Polarity Consistency Checking for Domain Independent Sentiment Dictionaries

Eduard C. Dragut, *Member, IEEE*, Hong Wang, Prasad Sistla, Clement Yu, and Weiyi Meng, *Member, IEEE*

Abstract—Polarity classification of words is important for applications such as Opinion Mining and Sentiment Analysis. A number of sentiment word/sense dictionaries have been manually or (semi)automatically constructed. We notice that these sentiment dictionaries have numerous inaccuracies. Besides obvious instances, where the same word appears with different polarities in different dictionaries, the dictionaries exhibit complex cases of polarity inconsistency, which cannot be detected by mere manual inspection. We introduce the concept of polarity consistency of words/senses in sentiment dictionaries in this paper. We show that the consistency problem is NP-complete. We reduce the polarity consistency problem to the satisfiability problem and utilize two fast SAT solvers to detect inconsistencies in a sentiment dictionary. We perform experiments on five sentiment dictionaries and WordNet to show inter- and intra-dictionaries inconsistencies.

Index Terms—Sentiment analysis, sentiment dictionary, polarity inconsistency

1 Introduction

The opinions expressed in various web and media outlets (e.g., blogs, newspapers) are an important yardstick for the success of a product or a government policy. For instance, a product with consistently good reviews is likely to sell well. The general approach of determining the overall orientation (i.e., positive or negative) of a sentence/document is by analysis of the orientations of the individual words [1], [2], [3], [4]. Sentiment dictionaries are utilized to facilitate the summarization. There are numerous works that, given a sentiment lexicon, analyze the structure of a sentence/document to infer its orientation, the holder of an opinion, the sentiment of the opinion, etc. [3], [5], [6], [7]. Several domain independent sentiment dictionaries have been manually or (semi)-automatically created, e.g., general inquirer (GI) [8], opinion finder (OF) [9], appraisal lexicon (AL) [10], SentiWordNet (SWN) [11] and Q-WordNet (QW) [12]. QW and SWN are lexical resources which classify the synsets(senses) in WordNet according to their polarities. We call them *sentiment sense dictionaries* (SSD). OF, GI and AL are called *sentiment word dictionaries* (SWD). They consist of words manually annotated with their corresponding polarities. We have noticed the following problems with the sentiment dictionaries:

- They exhibit substantial (intra-dictionary) inaccuracies. For example, the synset {*Indo-European, Indo-Aryan, Aryan*} *(of or relating to the former Indo-European people)*, has a negative polarity in QW, while intuitively this synset has a neutral polarity instead.
- They have (inter-dictionary) inconsistencies. For example, the adjective `cheap` is positive in AL and negative in OF.
- These dictionaries do not address the concept of polarity (in)consistency of words/synsets.

We concentrate on the concept of (in)consistency in this paper. We define consistency among the polarities of words/synsets within and across sentiment dictionaries and give methods to check them. We provide two examples to illustrate the problem addressed in this paper.

The first example is the verbs `confute` and `disprove`, which have positive and negative polarities, respectively, in OF. According to WordNet, both words have a unique sense, which they share:

disprove, confute (prove to be false) "The physicist disproved his colleagues' theories"

Assuming that WordNet has complete information about the two words, it is rather strange that the words have distinct polarities. By manually checking two authoritative English dictionaries, Oxford[1] and Cambridge,[2] we confirm that the information about `confute` and `disprove` in WordNet is the same as that in these dictionaries. So, the problem seems to originate in OF.

The second example is the verbs `tantalize` and `taunt`, which have positive and negative polarities, respectively, in OF. They also have a unique sense in WordNet, which they share. Again, there is a contradiction. In this case Oxford dictionary mentions a sense of `tantalize` that is missing from WordNet: *"excite the senses or desires of (someone)"*. This

1. http://oxforddictionaries.com/.
2. http://dictionary.cambridge.org/.

- E.C. Dragut is with the Computer and Information Sciences, Temple University, Philadelphia, PA 19022. E-mail: edragut@temple.edu.
- H. Wang, P. Sistla, and C. Yu are with the Computer Science Department, University of Illinois at Chicago, Chicago, IL 60607-7053. E-mail: {hwang207, sistla, cyu}@uic.edu.
- W. Meng is with the Computer Science Department, Binghamton University, Binghamton, NY 13902. E-mail: meng@cs.binghamton.edu.

Manuscript received 25 Feb. 2013; revised 13 May 2014; accepted 6 June 2014. Date of publication 16 July 2014; date of current version 28 Jan. 2015. Recommended for acceptance by A.K.Tung. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2014.2339855

sense conveys a positive polarity. Hence, `tantalize` conveys a positive sentiment when used with this sense.

In summary, these dictionaries have conflicting information. Their use in sentiment analysis tasks can give conflicting results (see Section 9.5). Manual checking of sentiment dictionaries for inconsistency is a difficult endeavor. We deem words such as `confute` and `disprove` inconsistent. We aim to unearth these inconsistencies in sentiment dictionaries. Note that the presence of inconsistencies found via polarity analysis is not exclusively attributed to one party, i.e., either the sentiment dictionary or WordNet. Instead, as emphasized by the above examples, some of them lie in the sentiment dictionaries, while others lie in WordNet. Therefore, a by-product of our polarity consistency analysis is that it can also locate likely places where WordNet needs linguists' attention.

We will show that the problem of checking whether the polarities of a set of words are consistent is NP-complete. Although the problem is NP-complete [13], we present a reduction of this problem to the SAT problem (which is also an NP-complete problem) and employ existing *SAT solvers* [14] on the resulting SAT problem to obtain solutions to the polarity consistency problem. We utilize two fast SAT solvers to detect inconsistencies. Our experimental study show that substantial inconsistencies are discovered among words with polarities within and across sentiment dictionaries. This suggests that some remedial work needs to be performed on these sentiment dictionaries as well as on WordNet.

The contributions of this paper are:

- address the consistency of polarities of words/senses. We originally introduced the problem in [15].
- show that the consistency problem is NP-complete;
- reduce the polarity consistency problem to SAT;
- give a new polynomial-length reduction and further improvement of it, which are far more efficient than the exponential-length reduction given in [15];
- give a method for detecting inconsistencies between an SWD and an SSD;
- utilize fast SAT solvers to detect inconsistencies in sentiment dictionaries;
- give experimental results to demonstrate that our technique identifies considerable inconsistencies in various sentiment lexicons as well as discrepancies between these lexicons and WordNet.

Our initial solution to the problem of *polarity consistency checking* (PCC) was published in [15]. For completeness, this solution is described in this paper in Section 5, in particular in Section 5.2. This solution has an important shortcoming: it generates boolean formulas that have exponential lengths when converting PCC into SAT. We experimentally show that this solution cannot handle words such as `give` and `make` which have large numbers of synsets—we left the implementation of this solution running on a quad-core computer with 12 GB of memory for a week without ever terminating. In this paper, we present a new solution that is proven to generate boolean formulas of polynomial lengths. The new solution can handle all the words in WordNet and it takes only **24 minutes** to complete its computations. This solution has one small disadvantage though: it introduces a large number (thousands) of variables in the Boolean formula obtained when converting PCC into SAT. Consequently, we propose a hybrid solution that attempts to capitalize on the strengths of the two solutions. The hybrid solution has fewer variables and can solve the problem faster than either of the previous solutions: it takes only about **10 minutes** to complete. This solution is presented in Section 7. While our contribution in [15] is that we formalized and gave a first "incomplete" solution to PCC, the main contribution of this paper is that it provides linguists with real practical solutions for polarity oriented lexicon exploration.

2 Problem Definition

As argued above, the polarities of the words in a sentiment dictionary may not necessarily be consistent (or correct). In this paper, we focus on the detection of polarity assignment inconsistency for the words and synsets within and across sentiment dictionaries (e.g., OF versus. GI). We attempt to pinpoint the words with polarity inconsistencies and classify them (Section 3). We will use $-$, $+$, 0 to denote negative, positive and neutral polarities, respectively, throughout the paper.

2.1 WordNet

We give a formal characterization of WordNet. This consists of words, synsets and frequency counts. A **word-synset network** $\mathcal{N}$ is quadruple $(\mathcal{W}, \mathcal{S}, \mathcal{E}, f)$ where $\mathcal{W}$ is a finite set of words, $\mathcal{S}$ is a finite set of synsets, $\mathcal{E}$ is a set of undirected edges between elements in $\mathcal{W}$ and $\mathcal{S}$, i.e., $\mathcal{E} \subseteq \mathcal{W} \times \mathcal{S}$ and f is a function assigning a positive integer to each element in $\mathcal{E}$. For an edge (w, s), $f(w, s)$ is called the **frequency of use** of w in the sense given by s. For any word w and synset s, we say that s is a synset of w if $(w, s) \in \mathcal{E}$. Also, for any word w, we let $freq(w)$ denote the sum of all $f(w, s)$ such that $(w, s) \in \mathcal{E}$. If a synset has a 0 frequency of use then we add a small constant ϵ to the frequencies of use of all the synsets of the word. In the current implementation $\epsilon = 0.1$. This is a standard smoothing technique [16]. For instance, the word `cheap` has four senses. The frequencies of use of the word in the four senses are $f_1 = 9$, $f_2 = 1$, $f_3 = 1$ and $f_4 = 0$, respectively. With smoothing, they become $f_1 = 9.1$, $f_2 = 1.1$, $f_3 = 1.1$ and $f_4 = 0.1$. Hence, $freq(\texttt{cheap}) = f_1 + f_2 + f_3 + f_4 = 11.4$. The **relative frequency** of the synset in the first sense of `cheap`, which denotes the probability that the word is used in the first sense, is $\frac{f_1}{freq(cheap)} = \frac{9.1}{11.4} = 0.8$.

