Description Logic Class Expression Learning Applied to Sentiment Analysis

Alberto Salguero and Macarena Espinilla

Abstract Description Logic (DL) Class Expression Learning (CEL) is a recent research topic of interest in the field of machine learning. Given a set of positive and negative examples of individuals in an ontology, the learning problem consists of finding a new class expression or concept such that most of the positive examples are instances of that concept, whereas the negatives examples are not. Therefore, the class expression learning can be seen as a search process in the space of concepts. In this chapter, the use of CEL algorithms is proposed as a tool to find the class expression that describes as much of the instances of positive documents as possible, being the main novelty of the proposal that the ontology is focused on inferring knowledge at syntactic level to determine the orientation of opinion. Furthermore, the use of CEL algorithms can be an alternative to complement other types of classifiers for sentiment analysis, incorporating such description classes as relevant new features into the knowledge base. To do so, an ontology-based text model for the representation of text documents is presented. The process for the ontology population and the use of the class expression learning of sentiment concepts are also described. To show the usefulness and effectiveness of our proposal, we use a set of documents about positive feedback focused on films to learn the positive sentiment concept and to classify the documents, comparing the results obtained against the result obtained by a C4.5 decision tree classifier, using the standard bag of words structure. Finally, we describe the problems that have arisen and solutions that have been adopted in our proposal.

Keywords Description logic class expression learning · Ontology-based text model · Text mining · Sentiment analysis

A. Salguero (✉)
Computer Sciences Department, University of Cádiz, Cádiz, Spain
e-mail: alberto.salguero@uca.es

M. Espinilla
Computer Sciences Department, University of Jaén, Jaén, Spain
e-mail: mestevez@ujaen.es

W. Pedrycz and S.-M. Chen (eds.), *Sentiment Analysis and Ontology Engineering*,
Studies in Computational Intelligence 639, DOI 10.1007/978-3-319-30319-2_5

1 Introduction

Sentiment analysis, also called opinion mining, is a research area that analyzes people's opinions or sentiments such as products, services, organizations, individuals, issues, events, topics, and their attributes Refs. [8, 18]. In recent years there is a huge amount of research related to the extraction and analysis of the opinions like subjectivity detection, opinion extraction, irony detection, etc. However, among these areas, the process to determine opinions, which express or imply positive or negative sentiments, is becoming an area increasingly important Refs. [9, 22].

Several proposals have been presented in the literature to determine positive or negative opinions, which can be classified into two different groups Ref. [18]. On the one hand, supervised methodologies that use machine learning algorithms, specific to text mining, in which training data exist. This kind of methodology requires large datasets for training and learning that, due to the novelty of the research area, it is sometimes difficult to obtain. On the other hand, unsupervised methodologies that use resources such as dictionaries or lexical ontologies. Therefore, it is necessary linguistic resources, which generally depend on the language, that are used to determine the orientation of opinion.

Furthermore, there are hybrid methodologies that combine both groups. The main advantage of the hybrid methodology is that takes advantage of the strengths of both methodologies without allowing them to get in each others way Ref. [20]. Usually, the hybrid methodologies are focused on to apply learning algorithms (supervised methodology) with a set of attributes that are obtained by means of linguistic resources (unsupervised methodology) in order to improve the results.

The standard Bag Of Words (BOW) is typically used to classify textual data when using supervised learning techniques. With the standard BOW approach, each document is transformed into a vector containing a set of features, which usually represent the frequency of each term. There exist many algorithms that try to improve the quality of the classifiers based on this technique by learning more features Ref. [26]. However, those new features are obtained using an *ad hoc* application that usually works for a very limited domain.

The algorithms for Class Expression Learning (CEL) are mainly used in the field of ontology engineering. They can be used to suggest new class descriptions that are relevant for the problem while the ontologies are being developed. So, CEL is a recent research topic of interest in the field of machine learning that can be used as an learning tool in order to improve the obtained results. In this chapter, we propose the use of CEL algorithms as a useful tool to find the class expression that has as much of the instances of positive documents or sentences as possible. So, given a set of positive and negative examples of individuals in an ontology, the learning problem consists on finding a new class expression or concept such that most of the positive examples are instances of that concept, whereas the negatives examples are not.

Therefore, we propose the use of the Description Logic (DL) Class Expression Learning technique for sentiment classification with the main advantage that offers a very flexible approach that can be used to classify textual data in many domains

without the need of developing *ad hoc* applications. To do so, first, we present an ontology-based text model that contains the concepts and properties needed in order to represent the text documents, presenting also a procedure to generate the ontology population. Finally, the CEL technique is used to find the class expression that represents as many of the positive documents as possible.

The main novelty of the proposal, in contrast to most of the ontology-based text analyzers, is that our proposal is focused on inferring knowledge at syntactic level to learn an equivalent description for the positive or negative opinions. So, the reasoning capabilities of ontologies are used in this contribution to infer hidden features and relationships among terms with the advantage of that not all the relationships among terms need to be explicitly recorded due to the fact that they can be inferred while traversing the space of concepts.

To illustrate our proposal, we apply the CEL technique to a set of documents containing opinions about films, so we get as results DL class descriptions that describe documents expressing positive opinions about films, being used to classify the documents. Furthermore, we compare the results obtained by the CEL-based classifier against the result obtained by a C4.5 decision tree classifier that uses the usual BOW structure.

The chapter is structured as: Sect. 2 provides a brief introduction to ontologies as well as some related concepts and tools that will be used throughout the content of this chapter. Section 3 introduces the CEL problem. Section 4 proposes a procedure for applying CEL techniques in the field of sentiment analysis. Section 5 presents the results obtained by the CEL-based classifier applied to sentiment analysis and compares the results obtained by the CEL-based classifier with respect to a C4.5 decision tree classifier that uses the standard BOW approach. Section 6 the encountered problems and the solutions that have been adopted are discussed in this section. Finally, in Sect. 7, conclusions and future works are pointed out.

