Unsupervised Word-Level Affect Analysis and Propagation in a Lexical Knowledge Graph

Mireille Fares[1] . Angela Moufarrej[1] . Eliane Jreij[1] . Joe Tekli[1*]. William Grosky[2]

Abstract. Lexical sentiment analysis (LSA) is of central importance in extracting and analyzing user moods and views on the Web. Most existing LSA approaches have utilized *supervised* learning techniques applied on *corpus-based* statistics, requiring extensive training data, training time, and large statistical corpora which are not always available. Other studies have utilized *unsupervised* and *lexicon-based* approaches to match target words in a lexical knowledge base (KB) with seed words in a *sentiment lexicon*, usually suffering from the limited coverage or inconsistent connectivity of affective concepts. In this paper, we introduce LISA, an unsupervised word-level knowledge graph-based LSA framework. It uses different variants of shortest path graph navigation techniques to compute and propagate affective scores in a lexical-affective graph (LAG), created by connecting a typical lexical KB like WordNet, with a reliable affect KB like WordNet-Affect Hierarchy (where any other lexical or affective KB can be utilized). LISA was designed in two consecutive iterations, producing two main modules: i) LISA 1.0 for affect navigation, and ii) LISA 2.0 for affect propagation and lookup. LISA 1.0 suffered from the semantic connectivity problem shared by some existing lexicon-based methods, and required polynomial execution time. This led to the development of LISA 2.0, which i) processes affective relationships separately from lexical/semantic connections (solving the semantic connectivity problem of LISA 1.0), and ii) produces a *sentiment lexicon* which can be searched in logarithmic time (handling LISA 1.0's efficiency problem). Experimental results on the ANEW dataset show that our approach, namely LISA 2.0, while completely unsupervised, is on a par with existing (semi)supervised solutions, highlighting its quality and potential.

Keywords: Sentiment Analysis, Affect Analysis, Knowledge Base, Graph Navigation, Sentiment Lexicon, ANEW

1. Introduction

Lexical sentiment analysis (or LSA) systems are automated tools which analyze words and text extracts provided by users, and attempt to classify them under different sentiment categories, such as: *positive*, *negative*, or *neutral* emotions. Affect analysis can be viewed as a more fine-grained approach of LSA, which involves more specific classes of affective emotions such as: *happiness*, *sadness*, *surprise*, and *anger*, etc. Methods that perform LSA utilize Natural Language Processing (NLP) and Machine Learning (ML) techniques to automatically identify the underlying emotions carried in the textual data.

LSA methods are of central importance in extracting and analyzing public moods and views in a digital ecosystem, and are becoming increasingly popular in a wide range of Web applications covering: blog sentiment analysis [1, 29, 128] (analyzing bloggers' reviews in web forms regarding certain topics, events, or people), client feedback analysis [35, 76, 114] (automated analysis of customer opinions on purchased products), opinion mining and sentiment analysis on social media [62, 111, 122] (analyzing texts or tweets posted by users on social media outlets, sounding their expressed emotions, suggesting reformulated sentences or emoticons based on the sentiment scores produced), as well as therapeutic and social emotion analysis [32, 79, 91] (e.g., helping autistic children express their emotions through simple texts and the associated affective feedback, e.g., fatigue, frustration, etc.).

Most existing LSA approaches (cf. Background in Section 2) have utilized *supervised* learning techniques applied on *corpus-based* statistics in order to match words or textual patterns with sentiments represented as labeled categories, e.g., [41, 59, 65]. They usually require extensive training data, training time, and large statistical corpora which are not always available and require significant manual effort. In addition, most methods usually produce discrete sentiment labels (e.g., *joy*, *surprise*) without however evaluating sentiment intensity (valence) scores (e.g., 20% *joy*, 35% *surprise*). On the other hand, other studies have utilized *unsupervised* and *lexicon-based* approaches, e.g., [40, 117, 130], in order to match target words with seed words in a *sentiment lexicon* (e.g., LEW list [37], or WNA list [115]), by evaluating their semantic similarity or distance in a reference lexical knowledge base (KB, e.g.,

Mireille Fares mireille.fares@lau.edu	Angela Moufarrej angela.moufarrej@lau.edu	Eliane Jreij Eliane.jreij@lau.edu	Joe Tekli Joe.tekli @lau.edu.lb	William Grosky wgrosky@umich.edu

[1] Department of Electrical and Computer Engineering, Lebanese American University (LAU), Byblos, Lebanon

* This study was partly conducted during the author's Fulbright Visiting Scholar research mission in the Computer and Information Science department, University of Michigan (UMich), Dearborn, USA

[2] Department of Computer and Information Science, University of Michigan (UMich), Dearborn, USA

WordNet [73]). The latter usually suffer from the limited coverage of manually created sentiment lexicons, as well as the limited or inconsistent connectivity of affective concepts in the lexical KB (cf. Section 2.5). Recent efforts have focused on the automatic creation of sentiment corpora, e.g., [6, 15, 89], in order to address some of the above limitations. Yet most rely on (semi)supervised processes for their construction, thus sharing the limitations of supervised method mentioned above.