2.2 Consistent Polarity Assignment

We assume that each synset has a *unique* polarity. We define the polarity of a word to be a discrete probability distribution: $P_+, P_-, P_0 \geq 0$ with $P_+ + P_- + P_0 = 1$, where they represent the "likelihoods" that the word is positive, negative or neutral, respectively. We call this distribution a **polarity distribution**. For instance, the word `cheap` has the polarity distribution $P_+ = 0.8$, $P_- = 0.2$ and $P_0 = 0$. The polarity distribution of a word is estimated using the polarities of its underlying synsets. For instance `cheap` has four senses, with the first sense being positive and the last three senses being negative. The probability that the word expresses a negative sentiment is $P_- = \frac{f_2+f_3+f_4}{freq(cheap)} = 0.2$, while the probability that the word expresses a positive sentiment is $P_+ = \frac{f_1}{freq(cheap)} = 0.8$. $P_0 = 1 - P_+ - P_- = 0$.

Our view of characterizing the polarity of a word using a polarity distribution is shared with other works [17], [18]. Nonetheless, we depart from these works in the following key aspect. We say that a word has a (mostly) positive (negative) polarity if the **majority sense of the word** is positive (negative). That is, a word has a mostly positive polarity if $P_+ > P_- + P_0$ and it has a mostly negative polarity if $P_- > P_+ + P_0$. Or, equivalently, if $P+ > \frac{1}{2}$ or $P_- > \frac{1}{2}$, respectively. For example, on majority, `cheap` conveys positive polarity since $P_+ = 0.8 > \frac{1}{2}$, i.e., the majority sense of the word `cheap` has a positive connotation.

Based on this study, we contend that GI, OF and AL tacitly assume this property. For example, the verb `steal` is assigned only negative polarity in GI. `steal` has two other less frequently occurring senses, which have positive polarities. The polarity of `steal` according to these two senses is not mentioned in GI. This is the case for the overwhelming majority of the entries in the three dictionaries: only 112 out of a total of 14,105 entries in the three dictionaries are words with multiple polarities. For example, the verb `arrest` is mentioned with both negative and positive polarities in GI. We regard an entry $\langle w, pos, p\rangle$ in an SWD as saying that the majority sense of the word w with the part of speech pos has polarity p, although the word may carry other polarities. For instance, the adjective `cheap` has positive polarity in GI. The only assumption we make about the word is that it has a polarity distribution such that $P_+ > P_- + P_0$. This interpretation is consistent with the senses of the word. In this work we show that this property allows the polarities of words in input sentiment dictionaries to be checked. We formally state this property. Let w be a word and S_w its set of synsets. Each synset in S_w has an associated polarity and a frequency of use with respect to w. Let $S' \subseteq S_w$ and $p \in \{+, -\}$.

Definition 1. *S' is called a **polarity dominant subset of synsets** if each synset $s \in S'$ has polarity p and $\sum_{s\in S'} \frac{f(w,s)}{freq(w)} > 0.5$.*

Definition 2. *w has polarity p if there is a polarity dominant subset $S' \subseteq S_w$ such that each synset $s \in S'$ has polarity p. If there is no such subset then w has a neutral polarity.*

Definition 3. *S' is a **minimally dominant subset of synsets** (MDSs) if $\sum_{s\in S'} \frac{f(w,s)}{freq(w)} > 0.5$ and $\sum_{s\in S''} \frac{f(w,s)}{freq(w)} \leq 0.5$ for $S'' = S' - \{s\}, \forall s \in S'$.*

Definition 2 does not preclude a word from having a polarity with a *majority* sense and a different polarity with a *minority* sense. For example, the definition does not prevent a word from having both positive and negative senses, but it prevents a word from concomitantly having a majority sense of being positive and a majority sense of being negative.

Despite using a "hard-coded" constant in the definition, our approach is generic and does not depend on the constant .5. The constant is just a lower bound for deciding if a word has a majority sense with a certain polarity. It also is intuitively appealing. The constant can be replaced with an arbitrary threshold $\tau \in [.5, 1]$.

We need a formal description of polarity assignments to the words and synsets in WordNet. We assign polarities from the set $\mathcal{P} = \{+, -, 0\}$ to elements in $\mathcal{W} \cup \mathcal{S}$.

Definition 4. *Formally, a polarity assignment γ for a network $\mathcal{N}$ is a function from $\mathcal{W} \cup \mathcal{S}$ to the set $\mathcal{P}$. γ is **consistent** if it satisfies the following condition for each $w \in \mathcal{W}$:*

For $p \in \{+, -\}$, $\gamma(w) = p$ iff the sum of all $f(w, s)$ such that $(w, s) \in \mathcal{E}$ and $\gamma(s) = p$, is greater than $\frac{freq(w)}{2}$. Furthermore, for any $w \in \mathcal{W}$, $\gamma(w) = 0$ iff the above inequality is not satisfied for either value of p in $\{+, -\}$.

We contend that our approach is applicable to domain dependent sentiment dictionaries, too. We can employ WordNet Domains [19]. WordNet Domains augments WordNet with domain labels such as `art`, `sport`, `religion` and `history`. Hence, we can project the words and synsets in WordNet according to a domain label and then apply our methodology to the projection.

3 Inconsistency Classification

In this section, we attempt to give a thorough classification with examples of the possible types of polarity inconsistencies occurring within and across sentiment dictionaries. Polarity inconsistencies are of two types: input and complex. We present them in turn.

3.1 Input Dictionaries Polarity Inconsistency

Input polarity inconsistencies are of two types: intra-dictionary and inter-dictionary inconsistencies. The latter are obtained by comparing (1) two SWDs, (2) an SWD with an SSD and (3) two SSDs.

3.1.1 Intra-Dictionary Inconsistency

An SWD may have triplets of the form (w, pos, p) and (w, pos, p'), where $p \neq p'$. For instance, the verb `brag` has both positive and negative polarities in OF. For these cases, we apply Definition 2 to determine the polarity of word w with part of speech pos. The verb `brag` has negative polarity according to Definition 2. Such cases simply say that the team who constructs the dictionary believes `brag` has multiple polarities as they do not adopt our dominant sense principle. There are 58 such inconsistencies in GI, OF and AL. QW, a sentiment sense dictionary, does not have intra-inconsistencies as it does not have a synset with multiple polarities.

3.1.2 Inter-Dictionary Inconsistency

A word belongs to this category if it appears with different polarities in different SWDs. For instance, the adjective `joyless` has positive polarity in OF and negative polarity in GI. Table 1 depicts the overlapping relationships between the three SWDs: e.g., OF has 2,933 words in common with GI. The three dictionaries largely agree on the polarities of the words they pairwise share. For instance, out of 2,924 words shared by OF and GI, 2,834 have the same polarities. However, there are also a significant number of words

TABLE 1
Disagreement between Dictionaries

Pairs of Dictionaries	Word Polarity Inconsistency	Disagreement Overlap
OF & GI	90	2,924
OF & AL	73	1,150
GI & AL	18	712

which have different polarities across dictionaries. Case in point, OF and GI disagree on the polarities of 90 words. Among the three dictionaries there are 181 polarity inconsistent words. The polarities of these words are manually corrected using Definition 2 before the polarity consistency checking is applied to the union of the three dictionaries. This union is called *disagreement-free union*.

3.2 Complex Polarity Inconsistency

This kind of inconsistency is more subtle and cannot be detected by direct comparison of words/synsets. It consists of a *set* of words and/or synsets whose polarities cannot concomitantly be satisfied. Recall the example of `confute` and `disprove` in OF given in Section 1. Recall our argument that by assuming that WordNet is correct, it is not possible for the two words to have different polarities: the sole synset, which they share, would have two different polarities, which is a contradiction.

The occurrence of an inconsistency points out the presence of incorrect input data:

- the information given in WordNet is incorrect, or
- the information in the given sentiment dictionary is incorrect, or both.

Regarding WordNet, the errors may be due to (1) a word has senses that are missing from WordNet or (2) the frequency counts of some synsets are inaccurate. A comprehensive analysis of every synset/word with inconsistency is a tantalizing endeavor requiring not only a careful study of multiple sources (e.g., dictionaries such as Oxford and Cambridge) but also linguistic expertise. It is beyond the scope of this paper to enlist all potentially inconsistent words/synsets and the possible remedies. Instead, we limit ourselves to drawing attention to the occurrence of these issues through examples, welcoming experts in the area to join the corrective efforts. We give more examples of inconsistencies in order to illustrate additional discrepancies between the input dictionaries.

3.2.1 WordNet versus Sentiment Dictionaries

The adjective `bully` is an example of a discrepancy between WordNet and a sentiment dictionary. The word has negative polarity in OF and has a single sense in WordNet. The sense is shared with the word `nifty`, which has positive polarity in OF and has a unique sense. By applying Definition 2 to `nifty` we obtain that the sense is positive, which in turn, by Definition 2, implies that `bully` is positive. This contradicts the polarity of `bully` in OF. According to the Webster dictionary, the word has a sense (i.e., *resembling or characteristic of a bully*) which has a negative

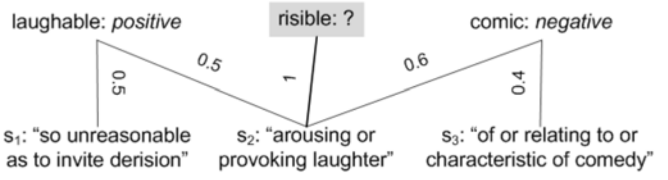

laughable: positive
risible: ?
comic: negative
0.5
0.5
1
0.6
0.4
s$_1$: "so unreasonable as to invite derision"
s$_2$: "arousing or provoking laughter"
s$_3$: "of or relating to or characteristic of comedy"

Fig. 1. Example of an inconsistency across sentiment dictionaries.

polarity, but it is not present in WordNet. The example shows the presence of a discrepancy between WordNet and OF, namely, OF seems to assign polarity to a word according to a sense that is not in WordNet.

3.2.2 Across Sentiment Dictionaries

We provide examples of inconsistencies across sentiment dictionaries here. Our first example is from SWDs. The adjective `comic` has negative polarity in AL and the adjective `laughable` has positive polarity in OF. Through deduction (i.e., by successive applications of Definition 2), we show that the adjective `risible`, which is not present in either of the dictionaries, is assigned negative polarity because of `comic` and is assigned positive polarity because of `laughable`. The relationship between these words in WordNet is depicted in Fig. 1. We note that because `comic` is negative and $\{s_2\}$ is the unique MDS among the synsets of `comic` it follows that $\{s_2\}$ is a polarity dominant subset. Thus, s_2 has negative polarity. Since $\{s_2\}$ is a polarity dominant subset for `risible` it follows that `risible` has negative polarity. In a similar vein, we can show that `risible` acquires a positive polarity because of `laughable`.