2 Foundations of Ontologies

Ontologies are used to provide structured vocabularies that explain the relationship among terms, allowing an unambiguous interpretation of their meaning. So, ontologies are formed by concepts (or classes) which are usually organized into a hierarchy of concepts Refs. [1, 27], being the ontologies more complex than taxonomies because they not only consider type-of relations, but they also consider other relations, including part-of or domain-specific relations Ref. [10].

The main advantage of the ontologies is that they codify knowledge and make it reusable by people, databases, and applications that need to share information Refs. [10, 28]. Due to this fact, the construction, the integration and the evolution of ontologies have been critical for the so-called Semantic Web Refs. [4, 6, 11, 19]. However, obtaining a high quality ontology largely depends on the availability of well-defined semantics and powerful reasoning tools.

The ontologies have been applied to sentiment analysis, obtaining successful results in several applications. So, in Ref. [13], the aspect-oriented sentiment analysis is studied, learning fuzzy product ontologies. In Ref. [17] was proposed an ontology-based sentiment analysis of network public opinions by applying semantic web technology. An ontology-based linguistic model was presented in Ref. [29] to identify the basic appraisal expression in Chinese product by mapping product features and opinions to the conceptual space of the domain ontology. In Ref. [12] was proposed an ontology-based sentiment analysis of twitter posts, computing a sentiment grade in the post. Furthemore, in Ref. [3] was presented an approach that shows how to automatically mine positive and negative sentiments with an ontological filtering. So, we can see that the proposals presented in the literature are focused on the representation and the analysis only at semantic level instead of our proposal that will be focused on syntactic level to learn class description for positive or negative opinions.

Regarding semantic web, a formal language is OWL Refs. [7, 24] that was developed by the World Wide Web Consortium (W3C) in the Web Ontology Working Group. Originally, OWL was designed to represent information about categories of objects and how objects are related. OWL inherits characteristics from several representation languages families, including the Description Logics and Frames basically. Furthermore, OWL shares many characteristics with Resource Description Framework (RDF), the W3C base for the Semantic Web. The major extension over RDF Schema (RDFS) is that OWL has the ability to impose restrictions on properties for certain classes.

The design of OWL is greatly influenced by Description Logics (DL), particularly in the formalism of semantics, the choice of language constructs and the integration of data types and data values. In fact, OWL DL and OWL Lite (subsets of OWL) is seen as expressive DL, offering a DL knowledge base equivalent ontology. They are in fact extensions of the DL "Attributive Concept Language with Complements" ($\mathcal{ALC}$). More formally, let N_C, N_R and N_O be (respectively) sets of "concept names", "role names" (also known as "properties") and "individual names". The semantics of DL are defined by interpreting concepts as sets of individuals and roles as sets of ordered pairs of individuals.

The Manchester OWL Syntax is derived from the OWL Abstract Syntax, but is less verbose and minimises the use of brackets. This means that it is quicker and easier to read and write by humans than DL formal syntax Ref. [5]. We introduce the Manchester OWL Syntax in Definition 2.1 because the solutions found by the tool we will use to solve the CEL problem are expressed in such language.

Definition 2.1 (*Terminological interpretation*) A "terminological interpretation" $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over a "signature" (N_C, N_R, N_O) for $(\mathcal{ALC})$ consists of the following concepts:

- A non-empty set $\Delta^{\mathcal{I}}$ called the "domain".
- A "interpretation function" $\cdot^{\mathcal{I}}$ that maps:

Table 1 Semantics of DL constructions

DL Syntax	Manchester syntax	Semantics
$\top^{\mathcal{I}}$	*Thing*	$\Delta^{\mathcal{I}}$
$\bot^{\mathcal{I}}$	*Nothing*	$\emptyset$
$(C \sqcup D)^{\mathcal{I}}$	C or D	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(C \sqcap D)^{\mathcal{I}}$	C and D	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$(\neg C)^{\mathcal{I}}$	not C	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(\forall R.C)^{\mathcal{I}}$	R only C	$\{x \in \Delta^{\mathcal{I}} \mid \texttt{for every } y, (x, y) \in R^{\mathcal{I}} \texttt{ implies } y \in C^{\mathcal{I}}\}$
$(\exists R.C)^{\mathcal{I}}$	R some C	$\{x \in \Delta^{\mathcal{I}} \mid \texttt{there exists } y, (x, y) \in R^{\mathcal{I}} \texttt{ and } y \in C^{\mathcal{I}}\}$

- every "individual" a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- every "concept" to a subset of $\Delta^{\mathcal{I}}$
- every "role name" to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

such that the semantics in Table 1 holds.

OWL extends $\mathcal{ALC}$ with role hierarchies, value restrictions, inverse properties, cardinality restrictions and transitive roles. One of the main advantages of high formalization of the OWL language is the possibility of using automated reasoning techniques. In 2009, the W3C proposed the OWL 2 recommendation in order to solve some usability problems detected in the previous version, keeping the base of OWL. So, OWL 2 adds several new features to OWL, some of the new features are syntactic sugar (e.g., disjoint union of classes) while others offer new expressivity, including: increased expressive power for properties, simple metamodeling capabilities, extended support for datatypes, extended annotation capabilities, and other innovations and minor features Ref. [30].

Sometimes, it is difficult to express certain kind of knowledge in OWL. In such cases, it is possible to use OWL extensions such as the Semantic Web Rule Language (SWRL) in order to include a high-level abstract syntax for Horn-like rules in OWL.[1] The SWRL is used to provide more powerful deductive reasoning capabilities than OWL alone Ref. [2]. It is important to note that several OWL reasoners, which are well known open source, support the SWRL language as Hermit[2] and Fact++.[3]

Another key tool that can be used in conjunction with OWL is SPARQL,[4] which is a query language able to retrieve and manipulate data stored in RDF triples. The main difference with SWRL is that the SPARQL language has been designed to work with RDF triples, at a lower abstraction level. While, the SWRL has been built on top of OWL, extending the set of its axioms, SPARQL is designed to work with individuals. Therefore, it is mainly used to retrieve individuals meeting certain conditions, moreover, it can also be used to add new knowledge to the ontology

[1] http://www.w3.org/Submission/SWRL.