In this study, we introduce UWKG_LISA (or LISA for short), a framework for Unsupervised Word-level Knowledge Graph-based Lexical Sentiment Analysis. In contrast with most existing supervised or corpus-based approaches, we provide an unsupervised knowledge-based solution which utilizes graph navigation techniques applied on a lexical-affective graph (LAG), in order to infer word affect scores while avoiding the training data and training time bottlenecks. The LAG is created by connecting a typical lexical KB graph like WordNet, with a reliable affect KB like WordNet-Affect Hierarchy (WNAH) [103] (although any other lexical or affective KB sharing similar properties can be utilized).

LISA was designed in two consecutive iterations, producing two main modules: i) LISA 1.0 for affect navigation, and ii) LISA 2.0 for affect propagation and lookup. LISA 1.0 accepts as input a set of user input words (e.g., extracted from a sentence) and a set of target affect categories in a LAG, and produces as output the target affect scores (intensity weights) for every input word located in the LAG. It consists of two main components: i) *linguistic pre-processing*, to process input words, identifying their proper word concepts (synsets) in the lexical KB graph (e.g., WordNet), and ii) *Max_Affect* which navigates the LAG from the input word concepts to the target affect categories, using an adaptation of the shortest path problem in order to identify the maximum word affect scores. Yet, preliminary experiments and a careful analysis of LISA 1.0 highlighted two main issues regarding the module's effectiveness and efficiency. On the one hand, we realized that the semantic connectivity between affect concepts in the LAG does not always accurately portray their affective expressiveness (e.g., concepts *good* and *bad* are only three hops away from each other in WordNet, despite their opposing sentiments, cf. Fig. 1 in Section 2.5), which reduced the module's accuracy in computing affect scores. On the other hand, LISA 1.0's main *Max_Affect* process requires average polynomial (quadratic) complexity in the size of the LAG, which, despite LAG navigation optimizations and parallelization, remained relatively time consuming.

This led us to improve our design by producing LISA 2.0, which first i) propagates the sentiment scores over all connected word concepts in the LAG, from a set of user target affect categories (target emotions), and then ii) allows fast lookups of the computed word affect scores to perform LSA. It encompasses three main components: i) *WNAH_Propagation* which propagates the affect score of every affect category in WNAH to all other affect categories in WNAH, considering affective connections only, such that each category becomes fully representative of all of the others (solving the LAG semantic connectivity problem of LISA 1.0), ii) *Back_Propagation* which backward propagates the affect scores, from user chosen affect categories (pre-processed by *WNAH_Propagation*) to all connected concepts in the LAG, and iii) *Affect_Lookup* which performs fast search/lookup operations over the affect-scored concepts of the LAG (requiring average logarithmic time, thus alleviating the polynomial complexity problem of LISA 1.0).

We have implemented LISA 1.0 and 2.0 to test and evaluate their performance. Experimental results on the Affective Norms for English Words (ANEW) dataset [10, 100] show that our approach, namely LISA 2.0, while completely unsupervised, is on a par with existing (semi)supervised solutions, highlighting its quality and potential.

The remainder of the paper is organized as follows. Section 2 provides background information and briefly reviews the literature on LSA techniques. Our LISA framework is developed in Section 3. Section 4 presents experimental results, while Section 5 concludes with future directions.

2. Background and Related Works

Lexical sentiment analysis (LSA) is concerned with the analysis of text containing or reflecting sentiments [93, 104]. It is usually viewed as a sentiment classification or opinion mining activity concerned with determining the overall sentiment orientation of the elements (e.g., words, phrases) within a text [124]. In the following sub-sections, we provide a brief overview of LSA approaches, covering: i) sentiment categories; ii) granularities, iii) features, iv) resources, v) techniques, and vi) sentiment lexicon creation.