The second example illustrates that an SWD and an SSD may have contradicting information. The verb `intoxicate` has three synsets in WordNet, each with the same frequency. Hence, their relative frequencies with respect to `intoxicate` are $\frac{1}{3}$. On one hand, `intoxicate` has a negative polarity in GI. This means that $P_- > \frac{1}{2}$. On the other hand, two of its three synsets have positive polarity in QW. So, $P_+ = \frac{2}{3} > \frac{1}{2}$, which means that $P_- < \frac{1}{2}$. This is a contradiction. This example can also be used to illustrate the presence of a discrepancy between WordNet and sentiment dictionaries. Note that all the frequencies of use of the senses of `intoxicate` in WordNet are 0. The problem is that when all the senses of a word have a 0 frequency of use, wrong polarity inference may be produced.

4 Consistent Polarity Assignment

Given the above discussion, it clearly is important to find all polarity inconsistencies. This in turn boils down to finding those words such that there does not exist any polarity assignment to their synsets that is consistent with their polarities. It turns out that the complexity of the problem of assigning polarities to the synsets such that the assignment is consistent with the polarities of the input words, called `Consistent Polarity Assignment` (CPA) problem, is a "hard" problem, as described below. The problem is stated as follows:

Consider two sets of nodes of type synsets and type words, in which each synset of a word has a relative frequency of use with respect to the word. Each synset can be assigned a positive, negative or neutral polarity. A word has polarity p if it satisfies the hypothesis of Definition 2. The question to be answered is: Given an assignment of polarities to the words, does there exist an assignment of polarities to the synsets that agrees with the assignment of the polarities of the words?

In other words, given the polarities of a subset of words (e.g., given by one of the three SWDs) the problem of determining whether there exists an assignment of polarities to the synsets that agrees with this assignment is a "hard" problem.

Theorem 1. *The CPA problem is NP-complete.*

Proof. Proof given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2014.2339855. □

We note that the following related problem is solvable in polynomial time. Suppose that the polarities of all the synsets in the word-synset network $\mathcal{N}$ are given. The question whether there exist a polarity assignment to words such that the two assignments are consistent can always be answered in polynomial time. All we have to do is to compute P_+, P_- and P_0 for each word $w \in W$ and then apply Definition 2 in order to find a polarity assignment to the words consistent to that of the synsets.

5 Polarity Consistency Checking

To "exhaustively" solve the problem of finding all polarity inconsistencies in a sentiment word dictionary, we propose a solution that reduces an instance of the problem to an instance of CNF-SAT. We can then apply one of the fast SAT solvers (e.g., [20], [21]) to solve our problem. CNF-SAT is a decision problem of determining if there is an assignment of True and False to the variables of a Boolean formula Φ in conjunctive normal form (CNF) such that Φ evaluates to True. A formula is in CNF if it is a conjunction of one or more clauses, each of which is a disjunction. CNF-SAT is a classic NP-complete problem [13], but, modern SAT solvers are capable of solving many practical instances of the problem. Since, in general, there is no easy way to tell the difficulty of a problem without trying it, SAT solvers have the following termination outputs. A SAT solver terminates and returns "satisfiable" or "not satisfiable"; it can also terminate with a time-out without indicating anything about satisfiability. So, a SAT solver terminates even if it cannot find a solution.

In this section we describe two methods of converting an instance of the polarity consistency checking problem into an instance of CNF-SAT. The first method, called *exhaustive enumeration of MDS method* (EEM), was first described in [15] and is given here for completeness, while the second method, called *frequency summation method* (FSM), is new.

5.1 Conversion to CNF-SAT

The input consists of a sentiment word dictionary D and the word-synset network $\mathcal{N}$. We partition $\mathcal{N}$ into connected components and we consider only those components that have a word in D. For each synset s we introduce three Boolean variables s_-, s_+ and s_0, corresponding to the negative, positive and neutral polarities, respectively. In this section we use $-, +, 0$ to denote negative, positive and neutral polarities, respectively.

Now we describe the construction of the Boolean formula Φ corresponding to a connected component M of the word-synset network $\mathcal{N}$. Φ has the following clauses. First, for each synset s it has a clause $C(s)$ expressing that s can have only one of the three polarities:

$$C(s) = (s_+ \wedge \neg s_- \wedge \neg s_0) \vee (s_- \wedge \neg s_+ \wedge \neg s_0) \vee (s_0 \wedge \neg s_- \wedge \neg s_+).$$

Since a word has a neutral polarity if it has neither positive nor negative polarities, we have that $s_0 = \neg s_+ \wedge \neg s_-$. Replacing this expression in the above equation and applying standard Boolean algebra laws, we obtain

$$C(s) = \neg s_+ \vee \neg s_-. \tag{1}$$

For each word w with polarity $p \in \{-, +, 0\}$ in D and M we need a clause $C(w, p)$ that states that w has polarity p. Thus, the Boolean formula for a connected component M of the word-synset network N is:

$$\Phi = \bigwedge_{s \in M} C(s) \wedge \bigwedge_{(w,p) \in D, w \in M} C(w, p). \tag{2}$$

From Definition 2, w is neutral if it is neither positive nor negative. Hence, $C(w, 0) = \neg C(w, -) \wedge \neg C(w, +)$. Thus, we need to define only the clauses $C(w, -)$ and $C(w, +)$, which correspond to w having negative and positive polarities, respectively. Therefore, herein $p \in \{-, +\}$, unless otherwise specified.

The two methods (i.e., EEM and FSM) differ in the way they express $C(w, p)$. EEM is based on the following statement in Definition 2: w has polarity p if there exists a polarity dominant subset among its synsets. Thus, in this method, $C(w, p)$ is defined by enumerating all the minimally dominant subsets of synsets of w. If at least one of them is a polarity dominant subset then $C(w, p)$ evaluates to True. The advantage of this approach is that it is easy to implement. Its disadvantage is that it generates formulas that have exponential lengths. For example, with this approach the word `give` has 627,527 minimally dominant subsets of synsets, which cannot be handled with this approach because the SAT solvers time-out (see Section 9.6). Although less than 3 percent of the words in WordNet have large sets of minimally dominant subsets of synsets, we still need a solution that accounts for these words. We give another transformation (i.e., FSM) that generates a Boolean formula of polynomial length. This transformation is based on the following observation. An equivalent way of expressing that w has polarity $p \in \{-, +\}$ is

$$\sum_{s \in S_w} f(w, s)\, pol(s, p) > \frac{1}{2} \sum_{s \in S_w} f(w, s) = \frac{1}{2} freq(w), \tag{3}$$

where $pol(s, p) = 1$, if s has polarity p and $pol(s, p) = 0$, otherwise. $pol(s, p)$ is a new Boolean variable introduced. Note that $\frac{1}{2} freq(w)$ is a constant, which is known for each word. For example, for the word `cheap` it is $\frac{11.4}{2}$.

The idea is to construct a Boolean formula for $C(w, p)$ such that $C(w, p)$ = True if the above inequality is satisfied and $C(w,p)$ = False when it is not. The disadvantages of this solution are that it introduces a large number of variables and is difficult to implement. We describe the methods in the following two sections.

5.2 Exhaustive Enumeration of MDS Method

We now elaborate the construction of $C(w,p)$ for our first method. In this method, we enumerate all the MDS of w and for each of them we introduce a clause. The clauses are then concatenated by OR in the Boolean formula. Let $C(w,p,T)$ denote the clause for an MDS T of w, when w has polarity $p \in \{-,+\}$. Hence,

$$C(w,p) = \bigvee_{T \in MDS(w)} C(w,p,T), \tag{4}$$

where $MDS(w)$ is the set of all minimally dominant subsets of synsets of w.

For each MDS T of w, the clause $C(w,p,T)$ is the AND of the variables corresponding to polarity p of the synsets in T. That is,

$$C(w,p,T) = \bigwedge_{s \in T} s_p, p \in \{-,+\}. \tag{5}$$

The formula Φ is not in CNF after this construction and it needs to be converted. The conversion to CNF is a standard procedure [22] and we omit it throughout the paper. Φ in CNF is input to a SAT solver.

Example 1. Consider a connected component consisting of the words $w =$ `cheap`, $v =$ `inexpensive` and $u =$ `sleazy`. `cheap` has a positive polarity, whereas `inexpensive` and `sleazy` have negative polarities. The synsets of these words are: $\{s^1, s^2, s^3, s^4\}$, $\{s^1\}$ and $\{s^3, s^4, s^5\}$, respectively (refer to WordNet), where some synsets are shared among the three words. The relative frequencies of use of s^3, s^4 and s^5 w.r.t. `sleazy` are all equal to 1/3. We have 15 binary variables, 3 per synset, $s^i_-, s^i_+, s^i_0, 1 \leq i \leq 5$. The only minimally dominant subset of synsets of `cheap` is $\{s^1\}$, which coincides with that of `inexpensive`. Those of `sleazy` are $\{s^3, s^4\}$, $\{s^3, s^5\}$ and $\{s^4, s^5\}$. For each s^i we need a clause $C(s^i)$. Hence, $C(w,+) = s^1_+$, $C(v,-) = s^1_-$ and $C(u,-) = (s^3_- \wedge s^4_-) \vee (s^3_- \wedge s^5_-) \vee (s^4_- \wedge s^5_-)$. Thus,

$$\begin{aligned}\Phi = \bigwedge_i C(s^i) \wedge \big[s^1_+ \wedge s^1_- \wedge \big(\big(s^3_- \wedge s^4_-\big) \\ \vee \big(s^3_- \wedge s^5_-\big) \vee \big(s^4_- \wedge s^5_-\big)\big)\big].\end{aligned} \tag{6}$$

Φ is not in CNF and needs to be converted. Φ is True if the clauses $C(w,+) = s^1_+$ and $C(v,-) = s^1_-$ are True. But, this makes $C(s^1)$ False. Hence, Φ is not satisfiable. The clauses $C(w,+) = s^1_+$ and $C(v,-) = s^1_-$ are unsatisfiable and thus the polarities of `cheap` and `inexpensive` are inconsistent.

