and Evaluation, (LREC 2000), Vol. 6, ACL, 2000, p. 2.

[39] A. Hogenboom, P. van Iterson, B. Heerschop, F. Frasincar, U. Kaymak, Determining Negation Scope and Strength in Sentiment Analysis, in: IEEE International Conference on Systems, Man and Cybernetics 2011 (SMC 2011), IEEE, 2011, pp. 2589–2594.

[40] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, C. Potts, Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank, in: 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), 2013, pp. 1631–1642.

[41] J. Liu, J. Shang, C. Wang, X. Ren, J. Han, Mining Quality Phrases from Massive Text Corpora, in: 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD 2015), ACM, 2015, pp. 1729–1744.

[42] M. Sanderson, B. Croft, Deriving Concept Hierarchies from Text, in: 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SGIR 1999), ACM, 1999, pp. 206–213.

[43] C. Cesarano, B. Dorr, A. Picariello, D. Reforgiato, A. Sagoff, V. Subrahmanian, Oasys: An Opinion Analysis System, in: AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs 2006 (CAAW 2006), AAAI Press, 2006, pp. 21–26.

[44] Yelp, Yelp Dataset Challenge, retrieved 11-06-2017, from https://www.yelp.nl/ dataset_challenge. (2017).

[45] R. He, J. McAuley, Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering, in: 25th International Conference on World Wide Web (WWW 2016), International World Wide Web Conferences Steering Committee, 2016, pp. 507–517.

[46] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, A.-S. Mohammad, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, et al., SemEval-2016 Task 5: Aspect Based Sentiment Analysis, in: 10th International Workshop on Semantic Evaluation (SemEval 2016), ACL, 2016, pp. 27–35.

[47] D. C. Martin, P. Diehr, E. B. Perrin, T. D. Koepsell, The Effect of Matching on the Power of Randomized Community Intervention Studies, Statistics in Medicine 12 (3-4) (1993) 329–338.

[48] R. Navigli, Word Sense Disambiguation: A Survey, ACM Computing Surveys 41 (2) (2009) 10:1–10:69.

[49] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, I. Androutsopoulos, Semeval-2015 Task 12: Aspect Based Sentiment Analysis, in: 9th International Workshop on Semantic Evaluation (SemEval 2015), ACL, 2015, pp. 486–495.