[2] http://hermit-reasoner.com.

[3] http://owl.man.ac.uk/factplusplus.

[4] http://www.w3.org/TR/sparql11-query.

through the CONSTRUCT clause. By using this clause, it is possible to establish new relations among individuals or among individuals and classes.

Due to the fact that OWL is heavily based on formal semantics, there are some situations in which SPARQL is highly useful because it overcomes some of the limitations of OWL when, for example, open-world assumption issues arise. Finally, it also supports aggregations, which are very useful on the extraction of information from ontologies.

3 Class Expression Learning in Ontologies

In the field of ontologies, we can find two different types of statements. On the one hand, we have the set of statements that define the classes and properties in the scheme of the knowledge base. Those statements are typically used by reasoners in order to obtain new knowledge about the classes and properties already defined in the ontology. Normally, the task of automatic reasoners consists on determining the subsumption relationship between classes in the knowledge base, extending the asserted hierarchy of concepts. On the other hand, it is possible to define instances of the classes defined in the schema. In this case, the task of reasoners consists on classifying individuals as instances of the classes defined in the scheme of the ontology. Following, we provide a typical example in order to illustrate the reasoning.

Example The *GrandParent* class is a subclass of *Parent* class, assuming the following statements are defined in the ontology.

$$Grandparent \equiv \exists\, hasChild.(\exists\, hasChild.\top)$$
$$Domain(hasChild) : Father$$

The reasoners would classify all the individuals doubly related to other individuals through the *hasChild* property as instances of the *GrandParent* class.

In both cases, the reasoners cannot modify the description of a class or suggest the existence of new classes. For this reason, the usefulness of ontologies is usually limited to verify the consistency of the knowledge base or to extend the hierarchy of concepts.

On the contrary, the objective of CEL algorithms is to determine new class descriptions for concepts that may be used to classify individuals in an ontology according to some criterion. More formally, given a class C, the goal of CEL algorithms is to determine a class description A such that $A \equiv C$.

Let suppose an ontology O that has enough individuals defined in it. The set of individuals in $\Delta^{\mathcal{I}}$ is the search space S. CEL algorithms search in S, trying to find a description for class A such that $A^{\mathcal{I}}$ contains the same individuals in $C^{\mathcal{I}}$.

Definition 3.1 (*Pos(C)*) $Pos(C) \subseteq \Delta^{\mathcal{I}}$ is the set of individuals in O such that $x \in Pos(C) \implies x \in C^{\mathcal{I}}$.

Definition 3.2 (*Neg(C)*) $Neg(C) \subseteq \Delta^{\mathcal{I}}$ is the set of individuals in O such that $x \in Neg(C) \implies x \notin C^{\mathcal{I}}$.

As it can be seen, the CEL problem may be defined as a supervised learning problem but unlike the usual supervised learning problems the number of features for each instance is not fixed. They are dynamically generated as the CEL algorithm moves along the search space S. To navigate through the space S the CEL algorithms usually apply a refinement operator to existing classes in the knowledge base Ref. [15].

Definition 3.3 (*Refinement operator*) A refinement operator ρ is a mapping from S to 2^S such that for any $C \in S$ we have that $C' \in \rho(C)$ implies C' is a generalisation or a specialisation of C.

In addition to this operator, it is also necessary to establish a search strategy in S that maximizes the searched area and avoid the analysis of already visited areas. In literature, we can find several proposed search strategies Refs. [15, 25]. Most of them are based on graph exploration algorithms and some of them are based on computational intelligence Ref. [23] like a genetic algorithms Ref. [16] where the refinement operator consists on the combination of existing classes in the knowledge base.

The Algorithm 1 represents a very basic implementation of a CEL algorithm. First, the algorithm gets the current class description that best fit the POS(C) and NEG(C) sets. This class description is combined, using a selected refinement operator, with all of the other class descriptions that are present in the ontology and only the valid descriptions are added to the ontology. The process is restarted until the stopping condition is met. In this case the algorithm stops when an number of class description are evaluated.

Example Let suppose for example the existence of a family ontology O, having a sufficient number of individuals, where the concepts *Male*, *Female*, *Parent* and *Child* and the property *hasChild* are defined conveniently. Let suppose we want to automatically find a description for a new class *Father*.

1. First, the individuals in the *Pos(Father)* and *Neg(Father)* sets have to be identified. The *Pos(Father)* set contains individuals in O that should be classified as *Father*. The *Neg(Father)* set contains individuals that should be classified as $\neg Father$.
2. Using a refinement operator ρ the search space S is travelled. During this travel a set of class descriptions $D \subseteq S$ is generated, where the classes and properties in O are combined using the DL operators. Following the example of the family ontology, the following class descriptions may be eventually generated: $\neg Male$, $Male \sqcap Female$, $\exists\, hasChild.\top$, $\forall\, hasChild.(\neg(Parent \sqcup Child))$.
3. For each class description $d_i \in D$, the sets $d_i^{\mathcal{I}}$ and $(\neg d_i)^{\mathcal{I}}$ are calculated. The process stops when $d_i^{\mathcal{I}} = Pos(Father)$ and $(\neg d_i)^{\mathcal{I}} = neg(Father)$. Depending on the complexity of the concept that we want to learn and the number of individuals in O, it may be complex to travel the entire search space S and find