5.3 Frequency Summation Method

In this section we describe our new method for reducing an instance of the PCC problem into an instance of CNF-SAT, which gives a polynomial length formula for $C(w,p)$. We start by giving the intuition of the solution and then present the formal derivation.

The idea is to simulate a logic circuit that evaluates Inequality 3 and outputs true when this inequality is satisfied and false when it is not. Then, we derive the Boolean expression associated with the circuit. A careful analysis of the inequality reveals that we need three main circuit components: a *SUM* component that computes the summation $\sum_{s \in S_w} f(w,s)pol(s,p)$, an *Instantiation* component that evaluates each term $f(w,s)pol(s,p)$ before it is input to the SUM component and a *Digital Comparator* component that asserts the inequality. Bottom up, the logic circuit is constructed as follows:

1) For each $s \in S_w$, we need an Instantiation component I_s. The inputs of I_s are $f(w,s)$ and s_p. I_s outputs $f(w,s)$ if $s_p = True$ (i.e., the synset s has polarity p) and outputs 0 if $s_p = False$ (i.e., s does not have polarity p).
2) The SUM component adds the outputs of I_s's pairwise; then, it adds their results pairwise; so on. This scheme can be captured as a full binary tree whose leaf nodes denote the frequencies of use of the synsets and whose internal nodes represent the sum of values of the frequencies.
3) The output of SUM is input together with the constant $\frac{1}{2}freq(w)$ to the Digital Comparator.

A tree arrangement of adders is used in practice to minimize adder propagation delays [23], because the depth of the circuit is only $O(log(n))$, whereas it is $O(n)$ in case of a linear scan adder. n is the number of operands.

We now give the formal derivation of the Boolean formula of $C(w,p)$ for a word w and $p \in \{-,+\}$, according to the above logic circuit. The logic circuit operates on binary numbers. We therefore need to express all the terms in Inequality 3 into binary numbers: that is, the frequencies of use of synsets $f(w,s)$ and $\frac{freq(w)}{2}$. In general, it is a lot easier to manipulate binary representations for integers than binary representations for rational numbers. All the terms in Inequality 3 are integers, except for $\frac{freq(w)}{2}$ and the terms modified by smoothing (Section 2). The former can be easily addressed by replacing $\frac{freq(w)}{2}$ with $\lfloor\frac{freq(w)}{2}\rfloor$ in Inequality 3 without altering the "meaning" of the inequality. If a synset s of w is such that $f(w,s) = 0$ then we first add $\epsilon = 0.1$ to all $f(w,s'), (w,s) \in \mathcal{E}$ ($\mathcal{E}$ was defined in Section 2.1, word-synset network). Then, we multiply by 10 all the frequencies of the synsets of w, i.e., $f(w,s') \leftarrow 10f(w,s), \forall(w,s') \in \mathcal{E}$. After these two operations Inequality 3 will be of the form:

$$\sum_{s \in S_w} f(w,s)\, pol(s,p) > \left\lfloor \frac{freq(w)}{2} \right\rfloor, \tag{7}$$

with all the terms being integers. For example, for the word `cheap` the frequencies of use of its four synsets initially are $f_1 = 9, f_2 = 1, f_3 = 1$ and $f_4 = 0$. After we perform the above steps they will become $f_1 = 91, f_2 = 11, f_3 = 11$ and $f_4 = 1$, while $freq($`cheap`$) = 114$. For each synset s of a word w we assign k_s binary variables to represent the frequency $f(w,s)$.

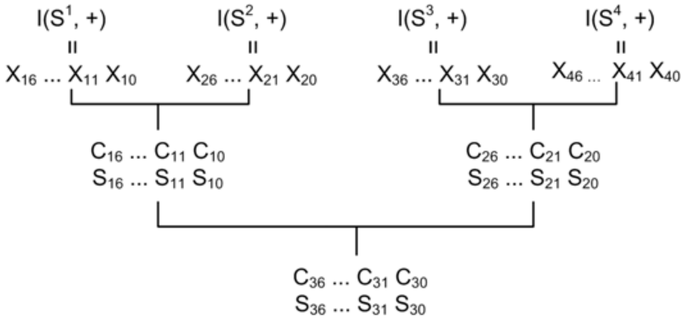


Fig. 2. The binary tree summation of the frequencies of the synsets of the word cheap.

Since each frequency is smaller than $freq(w)$, then $k_s \le \lceil log_2(freq(w)\rceil + 1$. For ease of exposition, we assign the same number of variables to each s of w, $k_s = k = \lceil log_2(freq(w)\rceil + 1$. In our running example, each of the four synsets is assigned $\lceil log_2(114)\rceil + 1 = 7$ binary variables. We denote by $bf(w, s)$ the binary representation of $f(w, s)$. For instance, 1011011 is the binary representation of $f_1 = 91$. It is sometimes easier to work with a vector notation for the binary variables associated with a synset. We denote by $\mathbf{x_s}$ the vector of binary variables associated with the synset s. In addition, $\neg\mathbf{x_s}$ is a vector of binary variables $\mathbf{y_s}$ such that $y_s^i = \neg x_s^i, \forall i \in [1..k]$.

Instantiation. We need to capture the semantics of the expression $f(w, s)\,pol(s, p)$ in a CNF expression. That is, if $s_p = True$ then the binary variables corresponding to the synset s are set to $bf(w, s)$ (i.e., $s_p \rightarrow (\mathbf{x_s} = bf(w, s))$), otherwise they are set to 0 (i.e., $\neg s_p \rightarrow (\mathbf{x_s} = 0)$). This is captured in the following formula:

$$I(s,p) = (s_p \rightarrow (\mathbf{x_s} = bf(w,s))) \wedge (\neg s_p \rightarrow (\mathbf{x_s} = 0)) \\ \Leftrightarrow (\neg s_p \vee (\mathbf{x_s} = bf(w,s))) \wedge \left(\bigwedge_{i=1}^{k} (s_p \vee \neg x_s^i) \right). \tag{8}$$

$\mathbf{x_s} = bf(w, s)$ is the short notation for $x_s^1 = bf(w, s)[1] \wedge x_s^2 = bf(w, s)[2] \wedge ... \wedge x_s^k = bf(w, s)[k]$, where x_s^i and $bf(w, s)[i]$ are the ith entries in the binary vectors $\mathbf{x_s}$ and $bf(w, s)$, respectively. In addition, $\mathbf{x_s} = \mathbf{0} \equiv x_s^1 = 0 \wedge x_s^2 = 0 \wedge ... \wedge x_s^k = 0 \equiv \neg x_s^1 \wedge \neg x_s^2 \wedge ... \wedge \neg x_s^k$. In Formula 8, the clause $s_p \rightarrow (\mathbf{x_s} = \mathbf{0}) \equiv s_p \rightarrow (\neg x_s^1 \wedge \neg x_s^2 \wedge ... \wedge \neg x_s^k) \equiv \neg s_p \vee (\neg x_s^1 \wedge \neg x_s^2 \wedge ... \wedge \neg x_s^k) \equiv \bigwedge_{i=1}^{k} (s_p \vee \neg x_s^i)$.

Example 2. The word cheap has a positive polarity. We exemplify the above formula for its first synset, i.e., $I(s^1, +)$. Recall that $f(\text{cheap}, s^1) = 91$ and $bf(\text{cheap}, s^1) = 1011011$. We assign a vector $\mathbf{x}$ of 7 binary variables to s^1. We have:

$$I(s^1,+) = (s_+^1 \rightarrow (\mathbf{x} = 1011011)) \wedge (\neg s_+^1 \rightarrow (\mathbf{x} = \mathbf{0})) \equiv \\ ((\neg s_+^1 \vee x_6) \wedge (\neg s_+^1 \vee \neg x_5) \wedge (\neg s_+^1 \vee x_4) \wedge (\neg s_+^1 \vee x_3) \wedge \\ (\neg s_+^1 \vee \neg x_2) \wedge (\neg s_+^1 \vee x_1) \wedge (\neg s_+^1 \vee x_0)) \wedge \bigwedge_{i=0}^{6} (s_+^1 \vee \neg x_s^i). \tag{9}$$

SUM. Now that the binary variables for the frequencies are initialized, we have to add them up. As mentioned previously, we add them pairwise. Fig. 2 illustrates the addition process for the frequencies of the four synsets of the word cheap. Namely, we first compute $S_1 = I(s^1, +) + I(s^2, +)$ and $S_2 = I(s^3, +) + I(s^4, +)$. We then compute $S_3 = S_1 + S_2$, which is the desired sum. These additions are represented as a full binary tree whose leaf nodes denote the frequencies of the synsets. Corresponding to each internal node, we have k summation variables and k carry variables.

In effect, at each internal node we simulate a *ripple carry adder* circuit [23]. The circuit adds two k-bit numbers and has k *one-bit full adders* [23]. A one-bit full adder adds three one-bit numbers, X, Y, and C_{in}, where X and Y are the operands, and C_{in} is a bit carried in from a past addition. The circuit produces a two-bit output represented by S (summation) and C_{out} (carry), where

$$S = (\neg X \wedge \neg Y \wedge C_{in}) \vee (\neg X \wedge Y \wedge \neg C_{in}) \\ \vee (X \wedge \neg Y \wedge \neg C_{in}) \vee (X \wedge Y \wedge C_{in}) \tag{10}$$

$$C_{out} = (X \wedge Y) \vee (X \wedge C_{in}) \vee (Y \wedge C_{in}). \tag{11}$$

These two formulas are not in CNF and they need to be converted into CNF formulas. A stated above, we omit this step.