2.1. Sentiment Categories

Researchers in LSA usually distinguish between two kinds of sentiments: i) opinions/polarity such as *like*/*dislike*, generally referred to as *positive*/*negative* opinions, and ii) emotions/feelings such as *happy*/*angry*/*afraid*/etc., generally referred to as affect categories [52]. Accordingly, LSA methods can be distinguished as: i) *opinion detection* (or *opinion mining*) methods [49], and ii) *affect analysis* methods [104]. Here, two main differences can be identified [1]. On the one hand, opinion detection usually involves two opinion categories: *positive* and *negative*, and is sometimes extended to include an additional *neutral* category [25, 124]; whereas affect analysis usually involves a larger number of affect classes, ranging from a reduced set of six basic emotions in [84] (i.e., *anger*, *fear*, *joy*, *love*, *sadness*, and *surprise*) to a more comprehensive hierarchy of 294 sentiment categories introduced in WNAH [103] (cf. Section 3.1.2). On the other hand, opinions associated with text segments are usually mutually exclusive, i.e., a word or a phrase is either *positive* or *negative* (or *neutral* when considered); whereas a text segment might contain multiple affect categories [46, 104]. For example, verb "alarm" simultaneously reflects *fear*, *warning*, and *excitement* [104], whereas adjective "thrilled" reflects *happiness* and *excitement* [46]. More recently, researchers have emphasized the need to identify – not only the sentiment category – but also the *intensity* of the sentiment reflected by the text, i.e., the sentiment's valence or strength (defined as a varying numerical range) [22, 47], allowing to perform more accurate and finer-scale LSA [1, 43] (e.g., word "trust" is 79% *positive* and 29% *negative*, and contains 82% *happiness*, 31% *anger*, and 31% *sadness* among other affects following the ANEW dataset [100]).

In our current study: i) we focus on the more inclusive task of affect analysis (including both opinion and affect categories), while ii) evaluating affect intensity (valence) accordingly.

2.2. Granularity of LSA

Sentiments can be extracted from text at different granularity levels: i) word, ii) phrase, iii) sentence, iv) document, and v) aspect. *Word-level* LSA is the most pinpointed/fine-grained approach to LSA, where individual words are associated with sentiment categories [1, 93], which are then utilized to allow LSA at higher granularities of text. Words can be processed: i) separately (i.e., standalone) [112, 115], or ii) in-context, considering their textual surroundings [1, 94], the domain or topic at hand [8, 27], or the author's perspective [30, 39]. For instance, the term "unpredictable" may have a negative orientation in an automotive review (e.g., "unpredictable steering"), but it could have a positive orientation in a movie review (e.g., "unpredictable plot") [112]. *Phrase-level* SA consists of associating sentiments with individual phrases, where a phrase designates an expression usually made of a couple of words occurring close to each other in a text (e.g., "unpredictable steering"). Phrase sentiments are generally deduced from word-level sentiments [124, 129]. *Sentence-level* LSA consists in associating sentiments with individual sentences, based on word-level or phrase-level LSA [5, 38]. Similarly, *document-level* LSA consists in associating sentiments with individual documents (e.g., product reviews, blogs, news articles, etc.), where document sentiments are usually deduced from constituent sentences. A major difference between word/phrase-level LSA on the one hand, and sentence/document-level LSA on the other hand, is that a word/phrase usually reflects a single opinion (e.g., *positive* or *negative*) and/or related emotions (e.g., *happy* and *excited*) [112, 115], whereas a sentence/document might contain opposing opinions (e.g., containing words/phrases with *positive* opinions, and others with *negative* opinions) and/or un-related emotions (e.g., *happy*, *angry*, and *relaxed*) [25, 124]. *Aspect-level* LSA consists in extracting the main aspects of a certain text (e.g., sentence or document) where aspects represent the interesting features mentioned in the text describing what the text is about (e.g., "battery", "processor", "touch screen" could be aspects describing *mobile phones*), and then estimating the sentiment scores of the text per aspect (e.g., how positive or negative the opinions are on average for every aspect) [3, 96]. Methods of this category need to address the added challenge of performing aspect identification as a prerequisite to performing LSA [86, 122].

In our current study, we focus on word-level LSA where words are considered separately or in-context, depending on the availability of surrounding words in the provided user input.