Algorithm 1 CEL algorithm

Require: $\mathcal{C}$ is the set of class descriptions in the ontology. $\mathcal{C}_{pos}$ and $\mathcal{C}_{neg}$ are the $Pos(C)$ and $Neg(C)$ sets, respectively. P is the set of refinement operators that is used to generate new class descriptions. n is the maximum number of class description the algorithm generates in the search process. α is a constant float value that indicates the importance of negative samples classification accuracy.

```
1: function CEL(C, C_pos, C_neg, P, n)
2:     while |C| < n do
3:         best ← BEST-DESCRIPTION(C, C_pos, C_neg)
4:         C' ← ∅
5:         for all ρ_i ∈ P do
6:             for all c_i ∈ C do
7:                 d ← ρ_i(best, c_i)
8:                 if valid(d) then
9:                     C' ← C' ∪ d
10:        C ← C ∪ C'
11:    return best
12:
13: function BEST- DESCRIPTION(C, C_pos, C_neg)
14:    v ← −∞
15:    c ← c_0
16:    for all c_i ∈ C do
17:        v_pos ← |c_i^I ∩ C_pos^I|/|C_pos^I|
18:        v_neg ← |(¬c_i)^I ∩ C_neg^I|/|C_neg^I|
19:        if v_pos − α · v_neg > v then
20:            v ← v_pos − α · v_neg
21:            c ← c_i
22:    return c
```

a solution within a reasonable time. So, CEL algorithms usually give the class descriptions that best approximate the *Father* concept as result.

If the process runs for enough time, the CEL algorithm will eventually found a description $d_i = Male \sqcap \exists hasChild.\top$ in the second stage. Assuming the individuals in O are correctly annotated, $d_i^{\mathcal{I}} = Pos(Father)$ and $(\neg d_i)^{\mathcal{I}} = Neg(Father)$, so the process will stop and d_i will be proposed as a solution. In some cases, the CEL algorithm may continue searching for other alternative solutions.

Therefore, one of the main advantages of the use of ontologies in the field of machine learning is that the information is perfectly structured, so it is possible to define refinement operators and search strategies in S regardless of the scope of the problem. Therefore, our proposal overcome the problem related to create *ad hoc* applications for this task every time. Furthermore, another advantage of using ontologies is that invalid solutions can be discarded quickly, without the need of evaluating them. Not all descriptions in D are valid. Some of them can be discarded because they produce contradictions in the knowledge base, reducing strongly the search space.

4 Class Expression Learning Applied to Sentiment Analysis

In this section we describe the procedure to find a class description that represents documents that express a positive or negative opinion based on the use of CEL algorithms in the field of sentiment analysis.

The procedure for obtaining a solution requires the steps that are illustrated in Fig. 1. First, all text documents need to be transformed in form of ontologies with the help of a POS tagger. The stop words may be taken into account by the POS tagger but for efficiency they are not included in the resulting ontologies. The text documents that are used to train the classifier are merged in a global ontology. This global ontology is used as the input of the CEL algorithm in order to find a class description that best fits *POS(Positive)*. The resulting expression is used by the reasoner to classify any other text document, which needs to be also expressed in form of ontology.

So, CEL algorithms require the knowledge base to be expressed in the form of ontology. Therefore, first, we present an ontology-based text model to transform the information contained in text documents to an ontology. Then, we present a procedure to built an ontology population based on a set of documents. Finally, we propose the use of a CEL algorithm to find a class description that describes the positive concepts in the set of documents.

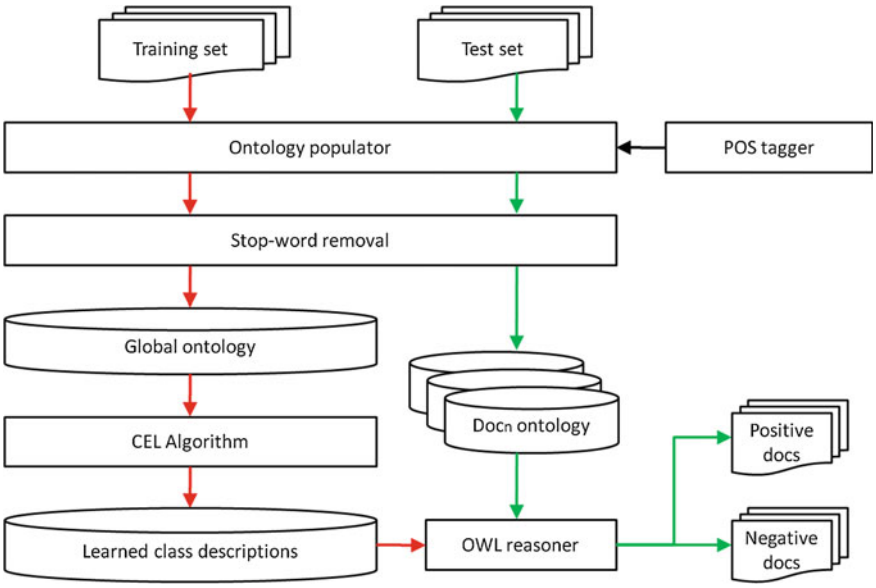

Fig. 1 Functional architecture

4.1 Ontology-Based Text Model

In this section, the ontology-based text model for ontology representation of text documents is presented. The kernel of the proposed ontology-based text model offers a reusable basis to the analysis of textual data. Here, the relevant DL axioms of the kernel are presented that include list patterns as well as the most important basic concepts.