For each internal node we have a CNF Boolean formula that consists of $2k$ clauses: k clauses defining the values of the k summation variables and k clauses defining the values of the k carry variables in terms of the values of the variables of their children. The Boolean formula for the SUM component, denoted $S(w, p)$, is the conjunction of the Boolean formulas at internal nodes.

Digital comparator. A digital comparator takes two numbers as input in binary form and determines whether one number is greater than, less than or equal to the other number [23]. For a simple 1-bit comparator, we have that $x \wedge \neg b \Rightarrow x > b$ and $\neg(x \wedge \neg b \vee \neg x \wedge b) \Rightarrow x = b$. A k-bit comparator can be constructed by cascading together k 1-bit comparators. Let $\mathbf{x}$ and $\mathbf{b}$ be two k-bit words. First, we define $E_i = \neg(x_i \wedge \neg b_i \vee \neg x_i \wedge b_i)$ and $D_i = x_i \wedge \neg b_i$, $i \in [0..k-1]$. Then, noting that the bit in position 0 is the most significant bit, the formula $DC(\mathbf{x}, \mathbf{b})$ to assert that $\mathbf{x}$ is greater than $\mathbf{b}$ is given by:

$$D_0 \vee (E_0 \wedge D_1) \vee ... \vee (E_0 \wedge ... \wedge E_{k-2} \wedge D_{k-1}). \tag{12}$$

In plain words, the formula says that $\mathbf{x}$ is grater than $\mathbf{b}$ if $x_0 > b_0$ or if $x_0 = b_0$ and $x_1 > b_1$ or if $x_0 = b_0$, $x_1 = b_1$ and $x_2 > b_2$, so on so forth (recall lexicographic order).

$DC(\mathbf{x}, \mathbf{b})$ is further simplified because $\mathbf{b}$ is a constant binary number, which corresponds to $\lfloor \frac{freq(w)}{2} \rfloor$. So, after applying the Boolean laws between the constants True/False and a binary variable (e.g., $True \wedge x = x$), $DC(\mathbf{x}, \mathbf{b})$ simplifies and contains only the variables $x_i, i \in [0..k-1]$. $DC(\mathbf{x}, \mathbf{b})$ needs to be converted into a CNF formula. We omit this step.

6 Word-Sense Polarity Consistency

We present here a methodology for polarity consistency checking between a sentiment word dictionary (e.g., OF) SWD and a sentiment sense dictionary (e.g., QW) SSD. It consists of the following steps: (1) instantiation of binary variables, (2) formula reduction and (3) satisfiability test.

We assume that the set of inconsistent words in SWD was already identified (using one of the methods described above) and discarded from SWD. These words are inconsistent regardless of the polarity assignments to their underlying synsets in SSD.

Instantiation of binary variables. Let Φ be a Boolean formula obtained with either of the methods described above for a connected component. Φ represents only the polarities of the words in the SWD. Consequently, we first need to reflect the polarities assigned to synsets in SSD in Φ. Recall that for each synset there are three binary variables, corresponding to the three possible polarity assignments. If a synset s has polarity $p, p \in \{+,-,0\}$, in SSD, then each occurrence of the binary variable s_p in Φ is replaced with the Boolean value True. The clause $C(s)$ (Equation (1)) evaluates to True.

For illustration, we use the formula from Example 1:

$$\Phi = \bigwedge_{i\in\{1..5\}} C(s^i) \wedge \left[s^1_+ \wedge s^1_- \wedge \left(\left(s^3_- \wedge s^4_-\right) \vee \left(s^3_- \wedge s^5_-\right) \vee \left(s^4_- \wedge s^5_-\right)\right)\right].$$

Suppose that the synset s^5 has polarity positive in SSD. Then, $s^5_+ = 1$ (True), $s^5_- = s^5_0 = 0$ (False). $C(s^5) = 1$. The formula becomes:

$$\Phi = \bigwedge_{i\in\{1..5\}} C(s^i) \wedge \left[s^1_+ \wedge s^1_- \wedge \left(\left(s^3_- \wedge s^4_-\right) \vee \left(s^3_- \wedge 0\right) \vee \left(s^4_- \wedge 0\right)\right)\right].$$

Formula reduction. The new formula contains both variables and constants (i.e., True and False). We need to reduce it to a constant-free formula. We reduce Φ to a constant-free formula using well known Boolean logic laws, e.g., $True \wedge C = C$. We apply them repeatedly until Φ has no constants or the entire formula reduces to a constant. The above formula becomes,

$$\Phi = \bigwedge_{i\in\{1..4\}} C(s^i) \wedge \left[s^1_+ \wedge s^1_- \wedge \left(s^3_- \wedge s^4_-\right)\right].$$

Satisfiability test. If the new formula does not reduce to a constant then it is input to the SAT solver. If it reduces to a constant, there is no need to use the SAT solver, because if the formula reduces to True then the formula is satisfied and if it reduces to False then the formula is not satisfied. In the latter case, the synsets in SSD and the words in SWD present in the formula are reported as inconsistent. For example, Φ evaluates to False if the polarity of the synset s^1 is set to positive.

7 Complexity Analysis of the Methods

In this section, we analyze the complexity of the Boolean formulas generated with the two methods. We start with the analysis of EEM.

7.1 Complexity Analysis of EEM

This method generates a formula, which has double exponential number of clauses in the worst case for a word. The reason is that we first generate a SAT formula that has exponential length in the number of clauses (see Equation (4)). This formula however is not in CNF and it needs to be converted to CNF. This in general can cause another exponential blow up. Thus, the overall blow up can be double exponential in the worst case. Because of this, we cannot handle the entire WordNet with it (see Section 9.6).

7.2 Complexity Analysis of FSM

We now show that the formula generated by FSM is of polynomial length in the number of clauses. Suppose that we have a word with m synsets. Corresponding to each internal node in the binary tree, we have $k = \lceil log_2(freq(w)) \rceil + 1$ variables representing the binary representation of the number associated with the node. For each such node we have a set of clauses that defines the values of these variables in terms of the values of the variables corresponding to its two children; we also use k additional auxiliary variables that denote the carry bits when the numbers of the children are added. The value of each bit in the sum is defined as a Boolean formula of the values of the corresponding bits of the two summands and the carry bit corresponding to the previous bit. Thus, this formula for each bit is a formula over four variables and is obtained directly in CNF. Similarly, we obtain formulas for each carry bit also. The conjunction of all these $2k$ formulas specifies the values of the bits in the sum in terms of values of bits in its arguments, i.e., the children. Note that each of these $2k$ formulas is of constant length. Hence the over all length of the formula is $O(k)$. Since there are at most m such internal nodes, the overall length of the formula is $O(mk)$. The formula for the digital comparator and that for the instantiation component are linear in k. Thus, the overall length of the translation is $O(mk + k)$. We note that this formula is in CNF.

7.3 A Hybrid Approach

One drawback of FSM is that it may generate Boolean formulas with a large number of variables (thousands). This is particularly the case for words with large number of synsets whose frequencies of use are large decimal numbers. For example, the formula derived for the verb `make` has over 1,800 binary variables. The reason is that the word has 49 synsets, the largest frequency of use among its synsets is 508 and has synsets with frequency of use 0 (before smoothing). While for the verb `make` we have to accept this large number of variables because EEM cannot handle it, for words such as the adjective `cheap`, which requires 70 variables, we can do better.

We explain first why the adjective `cheap` requires 70 binary variables. The adjective `cheap` has four synsets and 9 is the largest frequency of use among its synsets. `Cheap` also has a synset whose frequency of use is 0, before smoothing. Because of this synset, we need to first add 0.1 to the frequencies of use of all the synsets of `cheap` and second, to multiply the frequencies of use by 10. Hence, 9 becomes 91 and $91 = 1011011_2$. Thus, we need seven binary variables for each of the four synsets. In total, for the four synsets of `cheap` we need $4 \times 7 = 28$ binary variables. This is also illustrated in Example 2. Recall that we need to sum up the frequencies of the synsets using a binary tree scheme (see Fig. 2). For `cheap`, the tree has three internal nodes. At each internal node we need $2 \times 7 = 14$ binary variables. Hence, the internal nodes introduce $3 \times 14 = 42$ binary variables. So, in total we need $28 + 42 = 70$ variables to represent `cheap` and its synsets in the Boolean formula constructed with FSM. In contrast, the formula derived with EEM for the adjective `cheap` is very simple. EEM uses only eight binary variables (2 per each of the four synsets). In addition, because the synset whose frequency of use is 9 is a dominant synset, Equation (4) has only one clause. In general, the formulas generated by EEM for the words with "small" number of synsets is significantly simpler and shorter than the ones generated by FSM. Because, about 97 percent of the words in WordNet have at most 4 synsets, in the hybrid approach we propose to utilize EEM to construct the Boolean formulas corresponding to these words and FSM for the rest of the words, i.e., the words having at least five synsets.

For a connected component M whose set of words is W, we proceed as follows to construct the formula Φ (Equation(2)) with the hybrid approach:

1) *partition W into two sets.* W_4, the subset of words in W with at most four synsets, and W_5, the subset of words in W with at least five synsets. Φ becomes:

$$\bigwedge_{s \in M} C(s) \wedge \bigwedge_{\substack{(w,p)\in D \\ w \in W_4}} C(w,p) \wedge \bigwedge_{\substack{(w,p)\in D \\ w \in W_5}} C(w,p). \tag{13}$$

2) The expression of each $C(w,p)$ in the term $\bigwedge_{(w,p)\in D,}$ $w \in W_4 C(w,p)$ is defined using Formulas (4) and (5) (i.e., the formulas from EEM).
3) The expression of each $C(w,p)$ in the term $\bigwedge_{(w,p)\in D, w\in W_5} C(w,p)$ is defined using Formulas (2), (10), (11) and (12) (i.e., the formulas from FSM).

The formula Φ so obtained is input into a SAT solver. The hybrid approach generates significantly shorter formulas than either FSM and EEM, and can handle all the words in WordNet.