2.3. Features for LSA

Different features can be utilized to perform word-level LSA, ranging over: i) lexical form, ii) semantic meaning, and iii) part-of-speech tag. Words targeted for LSA are usually matched against a set of seed words with manually or automatically associated sentiments, in order to acquire/inherit the corresponding sentiment categories. Matching target words with seed words can be performed syntactically (i.e., exact matching), and can take into account other features. For instance, the *lexical form* feature allows matching target words with seed words based on their base forms by stemming morphological variants (e.g., "laughing" is stemmed and matched with "laugh") [43, 48]. The *part-of-speech* (POS) feature allows distinguishing between nouns, verbs, adjectives, and adverbs which might carry slightly different sentiment clues (e.g., adjective "funny" might have a lesser positive polarity than noun "fun") [55, 119]. The *semantic meaning* feature, as its name suggests, allows matching words based on their meanings, by comparing their semantic definitions and relationships w.r.t. a lexical KB like WordNet [15, 59] (e.g., word "mirth" is associated with concept *hilarity* in WordNet which means "great merriment and cheerfulness", and thus can be matched with any of the concept's synonymous terms, e.g., "playfulness", "gaiety", or "merriment", if they appear in the set of seed words).

Other features to perform phrase, sentence, or document-level LSA include: n-gram (word associations) [1, 78], syntactic structure (parse tree) [124, 132], valence shifters (intensifiers and modal operators, like "really", "could" and "should") [48, 132], and statistical features (e.g., contextual and co-occurrence frequencies) [66, 96].

In our current study, we target word-level LSA and thus focus on word-level features[3].

2.4. Resources for LSA

External resources are essential to perform LSA, providing reference data which is needed to associate sentiments with text. In this context, LSA methods can be distinguished as: i) corpus-based or ii) lexicon-based. The *corpus-based* approach, e.g., [66, 120, 124], is data-driven, as it relies on processing large text corpora (such as OpenMind [99] and ISEAR [95]) to identify the probability of occurrence of textual features (e.g., lexical forms, POS tags, n-grams, or phrasal patterns), in order to enable sentiment predictions for new texts. The *lexicon-based* approach, e.g., [15, 60, 105], is knowledge-driven, as it relies on acquiring sentiment clues from a readily available sentiment lexicon, i.e., a large collection of words or concepts (i.e., word senses) associated with sentiment categories (or intensity scores). Machine readable lexicons such as SentiWordNet [6], WNA [115], and SenticNet [16] are few of the most widely used sentiment lexicons in the literature. While corpus-based methods have been popular in the past few years [42, 63], yet they are generally data hungry and require extensive training, huge textual corpora, and a considerable amount of manual effort to produce a relevant sense-annotated corpus, which are not always available or feasible in practice. Therefore, lexicon-based methods have been receiving a lot of attention lately [63, 93].

Yet, lexicon-based LSA methods suffer in turn from two major limitations: i) ambiguity and ii) limited coverage [24, 82]. On the one hand, many widely used sentiment lexicons (such as General Inquirer [101] and LIWC [85]) associate sentiments with words instead of concepts (i.e., word meanings), and thus do not distinguish between the different meanings of the same word which might have – each – a different sentiment bearing. On the other hand, the limited coverage of manually created lexicons (such as the LEW list [37] and the core WNA[4] list [115]) is another major concern, due to the substantial effort in manually annotating terms or concepts [24]. To address the above limitations, various studies have been recently developed to automatically expand/combine sentiment lexicons with other information or common knowledge sources (such as Wikipedia), e.g., [19, 89, 93], promoting an increasingly distinguished line of LSA research: *Automatic Sentiment Lexicon Creation*, e.g., [4, 93, 97]. We further describe the latter in Section 2.6.

In our current study, we focus on lexicon-based LSA, and address both: i) the ambiguity problem by using unambiguous word meanings (concepts) to perform LSA, and ii) the limited coverage problem by processing a full-fledge LAG connecting a comprehensive affect KB (WNAH) with an expressive lexical KB (WordNet)[5].

2.5. Sentiment Analysis Techniques

LSA is usually viewed as a text-sentiment classification task. In this regard, existing approaches can be roughly categorized as: i) supervised, or ii) unsupervised.