4.1.1 List Pattern

In order to identify the most popular entities in a text and the relations among them, our proposal is based on a list structure. The basic concepts $List \sqsubseteq \top$ and $Item \sqsubseteq \top$ with the following relations among them are defined in the proposed model:

$$hasNext \sqsubseteq isFollowedBy \sqsubseteq itemProperty$$

hasNext is defined as a functional, asymmetric and irreflexive property. Because it has been defined as a functional property just one item can follow to an item. The inverse property is also defined as functional, forcing an item to be directly preceded by an unique item. The transitive property *isFollowedBy* is defined as a superproperty of *hasNext*. The property *hasNext* is referred to the item that is just after another item, while the property *isFollowedBy* is refereed to the set of items following an item in the list. Furthermore, the property *hasItem* establishes the membership of an item in its list.

$$\begin{aligned} hasItem &\sqsubseteq listProperty \\ hasNext &\sqsubseteq inTheContextOf \\ hasNext^{-} &\sqsubseteq inTheContextOf \end{aligned}$$

The symmetric property *inTheContextOf* is defined as a superproperty of both the *hasNext* property and $hasNext^{-}$, which is the inverse of property *hasNext*. This property relates an item in the list with any of the elements that are immediately before or after of this item.

There is a set of concepts and relations that do not need to be explicitly defined in the model because they can be expressed using DL operators. However, in order to simplify the design of new concepts and relations, these are explicitly defined in the model. So, *hasPrevious*, *isPrecededBy* and *isPartOf* are defined as inverse properties of *hasNext*, *isFollowedBy* and *hasItem*, respectively.

The concepts *First* and *Last* identifies the starting and ending items of the list. Due to open-world assumption in OWL, reasoners cannot automatically infer the individuals that belong to these concepts. Therefore, it is necessary to annotate these individuals when the text is processed.

$$\top \sqsubseteq \forall\, hasID.Datatype\#long$$

Finally, for practical reasons, a functional property *hasID* is used to identify all of the individuals in the model with an unique code. In this way, it is easier the addition of new items to the ontology without the need of asserting that all of them are different from the existing individuals.

4.1.2 Basic Concepts

Once the list structure has been defined, it is possible to describe the most popular entities that can be found in a text such as sentences, documents and terms. Following, we present the relevant axioms of these entities in the ontology-based text model.

In some applications, it is also desirable to take *Punctuation marks* into account. For this reason, the concept *Token* is defined that encompasses both the *Term* and the *Punctuation marks* concepts. A *Sentence* is defined as a list containing tokens and a *Document* is defined in turn as a list containing sentences. The concepts *Document*, *Sentence*, *Term* and *PunctuationMark* are established as disjoint concepts.

$$\begin{aligned} Token &\equiv PunctuationMark \cup Term \\ \top &\sqsubseteq \forall\, hasLexeme.Datatype\#string \\ Term &\sqsubseteq \leq 1\, hasLexeme \\ Sentence &\equiv \exists\, hasElement.Token \\ Document &\equiv \exists\, hasElement.Sentence \end{aligned}$$

The terms in a sentence are classified according to their Parts of Speech (POS). Each of them must be represented as an unique individual in the ontology regardless of their lexeme. Therefore, the property *hasLexeme* is used to link a term with its lexeme.

Depending on the objectives of the text analyzer, the sentences may be excessively large information units Ref. [28]. For this reason, sentences are sometimes split into shorter segments that are called *Contexts*. *Contexts* could be a powerful tool for text analyzers to identify patterns in the text. To do so, arbitrarily long sequences of tokens connected by the property *hasNext* should be established. To find a particular pattern in the text, it is only necessary to apply some restrictions to the items in the context. For example, to find contexts of four elements length, being the initial item a punctuation mark and the ending term a noun, the following class description may be defined:

$$PunctuationMark \sqcap \exists\, hasNext.(\exists\, hasNext.(\exists\, hasNext.Noun))$$

By default the relation *inTheContextOf* assumes contexts of three tokens length. If longer contexts are needed, it is necessary to increase the length of the contexts by means of rules such as the one shown below (in SWRL), for example, which sets

contexts of five tokens length.

$$hasNext(token_i, token_j) \wedge hasNext(token_j, token_k) \rightarrow inTheContextOf(token_i, token_k)$$

Although it might seem useful to define concepts such as *Positive*, *Neutral* or *Negative* to identify the different types of documents, it is not the aim. Due to the fact that the proposal is based on learning equivalent description for the concepts *Positive*, *Neutral* or *Negative*. Furthermore, if the concept *Positive* were included in the scheme of the ontology and the corresponding individuals annotated, the CEL problem would be trivially solved by giving the class expression *Positive* as result. Due to the fact that additions of these concepts to the core of the model for describing text documents would make it not very reusable. So, the model could be used in the field of sentiment analysis and, moreover, other purposes. Therefore, one of the advantages of using ontologies is kept due to the fact that they can be easily combined to form a global ontology.

4.2 Ontology Population

In this section, the construction of the ontology population is described for a set of text documents.

First, each sentence in a document is divided in terms. For each term in the sentence, a new individual is created in the ontology. The lexemes of the terms are related to the term individual by mean of the data type property *hasLexeme*. It is not necessary to assert that all of the terms are individuals of the concept *Term*. Any reasoner will identify these individuals as terms because the concept *Term* has been defined to be the domain of the property *hasLexeme*.

As was indicated in Sect. 4.1.2, each token (*Term* or *PunctuationMark*) has to be associated to an unique identifier through the property *hasID*, which has been defined as a functional relation. For this purpose, a sequence generator function has to be created with the aim that reasoners identify all tokens in the documents as different individuals.

The order among elements in a sentence is established by relating two consecutive tokens by mean of the property *hasPrevious*. The reasoners can always get the next token using the property *hasNext*, which has been defined as the inverse property of *hasPrevious*.

In order to relate all tokens to their sentences, the property *isPartOf* is used. In our proposal, it is not necessary to relate all tokens to the sentence they belong to because the tokens of a sentence are part of the same sentence the previous tokens are part of. Therefore, just the first token of the sentence needs to be associated to the sentence of which it is a member. It is important to note that in our proposal this kind of reasoning is based on the SWRL language. By relating the tokens to their

sentences, it is very easy to analyze the sentences according to their elements with the property *hasItem* that was defined as the inverse of the property *isPartOf*. So, for example, an interrogative sentence may be defined as a sentence having a question mark.

$$QuestionMark \sqsubseteq PunctuationMark$$
$$InterrogativeSentence \equiv \exists hasItem.QuestionMark$$

4.3 Class Expression Learning of Sentiment Concepts

Once the documents are expressed in the ontology-based model proposed in Sect. 4.1 by building the ontology population presented in Sect. 4.2, the CEL algorithm is proposed to find a class description that describes the positive documents or negative documents. To do so, in this chapter, we propose the use of the DL-Learner tool that was presented in Ref. [14].