8 Detecting Inconsistencies

In this section we describe the general procedure for detecting the words with polarity inconsistencies using the output of a SAT solver. When a formula is unsatisfiable, a SAT solver returns a *unsatisfiable core* (UC) from the original formula. An *unsatisfiable core* is a small unsatisfiable subset of clauses of the formula. An unsatisfiable core is minimal, called *minimal unsatisfiable core* (MUC), if it becomes satisfiable whenever any one of its clauses is removed. Some modern SAT solvers return a MUC from an unsatisfiable formula. There are no known optimal algorithms for computing the *minimum unsatisfiable core* [24]. In our problem an UC corresponds to a set of polarity inconsistent words. The argument is as follows. Consider W the set of words in a connected component and Φ the CNF formula generated with one of the above methods. During the transformation we keep track of the clauses introduced in Φ by each word. Suppose Φ is inconsistent. Then, the SAT solver returns a UC. Each clause in the UC is mapped back to its corresponding word. We obtain the corresponding subset of words $W', W' \subseteq W$. Since, the set of clauses in the UC is a subset of those in Φ and the UC is unsatisfiable it follows that the words in W' are inconsistent.

Consider Formula 6 in Example 1, which is unsatisfiable. One of its MUCs is $\Phi' = (\neg s^1_+ \vee \neg s^1_-) \wedge s^1_+ \wedge s^1_- \equiv C(s^1) \wedge C(w,+) \wedge C(v,-)$. The clauses $C(w,+)$ and $C(v,-)$ correspond to w and v, respectively. Thus, $W' = \{w, v\}$ is the set of inconsistent for Φ'.

Notice that a SAT solver cannot pinpoint which are the clauses, and consequently the words, that need to be changed to make a MUC satisfiable. This is to some extent a subjective issue. The output of the SAT solver needs to be viewed as saying that the set of words W' cannot all have the specified polarity in the input SWD in the same time; the polarities of some words in W' are wrong in the SWD. In our example, the SAT solver does not tell us whether the polarity of w or that of v needs to be changed to make Φ consistent. It only tells us that their polarities are not consistent to each other.

An unsatisfiable formula may contain multiple MUCs, and fixing any single MUC may not make the formula satisfiable. As long as any MUC is present in the formula, it will remain unsatisfiable. So, it is valuable to find the set of all MUCs of an unsatisfiable formula. Most SAT solvers return a single MUC per unsatisfiable formula. Finding all MUCs is a very difficult problem [14] and no open source SAT solver possesses this functionality. We use the open source SAT solvers SAT4j [25] and PicoSAT [26]. The former returns an UC and the latter returns a MUC for an unsatisfiable formula. PicoSAT uses the utility PicoMUS to compute a MUC. These two solvers return only one UC per unsatisfiable formula. Consequently, we only find a *single* subset of *all* possible subsets of inconsistent words within a connected component. In the experimental study we report the sets of inconsistent words produced after running the two SAT solvers once on the input sentiment dictionaries.

9 Experiments

The chief goal of the experimental study is to show that our techniques can identify considerable inconsistencies in sentiment dictionaries. We show that the hybrid method is significantly faster than FSM and both are faster than EEM, making our new solutions usable in practice. We also give more details about the inconsistencies: we show that most of the sets of inconsistent words have up to four words. This translates into a reduced workload for an expert who decides to analyze them.

9.1 Data Sets

In our experimental study, we use WordNet 3.0, the SWDs GI, OF and AL; and the SWDs QW and SWN. The statistics about the data sets are given in Table 2. The table shows the distribution of the words and synsets per part of speech. Columns 2 and 3 pertain to WordNet. For example, there are 21,479 adjective words, which have 18,156 unique synsets. In total WordNet 3.0 has 155,287 words and 117,659 unique synsets.

We discard the entries in each SWD that do not appear in WordNet. After this cleaning step, as shown in Table 2, there are 3,724 entries in GI, 1,759 entries in AL and 6,791 entries in OF which appear in WordNet.

TABLE 2
Distribution of Words and Synsets

POS	Words	Synsets	OF	GI	AL	QW
Noun	117,798	82,115	1,907	1,444	2	7,403
Verb	11,529	13,767	1,501	1,041	0	4006
Adjective	21,479	18,156	2,608	1,188	1,440	4050
Adverb	4,481	3,621	775	51	317	40
Total	**155,287**	**117,659**	**6,791**	**3,724**	**1,759**	**15,499**

Herein, whenever we refer to theses dictionaries we refer to their cleaned versions.

9.2 Inconsistency Detection

Recall that an inconsistency is a set of words whose polarities cannot all be concomitantly satisfied. Therefore, there are two ways to report inconsistencies: (1) by counting their occurrences and (2) by the total number of words involved in them. We report our experimental study using (2) in this section. In Section 9.3 we further analyze inconsistencies based on their size distribution. We use all three methods in our experiments. We apply them to (1) each of AL, GI and OF; (2) the disagreement-free union (UF); (3) each of AL, GI and OF together with QW and (4) the disagreement-free union and QW. The experiments are conducted with the SAT4j SAT solver, unless explicitly stated.

9.2.1 *Intra-Dictionary Inconsistencies*

Table 3 summarizes the outcome of the experimental study for (1) and (2) with the three methods. The table shows the total number of words involved in all inconsistencies per reduction method broken down by the parts the speech. For example, FSM finds a total of 240, 14 and 2 polarity inconsistent words in OF, GI and AL, respectively. The ratio between the number of inconsistent words and the number of input words used for deduction is the highest for OF and the lowest for AL. We note that the three methods find almost the same number of inconsistent words, regardless of the input SWD. We do not find any inconsistencies for nouns and verbs in AL because AL has 2 and 0 entries (Table 2), respectively, for these parts of speech. A surprising observation from Table 3 is that EEM finds more inconsistent words than either FSM or Hybrid, despite not being able to handle the entire WordNet. As we will show in Section 9.3, all three techniques find the same number of inconsistent sets of words, but we get different numbers of inconsistent words due to the different sizes of MUCs. The explanation for the surprising behavior of EEM is as follows. A number of connected components contain words with large numbers of synsets, such as `give` and `make`. On one hand, the polarity assignments to the words in these connected components are found to be consistent using the formulas generated with either FSM or Hybrid. On the other hand, EEM generates such long formulas for these connected components that the SAT solvers times-out. Nonetheless, EEM generates small enough formulas for the connected components with unsatisfiable formulas that the SAT solvers can decide on their satisfiability. Note that we know which connected components have (un)satisfiable formulas because of FSM (or Hybrid), which always terminates without time-out in our experiments.

TABLE 3
Intra- and Inter-Sentiment Word Dictionaries Inconsistency

	EEM				FSM				HYBRID			
POS	OF	GI	AL	UF	OF	GI	AL	UF	OF	GI	AL	UF
Noun	23	4	0	27	23	4	0	27	23	4	0	27
Verb	66	2	0	63	60	2	0	65	64	2	0	64
Adj.	90	8	0	90	95	8	0	95	92	8	0	89
Adv.	61	0	2	69	62	0	2	65	60	0	3	62
Total	**240**	**14**	**2**	**249**	**240**	**14**	**2**	**252**	**239**	**14**	**3**	**242**

TABLE 4
Intra- and Inter-Dictionaries Inconsistency

	OF		GI		AL		UF	
	QW	SWN	QW	SWN	QW	SWN	QW	SWN
Noun	119	1,598	61	1,302	42	2	140	2,030
Verb	113	1,167	67	966	0	0	137	1,489
Adj.	170	1,808	48	845	0	985	177	2,251
Adv.	1	680	0	47	0	305	1	855
Total	**403**	**5,253**	**176**	**3,160**	**42**	**1,292**	**455**	**6,625**

9.2.2 *Inconsistencies in the Disagreement-Free Union*

The union dictionary has 7,794 entries, i.e., triplets of the form $\langle word, pos, polarity\rangle$. We manually corrected the polarities of 159 words, to the best of our understanding. These words have different polarities across dictionaries: e.g., the adjective `joyless` has positive polarity in OF and negative in GI. EEM, FSM and HYBRID discover 249, 252 and 242 inconsistencies, respectively, in the union dictionary. So, in effect the union dictionary has at least 252 + 159 = 411 polarity inconsistent words. Recall however that generating all MUCs is an "overkill" and the SAT solvers do not implement such a functionality. In general, SAT solvers are used in an interactive-iterative manner. That is, the errors pointed out by a SAT solver via a MUC need to be corrected; then, the new improved formula is re-evaluated by the SAT solver. If an error is still present a new MUC is reported, and the process repeats until the formula has no errors; i.e., until the input sentiment dictionary is consistent.

9.2.3 *Inter-Dictionary Inconsistencies: SWD versus SSD*

We also paired QW and SWN with each of the sentiment word dictionaries. Table 4 presents the outcome of this study. From the fourth and sixth columns in Table 4, we observe that the polarities assigned to the words in GI and AL largely agree with the polarities assigned to the synsets in QW. This is expected for AL because it has only two nouns and no verb (Table 2), while QW has only 40 adverbs (Table 2). Consequently, these two dictionaries have limited "overlapping". Besides, the adjectives in AL and the adjective synsets in QW are distributed over 612 and 351 connected components, respectively, in WordNet. Hence, many of the words in AL cannot be checked against the synsets in QW. This is true for OF, GI and the union dictionaries, too. The union dictionary and QW have nonetheless substantial

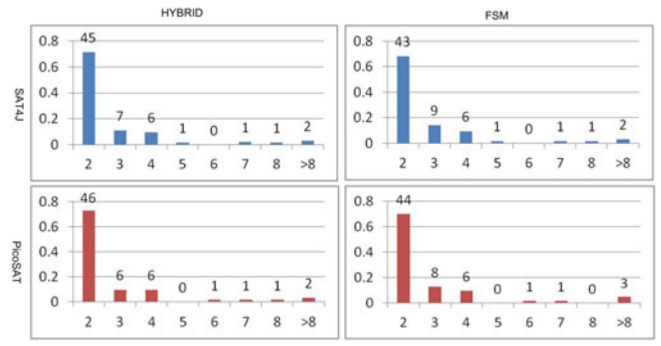


Fig. 3. The size distribution of polarity inconsistencies.

inconsistencies: the polarity of 455 words in the union dictionary disagrees with the polarities of their underlying synsets in QW. The numbers for SWN follow about the same trend as those obtained for QW, but they are several orders of magnitude larger. The main reason is that SWN covers the entire WordNet whereas QW covers only a fraction of it.

9.3 Size Distribution of Discovered Inconsistencies

From the corrective task perspective, the fewer words a polarity inconsistency consists of, the easier it is to analyze it and correct it; the larger it is the more challenging is to analyze it and suggest corrections. We report the polarity inconsistencies based on the distribution of their cardinalities (sizes) in this section. We show that the distribution of the cardinalities of inconsistencies is independent of the reduction method and the choice of the SAT solver. We consider the outputs of FSM and HYBRID and use both SAT4j and PicoSAT SAT solvers. Fig. 3 presents the results of this experimental study. All methods, including EEM, find 63 inconsistencies. These correspond to the number of connected components with unsatisfiable formulas. (Recall that we have a MUC per unsatisfiable formula and a Boolean formula per connected component.) Most of the inconsistencies have up to four words: about 92 percent of them. Only two of the discovered polarity inconsistencies (i.e., 0.3 percent) are large containing between 43 and 50 words, depending on the reduction method and the choice of SAT solver.

The key observation from this and the previous set of experiments is that the number of polarity inconsistencies are the same regardless of the reduction method and the choice of SAT solver. This corresponds to the number of connected components with unsatisfiable formulas. The number of words in these inconsistencies however may vary with the reduction method (Fig. 3) and the SAT solver (Fig. 3). In our experience, the variation is mostly due to the algorithms for computing MUCs in the SAT solvers. The reason is that an unsatisfiable formula may have multiple MUCs. Since these algorithms are heuristic in nature, for the same unsatisfiable formula for a set of sentiment words A they may find different MUCs. Consequently, we will find different subsets of inconsistent words in A with the property that if any one of the words is removed the rest of the words are polarity consistent. This problem would not occur if there was an algorithm for finding the *minimum* unsatisfiable core, because this would give us the *minimum* subset of inconsistent words in A.

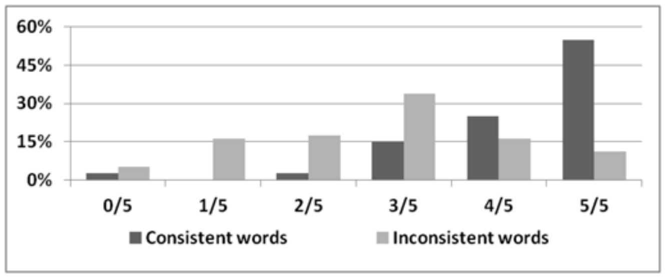


Fig. 4. Human classification of (in)consistent words.

9.4 Sentence Level Evaluation

We took 10 pairs of inconsistent words per part of speech; in total, we collected a set IW of 80 inconsistent words. Let $\langle w, pos, p\rangle \in IW$, p is the polarity of w. We collected five sentences for $\langle w, pos\rangle$ from the set of snippets returned by Google for query w. We parsed the snippets and identified the first five occurrences of w with the part of speech pos. Then two graduate students with English background analyzed the polarities of $\langle w, pos\rangle$ in the five sentences. The evaluators have worked together to align their evaluation criteria. We counted the number of times $\langle w, pos\rangle$ appears with polarity p and polarities different from p. We defined an agreement scale: total agreement (5/5), most agreement (4/5), majority agreement (3/5), majority disagreement (2/5), most disagreement (1/5), total disagreement (0/5). We computed the percentage of words per agreement category. We repeated the experiment for 40 randomly drawn words (10 per part of speech) from the set of consistent words. In total 600 sentences were manually analyzed. Fig. 4 shows the distribution of the (in)consistent words. For example, the annotators totally agree with the polarities of 55 percent of the consistent words, whereas they only totally agree with 16 percent of the polarities of the inconsistent words. The graph suggests that the annotators disagree to some extent (total disagreement + most disagreement + major disagreement) with 40 percent of the polarities of the inconsistent words, whereas they disagree to some extent with only 5 percent of the consistent words. We also manually investigated the senses of these words in WordNet. We noted that 36 of the 80 inconsistent words (45 percent) have missing senses in WordNet according to one of these English dictionaries: Oxford and Cambridge.

9.5 Impact of Polarity Inconsistency on Annotation

This experiment aims to show that two inconsistent sentiment dictionaries may give different results when applied to the task of text sentiment annotation. Toward this goal, we use a third-party annotation tool, Opinion Parser [27], the movie reviews dataset **aclImdb** [7] and two inconsistent dictionaries, UF and SWN. **aclImdb** consists of 50,000 reviews. From Table 4, we have that 15 percent of all $(word, pos)$ pair entries in UF have the same polarities in both UF and SWN. Not all these pairs appear in the reviews in **aclImdb**. If we restrict both UF and SWN to their entries that appear in **aclImdb**, we get 20.5 percent agreement between them. Let UF′ and SWN′ be the restrictions of UF and SWN, respectively. UF′ and SWN′ have 6,003 entries. They have the same set of $(word, pos)$ pair entries, the pairs in SWN′ however have the polarities according to SWN.

Since we want some control over the distribution of the set of inconsistent words in reviews, we select from **aclImdb** the reviews with the property that they contain at least 50 words with different polarities in SWN′ and UF′. A review contains a maximum of 136 such words. This gives 516 negative reviews and 567 positive reviews, a total of 1,083 reviews. Opinion Parser is run on this set of reviews using SWN′ and UF′. The tool takes each sentence from a review and splits it into segments and each segment received a polarity tag. For instance, the segments of the sentence "Much scarier in premise than the first, and very entertaining." are "Much scarier in premise", "than the first" and "and very entertaining". The segments received the polarities positive, negative and positive, respectively when UF′ is used. Hence, each sentence can be viewed as a polarity vector, e.g., $(+, -, +)$. The tool is run on a total of 31,701 sentences. If we simply compare the change in the polarity vectors of the sentences when the two dictionaries are used, we get that 16,741 (52.8 percent) sentences have different polarity vectors. We next ignore all those sentences with both positive and negative segments. (Their overall polarities are difficult to judge automatically; in many cases the majority rule fails.) The rest of them are tagged positive (negative) if they contain at least one positive (negative) segment. Otherwise, they are tagged neutral. We obtain that 10,434 (33 percent) such sentences acquire different polarity tags when SWN′ and UF′ are used. These numbers clearly show a strong correlation between polarity inconsistency in sentiment dictionaries and its effect on sentiment tagging in practice. Here is a concrete example. Consider the sentence: "The *premise* is something like a Soprano's episode only not realistic." The only sentiment conveying words are `premise` (assuming it is correct), `not` and `realistic`. `premise` and `realistic` are positive, but `realistic`'s polarity is flipped by `not`. The sentence thus has neutral polarity. If we correct `premise` to be neutral, the sentence becomes negative, which is its correct polarity.

9.6 Computational Issues

The experiments were run on a computer with 12GB of memory and a four-core CPU. For this experiment we use only the disagreement-free union dictionary because it is the largest of all sentiment dictionaries use in this paper. The execution times are the averages over five runs per method. The Boolean formulas constructed with EEM presented the most challenges to the SAT solvers. The SAT solvers required over 10 GB of memory. EEM could not handle words with more than 200,000 MDSes: we left the SAT solver running for a week without ever terminating. EEM however took only 4 hours when we limited the set of words to those that had up to 200,000 MDSes. Words such as `make` and `break` could not be handled with EEM. FSM could handle *all* the words in WordNet and it took about **24 minutes** to finish. It required about 7 GB of memory. The hybrid approach has even more efficient, terminating in about **10 minutes**. The execution performances of FSM and HYBRID are in steep contrast with that of EEM and we recommend them for use in practice. PicoSAT required the least amount of memory: around 2 GB for both FSM and HYBRID. Its computation time was comparable with that of SAT4j in our experiments.

10 Related Work

There are two lines of work on sentiment polarity lexicon induction: corpora- and WordNet-based. Our approach falls into the latter. We cover only this category of work.

WordNet-based approaches use lexical relations defined in WordNet to derive sentiment lexicons. For example, [28] determines sentiments of adjectives in WordNet by measuring the relative distance of a term from exemplars, such as "good" and "bad". The work reports results for adjectives alone. Other approaches use synonyms and antonyms to expand the sets of seeds [29]. Yet another technique is to add all synonyms of a polar word with the same polarity and its antonyms with reverse polarity [17]. It was shown [30] that the method suffers from low recall and is unsuitable in situations when the seed polar words are too few—not uncommon in low resource languages. Moreover, we have encountered instances of antonym pairs where the polarity is not necessarily reversed (e.g., the adjective `advance` has a positive polarity while one of its antonyms, `middle`, has neutral polarity). QW [12] aims to automatically annotate the synsets (senses) in WordNet. It starts from six synsets with known polarities: "positive", "negative", "good", "bad", "inferior" and "superior". These are precisely the synsets that are related to the noun "quality" through the *attribute* relation in WordNet. It navigates WordNet along the semantic relations defined in WordNet (e.g., hypernym, antonym) and assigns polarities to synsets. If two synsets are assigned conflicting polarities they are discarded. QW does not trace down inconsistencies as we do. Also, they do not assign polarities to words. Finally, the relations in WordNet do not have well-defined behavior with respect to preserving/reversing polarity. Recall the above example of the adjectives `advance` and `middle`, which are antonyms, but whose polarities are not reversed.

Unlike SWN, our view is that each synset does not have a degree associated with each polarity. Instead, each synset is 100 percent positive, 100 percent negative or 100 percent neutral.

Machine learning algorithms [31] as well as stochastic algorithms [32] can be employed to classify words into different polarities. According to [29], the performance of [31] is comparable or better than those in [28], [33].

The differences between our approach and earlier ones, including those that are not WordNet-based, are: (1) to our knowledge, none of the earlier works studied the problem of polarity consistency checking for sentiment dictionaries and (2) inconsistencies within individual dictionaries and across dictionaries can be pinpointed by our techniques.