The process begins with the identification of *Pos(Positive)* and *Neg(Positive)* individuals, where *Positive* is a new empty concept representing the positive documents. For the creation of these groups of individuals, we have developed a tool, which is also responsible for transforming text documents in form of ontologies. In addition, this tool is able to generate a configuration file for the DL-Learner tool where individuals belonging to the *Pos(Positive)* and *Neg(Positive)* sets are pointed out. The tool assigns individuals to either *Pos(Positive)* or *Neg(Positive)* sets depending on the location of the text file in the directory structure, so it can be used in other text classification problems without having to be modified.

In this case, we are trying to find the sentiment polarity of the whole document but in some cases a finer detail is needed and the polarity of each paragraph of sentence may be calculated. The CEL algorithm can also be applied to those cases. The *Pos(Positive)* and *Neg(Positive)* sets just need to be populated accordingly. In the later case, for instance, the application that processes the text documents just need to populates both sets with individuals of type *Sentence*. In this case, the class description obtained as result describes the sentences expressing a positive opinion.

5 Results

In order to show the usefulness and effectiveness of our proposal, results obtained by the CEL-based classifier applied to sentiment analysis are shown. Furthermore, the results obtained by the CEL-based classifier with respect to a C4.5 decision tree classifier that uses the standard BOW approach are compared. To do so, we have made use of the freely available documents in Ref. [21] that are a list of two thousand annotated documents containing opinions about films.

5.1 Class Expression Learning of Positive Documents

We have used some of the documents in Ref. [21] in order to learn the concept of document giving positive feedback about a film. The first twenty five positive documents have been selected as *Pos(Positive)* whereas the first twenty five negative documents have been selected as *Neg(Positive)*. As was shown in Fig. 1, all documents have been processed and a single global ontology has been generated as result, following the model presented in Sect. 4.1. This ontology, along with the list of individuals in *Pos(Positive)* and *Neg(Positive)* are used as the input for the DL-Learner tool, obtaining the following two class descriptions as result (in Manchester syntax), among others.

$$hasItem\ some\ (hasItem\ min\ 2\ (PastTense\ and\ (isPrecededBy\ some\ Thing))) \tag{1}$$

$$\begin{aligned} hasItem\ some\ (hasItem\ some(PossessiveEnding\ and\ (inTheContextOf\ some \\ (Noun\ and\ (isFollowedBy\ some\ Thing)))\ and\ (isPrecededBy\ some\ (Adverb\ and \\ (hasPrevious\ some\ ((Noun\ or\ (hasPrevious\ some\ Verb)) \\ and\ (isFollowedBy\ some\ (Adjective\ and \\ (isFollowedBy\ some\ Singular))))))))) \end{aligned} \tag{2}$$

The class description (1), with 60.42 % of accuracy when classifying positive documents, represents those documents that contain a sentence with at least two verbs in the past tense, not being the first terms of the sentence. Furthermore, the class description (2), illustrated in Fig. 2, with 81.25 % of efficacy, is the best solution found by the CEL algorithm. It represents documents that contain any sentence having a possessive ending that is in the context of a noun that is not the final term of that sentence. In addition, the possessive ending must be preceded in the sentence by some adverb, which must be immediately preceded in turn by a noun or some other term immediately preceded by a verb. That name or term preceded by a verb should be

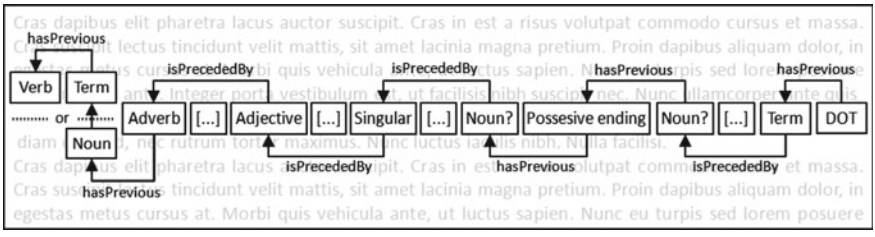

Fig. 2 Best class description found by CEL algorithm

followed by an adjective or a term followed, in turn, by a singular term. The following sentence is an example of a sentence represented by the this class description.

> "Even so, the strengths of election rely upon its fantastic performances from Broderick, Witherspoon, and newcomer Jessica Campbell, as Paul's anti-social sister, Tammy".

Let C be the class description that best describes text documents with a positive opinion. It is possible to use the class C to classify a single document d without using the DL-Learner tool. For this, the text document d has to be transformed in the form of an ontology, following the model proposed in Sect. 4.1, and check if the document complies with the class description C, that is, if $d \in C^{\mathcal{I}}$. To do this, we just need to check if $C^{\mathcal{I}} = \varnothing$, that is, if there is some individual in d that fits the class description C.

5.2 Comparative Analysis

A comparative analysis of the results obtained by the CEL-based classifier and a C4.5 tree-based classifier that use the usual BOW representation of the text documents are presented here. The DL-Learner tool is used as the CEL-based classifier. Default options for Weka's implementation of a C4.5 tree-based classifier (J48) have been used in all tests, which have been made in one of the nodes of the computing cluster at the University of Cádiz (2xIntel Xeon E5 2670, 2.6 GHz, 256 GB RAM). The memory of the Java Virtual Machine is limited to a maximum of 16 GB per process.