11 Conclusion

We study the problem of checking polarity consistency for sentiment word dictionaries. We prove that this problem is NP-complete. We show that in practice polarity inconsistencies of words both within a dictionary and across dictionaries can be obtained using SAT solvers. Sets of inconsistent words are pinpointed and this allows the dictionaries to be improved. Experiments with five sentiment dictionaries, including the union dictionary, are reported. There are several directions we plan to pursues in the future. First, we plan to categorize the polarity inconsistencies according to our classification (Section 3) and identify the reason behind each inconsistency. Second, as more and more polarity inconsistencies will be "repaired" we will analyze the correlation rate between polarity inconsistency in a dictionary and its effect on the results in sentiment analysis tasks. Third, we would like to expand our polarity consistency framework to other semantic relationships in WordNet (e.g., hypernym/hyponym). Forth, the work on subjective/objective word senses [34] shows that there are synsets in WordNet that seem to conflate subjective and objective senses of a word. The implication of this finding is that some synsets may not have a unique polarity, as we assume in this work. Consequently, we are interested in extending our technique to consider synsets with multiple polarities.

Acknowledgments

This work was supported in part by the following US National Science Foundation (NSF) grants: IIS-0842546, IIS-0842608, CCF-0916438, CNS-1035914, CCF-1319754, and CNS-1314485. The authors thank Bing Liu for running the experiments reported in Section 9.5 on his commercial tool Opinion Parser and Christiane Fellbaum for the invaluable discussions on polarity inconsistency. A portion of this manuscript has appeared in the *Proceedings of the ACL 2012 conference*. Please see the end of Introduction section for more details.

References

[1] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics*, 2004, pp. 271–278.

[2] C. Danescu-N.-M., G. Kossinets, J. Kleinberg, and L. Lee, "How opinions are received by online communities: A case study on amazon.com helpfulness votes," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 141–150.

[3] M. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proc. 20th Int. Conf. Comput. Linguistics*, 2004, pp. 1367–1373.

[4] H. Takamura, T. Inui, and M. Okumura, "Extracting semantic orientations of words using spin model," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics*, 2005, pp. 133–140.

[5] E. Breck, Y. Choi, and C. Cardie, "Identifying expressions of opinion in context," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 2683–2688.

[6] X. Ding and B. Liu, "Resolving object and attribute coreference in opinion mining," in *Proc. 23rd Int. Conf. Comput. Linguistics*, 2010, pp. 268–276.

[7] A. L. Maas, R. E. Daly, P. Pham, D. Huang, A. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics*, 2011, pp. 142–150.

[8] P. Stone, D. Dunphy, M. Smith, and J. Ogilvie, *The General Inquirer: A Computer Approach to Content Analysis*. Cambridge, MA, USA: MIT Press, 1996.

[9] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proc. Conf. Human Language Technol. Empirical Methods Natural Language Process.*, 2005, pp. 347–354.

[10] M. Taboada and J. Grieve, "Analyzing appraisal automatically," in *Proc. AAAI Spring Symp.*, 2004, pp. 158–161.

[11] S. Baccianella, A. Esuli, and F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," presented at the 7th Int. Conf. Language Resources and Evaluation, Valletta, Malta, May 2010.

[12] R. Agerri and A. García-Serrano, "Q-wordnet: Extracting polarity from wordnet senses," presented at the 7th Int. Conf. Language Resources and Evaluation, Valletta, Malta, 2010.

[13] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory Comput.*, 1971, pp. 151–158.

[14] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands: IOS Press, 2009.

[15] E. Dragut, H. Wang, C. Yu, P. Sistla, and W. Meng, "Polarity consistency checking for sentiment dictionaries," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics: Long Papers*, 2012, pp. 997–1005.

[16] J. Han, *Data Mining: Concepts and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2005.

[17] S.-M. Kim and E. Hovy, "Identifying and analyzing judgment opinions," in *Proc. Main Conf. Human Language Technol. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2006, pp. 200–207.

[18] A. Andreevskaia and S. Bergler, "Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses," in *Proc. 11th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2006, pp. 209–216.

[19] L. Bentivogli, P. Forner, B. Magnini, and E. Pianta, "Revising the wordnet domains hierarchy: Semantics, coverage and balancing," in *Proc. Workshop Multilingual Linguistic Resources*, 2004, pp. 101–108.

[20] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Satzilla: Portfolio-based algorithm selection for SAT," *J. Artif. Int. Res.*, vol. 32, pp. 565–606, 2008.

[21] D. Babic, J. Bingham, and A. J. Hu, "B-cubing: New possibilities for efficient sat-solving," *IEEE Trans. Comput.*, vol. 55, no. 11, pp. 1315–1324, Nov. 2006.

[22] P. Jackson and D. Sheridan, "Clause form conversions for boolean circuits," in *Proc. 7th Int. Conf. Theory Appl. Satisfiability Testing*, 2004, pp. 183–198.

[23] V. P. Nelson, H. T. Nagle, B. D. Carroll, and J. D. Irwin, *Digital Logic Circuit Analysis and Design*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.

[24] N. Dershowitz, Z. Hanna, and E. Nadel, "A scalable algorithm for minimal unsatisfiable core extraction," in *Proc. 9th Int. Conf. Theory Appl. Satisfiability Testing*, 2006, pp. 36–41.

[25] D. L. Berre and A. Parrain, "The sat4j library, release 2.2," *J. Satisfiability, Boolean Model. Comput.*, vol. 7, no. 2/3, pp. 59–64, 2010.

[26] A. Biere, "Picosat essentials," *J. Satisability, Boolean Model. Comput.*, vol. 4, no. 2–4, pp. 75–97, 2008.

[27] B. Liu, *Sentiment Analysis and Opinion Mining*. San Rafael, CA, USA: Morgan & Claypool, 2012.

[28] J. Kamps, M. Marx, R. Mokken, and M. de Rijke, "Using wordnet to measure semantic orientation of adjectives," in *Proc. 4th Int. Conf. Language Resources Eval.*, 2004, pp. 1115–1118.

[29] A. Esuli and F. Sebastiani, "Determining term subjectivity and term orientation for opinion mining," in *Proc. 11th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2006, pp. 193–200.

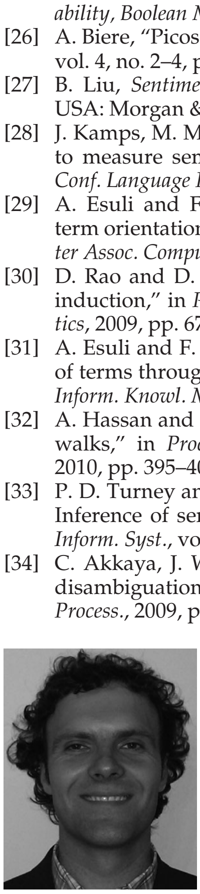
[30] D. Rao and D. Ravichandran, "Semi-supervised polarity lexicon induction," in *Proc. 12th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2009, pp. 675–682.

[31] A. Esuli and F. Sebastiani, "Determining the semantic orientation of terms through gloss classification," in *Proc. 14th ACM Int. Conf. Inform. Knowl. Manage.*, 2005, pp. 617–624.

[32] A. Hassan and D. Radev, "Identifying text polarity using random walks," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 395–403.

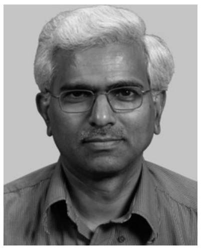
[33] P. D. Turney and M. L. Littman, "Measuring praise and criticism: Inference of semantic orientation from association," *ACM Trans. Inform. Syst.*, vol. 21, pp. 315–346, 2003.

[34] C. Akkaya, J. Wiebe, and R. Mihalcea, "Subjectivity word sense disambiguation," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2009, pp. 190–199.



Eduard C. Dragut received the PhD degree in computer science from the University of Illinois at Chicago in 2010. He is currently an assistant professor in the Department of Computer and Information Sciences, Temple University. His research interests include web-based information retrieval and web database integration. He is a coauthor of the book *Deep Web Query Interface Understanding and Integration*. He co-chaired the VLDB QDB 2012 and ICDE PhD Sympoisum 2014 workshops. He is a member of the IEEE.

Hong Wang is working toward the PhD degree in the Department of Computer Science, University of Illinois at Chicago. He has interned with Google, and with Huawei Technologies as a software engineer. His research interests include web table understanding, information extraction, and natural language processing.



Prasad Sistla received the PhD degree in computer science/applied mathematics from Harvard University in 1983. He is currently a professor in the Department of Computer Science, University of Illinois at Chicago. He has done extensive research in model checking, formal methods, databases, and security. He has published extensively in leading computer science venues. He cochaired and served on the Program Committees of leading Computer Science conferences. His research has been funded by leading organizations such as US National Science Foundation (NSF), AFOSR, and US Defense Advanced Research Projects Agency (DARPA).

Clement Yu received the PhD degree in computer science from Cornell University in 1973. He is a professor in the Department of Computer Science, University of Illinois at Chicago. His areas of interest include search engines and multimedia retrieval. He has published in leading journals such as the *IEEE TKDE, ACM TODS, and Journal of the ACM* and conferences such as VLDB, ACM SIGMOD, ACM SIGIR. He served as the chairman of the ACM SIGIR, the general chair of ACM SIGMOD, and as a member of the advisory committee to the US National Science Foundation (NSF). He was/is a member of the editorial boards of *IEEE TKDE, IJSEKE, Distributed and Parallel Databases, and WWW Journal*.

Weiyi Meng received the PhD degree in computer science from the University of Illinois at Chicago in 1992. He is currently a professor in the Department of Computer Science, State University of New York at Binghamton. His research interests include web-based information retrieval, metasearch engines, and web database integration. He is a coauthor of three books *Principles of Database Query Processing for Advanced Applications*, *Advanced Metasearch Technology*, and *Deep Web Query Interface Understanding and Integration*. He is on the editorial boards of *WWW Journal* and *Frontiers of Computer Science*. He is a member of the IEEE.