The efficacy percentages given in Table 2 are those indicated by the DL-Learner tool and they refer to the efficacy of the classifier for the training set, that is, for the case of the first fifty documents in Ref. [21], which were used to train both classifiers. DL-Learner makes use of an own approximate incomplete reasoning procedure for Fast Instance Checks (FIC) which partially follows a closed world assumption. When using a reasoner that complies with the OWL standard, such as Hermit, the accuracy rate downs to 78 %. A classifier based on a C4.5 decision tree has a 98 % accuracy for the same data set.

The next hundred documents in Ref. [21] are used as the test set. Fifty of them are documents with a positive opinion, while the remaining fifty documents have a negative opinion. Using the test data set the classifier based on a C4.5 decision tree offers an accuracy of 51 %, whereas the version of the classifier using the class description (2), in Sect. 4.3, is able to correctly classify 63 % of the documents. In this case, we have no information about the efficacy on the test data set for the classifier

Table 2 Accuracy of classifiers

	FIC (%)	Hermit (%)	C4.5 (%)
Training set	81.25	78.00	98.00
Test set	–	63.00	51.00

that uses the FIC of the DL-Learner tool because the reasoner runs out of memory. The learned class description has been tested on each document individually, using the Hermit reasoner, as described at the end of the previous section.

6 Discussion

In this section we describe the problems that we have found while building the former and the solutions we have adopted are discussed.

The DL-learner tool implements several CEL algorithms Ref. [16]. Some of them, like the Class Expression Learning for Ontology Engineering (CELOE), are focused on the process of building ontologies and are biased towards simple class descriptions. This type of algorithms sacrifices accuracy for the sake of simplicity of class descriptions so that the developer can easily understand them and incorporate to the ontology scheme, if appropriate. This is not the type of algorithms that should be used in our proposal, unless the learned class description is intended to be incorporated to the ontology. In this paper we have chosen to use the Ontology Class Expression Learning (OCEL) algorithm. Although it generates more complex class descriptions they offer the key advantage that are more effective for classifying text documents.

The complexity of the class description obtained by the algorithm is not only a problem from the point of view of the ontology developer. As the number of concepts and DL operators increase, the time required for the reasoners to classify the instances of ontology increases. In fact, since this task must be repeated for each new class description proposed in the learning process, the high processing time required for this task may be considered the main drawback. For example, the class description (2) shown in Sect. 4.3, was found after 155 minutes of executions of the DL-Learner tool, of which 145 were employed in the classification of instances of the ontology.

Despite the large amount of time required for the DL-Learner tool, this could have been much higher if the FIC were not used. The FIC implementation provides better reasoning performance when classifying individuals but uses an approximate reasoning. This means that the FIC classifies instances into classes that are excluded by standard OWL reasoners. Therefore, some of the class descriptions given as result of the CEL algorithm do not offer the same results when evaluated in OWL standard reasoners. For example, the class description shown in description (3) always represents an empty set of individuals when open world assumptions and the methodology proposed in Sect. 4.3 are followed, for example. For this reason we have decided to disable the inclusion of the complement operator in the class descriptions generated by the DL-Learner tool.

$$not\ (hasItem\ some\ PastTense) \tag{3}$$

A similar problem occurs with the semantics of the *allValuesFrom* constraint for a property r. The standard option is to also include all those individuals that have no value for property r at all, whereas the FIC in DL-Learner includes only explicitly

related individuals through the property r. To mitigate this problem an option has been activated in the DL-Learner tool that only includes the *allValuesFrom* constraints in the class descriptions for a property r when there is already a cardinality and/or existential restriction on r.

As shown above, the developers of the DL-Learner tool, aware of the problem of the high processing time required by the reasoners, have opted for the development of an approximate, incomplete reasoning procedure for fast instance classification. However, there are other problems arising from the use of reasoners for instance classification. The property *isFollowedBy* is a composite property because it is defined to be transitive.[5] To maintain the decidability of OWL, it is not allowed to be used in combination with *ObjectMinCardinality* constraint. Therefore, some of the class descriptions proposed by the CEL algorithm cannot be tested. Although it is possible to disable the use of cardinality restrictions in DL-Learner, it has not been deactivated because not all the properties in the model have been defined as transitive roles. The class description (2), which was illustrated in Sect. 4.3, is an example of a valid class description that includes cardinality restrictions.

To improve reasoning performance, we also decided to ignore all those terms providing little information about the content of the text. We have set a list of stop words that are ignored by the tool that transform text documents in ontologies, following the scheme of the model proposed in Sect. 4.1. Actually, as shown in Fig. 1, the stop words may be taken into account by the POS tagger but they are not included in the resulting ontologies. The list structure of sentences is maintained through the *hasNext* property, but only among non-stop words.

In view of the results obtained, it is shown how the accuracy of the CEL-based classifier for the sentiment classification is somewhat higher than the classifier based on the usual BOW structure for the test set. However, we have to keep in mind that the C4.5 decision tree classifier has not been optimized and its accuracy could probably be increased by tuning its parameters. So, its accuracy may also be improved by increasing the number of individuals in the training set. However, this would cause the reasoner of DL-Learner runs out of memory, so we could not compare the efficacy of both types of classifiers for the same training set data.

7 Conclusions and Future Works

In this chapter, we have presented the development of a text document classifier based on class descriptions in DL. The objective of the classifier is to identify text documents expressing a positive and negative opinions. To do so, an ontology-based text documents model has been proposed to represent text documents as ontologies, containing all the necessary concepts and properties for describing text documents.

[5] http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/#Property_Hierarchy_and_Simple_Object_Property_Expressions.

A procedure to transform the text documents to this ontology-based text model has also been proposed. Following this procedure, some of the documents have been expressed in form of ontology and the CEL technique has been used in order to find the class expressions that represent different types of documents.

The main proposal of this chapter have been to show that CEL-based classifiers are an excellent option for sentiment analysis and, moreover, they can complement other types of classifiers. Class descriptions obtained by the CEL algorithm can be used by the developers to incorporate relevant new features into the knowledge base. Those features add valuable information for the classifiers that is usually difficult to be identified by people.

The DL-Learner tool, which implements several CEL algorithms, has been used in order to find a class description that represents documents expressing positive opinions. Some of the annotated documents have been converted to the ontology-based text model and used as the training set of the CEL-based classifier. The class description given as result has a significant lower classification accuracy for the training set than the accuracy achieved by the classifier based on a C4.5 decision tree, which uses the usual BOW approach. However, the class description based classifier obtains a slightly higher accuracy when the test set is used.

Furthermore, we have pointed out the main advantages and disadvantages of using CEL-based classifiers for text classification. Those disadvantages are related with the large number of resources required by reasoners, mainly. Some of the solutions we have adopted to mitigate this problem have been proposed, but it remains an open research area to obtain improved results.

References

1. Chandrasekaran, B., Josephson, J.R., Benjamins, V.R.: What are ontologies, and why do we need them? IEEE Intell. Syst. Appl. **14**(1), 20–26 (1999)
2. Chen, R.C., Huang, Y.H., Bau, C.T., Chen, S.M.: A recommendation system based on domain ontology and swrl for anti-diabetic drugs selection. Expert Syst. Appl. **39**(4), 3995–4006 (2012)
3. Colace, F., De Santo, M., Napoletano, P., Becchi, C., Chang, S.K.: Ontological filtering for sentiment analysis. In: DMS'2012, pp. 60–66 (2012)
4. Gomez-Perez, A., Corcho, O.: Ontology languages for the semantic web. IEEE Intell. Syst. Appl. **17**(1), 54–60 (2002)
5. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Wang, H.H.: The manchester owl syntax. In: Proceedings of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006) (2006)
6. Horrocks, I.: Ontologies and the semantic web. Commun. ACM **51**(12), 58–67 (2008)
7. Horrocks, I., Patel-Schneider, P.F., Van Harmelen, F.: From SHIQ and RDF to OWL: the making of a web ontology language. Web Semant. **1**(1), 7–26 (2003)
8. Kaur, A., Gupta, V.: A survey on sentiment analysis and opinion mining techniques. J. Emerg. Technol. Web Intell. **5**(4), 367–371 (2013)
9. Kearney, C., Liu, S.: Textual sentiment in finance: a survey of methods and models. Int. Rev. Fin. Anal. **33**, 171–185 (2014)
10. Knijff, J., Frasincar, F., Hogenboom, F.: Domain taxonomy learning from text: the subsumption method versus hierarchical clustering. Data Knowl. Eng. **83**, 54–69 (2013)

11. Kohler, J., Philippi, S., Specht, M., Ruegg, A.: Ontology based text indexing and querying for the semantic web. Knowl.-Based Syst. **19**(8), 744–754 (2006)
12. Kontopoulos, E., Berberidis, C., Dergiades, T., Bassiliades, N.: Ontology-based sentiment analysis of twitter posts. Expert Syst. Appl. **40**(10), 4065–4074 (2013)
13. Lau, R.Y.K., Li, C., Liao, S.S.Y.: Social analytics: learning fuzzy product ontologies for aspect-oriented sentiment analysis. Decis. Support Syst. **65**(C), 80–94 (2015)
14. Lehmann, J.: DL-learner: learning concepts in description logics. J. Mach. Learn. Res. **10**, 2639–2642 (2009)
15. Lehmann, J., Auer, S., Bhmann, L., Tramp, S.: Class expression learning for ontology engineering. J. Web Seman. **9**(1), 71–81 (2011)
16. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Mach. Learn. **78**, 203–250 (2010)
17. Li, S., Liu, L., Xiong, Z.: Ontology-based sentiment analysis of network public opinions. Int. J. Digit. Content Technol. Appl. **6**(23), 371–380 (2012)
18. Liu, B.: Sentiment Analysis and Opinion Mining. In: Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers (2012)
19. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intell. Syst. Appl. **16**(2), 72–79 (2001)
20. Martin-Valdivia, M.T., Martinez-Camara, E., Perea-Ortega, J.M., Urena-Lopez, L.A.: Sentiment polarity detection in spanish reviews combining supervised and unsupervised approaches. Expert Syst. Appl. **40**(10), 3934–3942 (2013)
21. Pang, B., Lee, L.: A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the ACL (2004)
22. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2008)
23. Pedrycz, W.: Computational Intelligence: An Introduction. CRC Press Inc, Boca Raton (1997)
24. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical owl-dl reasoner. Web Seman. **5**(2), 51–53 (2007)
25. Tran, A., Dietrich, J., Guesgen, H., Marsland, S.: An approach to parallel class expression learning. In: Bikakis, A., Giurca, A. (eds.) Rules on the Web: Research and Applications. Lecture Notes in Computer Science, vol. 7438, pp. 302–316. Springer, Berlin (2012)
26. Uguz, H.: A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. Knowl.-Based Syst. **24**(7), 1024–1032 (2011)
27. Uschold, M., Gruninger, M.: Ontologies: principles, methods and applications. Knowl. Eng. Rev. **11**(2), 93–136 (1996)
28. Wei, T., Lu, Y., Chang, H., Zhou, Q., Bao, X.: A semantic approach for text clustering using wordnet and lexical chains. Expert Syst. Appl. **42**(4), 2264–2275 (2015)
29. Yin, P., Wang, H., Guo, K.: Feature-opinion pair identification of product reviews in chinese: a domain ontology modeling method. New Rev. Hypermedia Multimedia **19**(1), 3–24 (2013)
30. Zhang, F., Ma, Z.M., Li, W.: Storing owl ontologies in object-oriented databases. Knowl.-Based Syst. **76**, 240–255 (2015)