2.5.1. Supervised Sentiment Analysis

Supervised methods, e.g., [23, 53, 54], involve the use of supervised-learning techniques, using samples (a human expert manually annotates sample words/phrases with the intended sentiment in context) provided as training data for a learning algorithm that induces rules to be used for assigning sentiments with other occurrences of the word/phrase. External knowledge (mainly corpus-based) is used and combined with the human expert's own knowledge of word/phrase sentiments when manually annotating the training examples. In this context, different kinds of classifiers have been used, including Support Vector Machines (SVM) [20, 74], Naïve Bayes (NB) [58, 123], Maximum Entropy (ME) [69, 92], and Linear Regression [48, 127]. A few semi-supervised (or hybrid) approaches have also been developed, including Regularized Least Squares (RLS) classification [98], feature relation networks (FRN) [2], as well as different combinations of hybrid classifiers including induction rule-based, statistics-based, NB-based, and SVM-based [7, 90, 126].

While effective, (semi)supervised methods suffer from several disadvantages. First, they include a learning phase which is time-consuming and subject to over-fitting[6], depending on the quality, size, and domain of the

[3] We are currently investigating phrase-level and sentence-level features, namely word associations, n-gram repetitions, and valence shifters, combined with an emoji lexicon [40] in order to perform LSA on short social media texts like Facebook comments and Twitter messages.

[4] The WordNet-Affect Hierarchy (WNAH) [103] is different from the WordNet-Affect list (WNA) [109]. The latter is a list of words manually annotated following a set of affect categories embedded in WordNet, whereas the former designates an independent structure consisting of a hierarchical (*hypernymy*/*hyponymy*) organization of affect categories defined separately from WordNet (cf. Section 3.1).

[5] Other lexical and affective KBs sharing similar properties could also be used.

training data set which is not always available. Another shortcoming is that legacy supervised classifiers can only deal with discrete class labels (e.g., *positive*, *calm*, etc.), whereas sentiment intensity (valence) can vary along a continuum (e.g., 80% *positive*, 20% *calm*, etc.). Some works have attempted to address this problem by using regression-based classifiers [48, 75, 83]. For instance, the authors in [48] apply log-linear regression, trained on conjunctions and morphological features, to classify adjectives while producing corresponding opinion intensity scores. In [83], the authors assume that the words come from a discretization of a continuous function that maps the feature space to a metric space, and use regression to find the hyperplane that best fits the training data. A similar approach is developed in [75] where the authors use word n-grams to train a pace regression classifier [125] for assigning affect intensities to blogs. Nevertheless, regression-based learning methods have seen limited usage in LSA [1], in comparison with more popular SVM and NB methods, e.g., [20, 74, 123]. A third shortcoming is that supervised methods train their classifiers to recognize different classes separately, as if the produced categories are totally unrelated, e.g., [20, 74]. Yet, certain sentiment classes may be related [104] (cf. Section). For instance, *hate* and *anger* are related affects and usually co-occur together. In other words, a textual token reflecting *hate* would most probably reflect some *anger*, and vice versa. To our knowledge, the latter problem has only been explicitly addressed in [1], where the authors developed an ensemble classifier (i.e., grouping multiple classifiers) based on Support Vector Regression (SVR) [121]: a hybrid technique combining SVM with regression-based classification, in order to predict affect intensity scores while considering the correlation between different affect categories. Results in [1] showed improved sentiment prediction quality over legacy SVM methods.

Our current study handles the above mentioned aspects, by: i) identifying multiple sentiment categories for every input word, ii) assigning different intensity scores describing the word's expressiveness in every category (e.g., 25% happiness, 10% fear, 15% surprise, etc.), and iii) considering affect category relationships and their relative expressiveness w.r.t. each other, which was shown crucial to improving LSA performance (cf. Section 4).

2.5.2. Unsupervised Sentiment Analysis

Unsupervised methods, e.g., [40, 117, 130], are usually fully automated and do not require human intervention or a training phase. Most approaches in this category make use of a machine-readable sentiment lexicon (e.g., SentiWordNet [6] or WNA [115]) usually represented as a set of words/expressions or concepts with their sentiment categories or intensity scores. Given a target text to be processed, unsupervised LSA consists in assigning each constituent textual token (e.g., word or phrase) and consequently the whole target text, with a sentiment score. The score is a measurement of the intensity of the token w.r.t. to one (or many) sentiment category(ies). Once the score for every token is obtained, the score for the whole text can be calculated by applying aggregation functions (e.g., average, maximum, or linear combination). Clearly, the core step in this family of techniques is the way to score textual tokens. In this context, scoring methods can be distinguished as: i) statistical, or ii) semantic.

Statistical scoring methods evaluate word average sentiment intensities across the lexicon's items occurring in a text [67, 104]. The authors in [26, 113] assess the intensity of each word based on its co-occurrence frequency with a set of core words reflective of a given affect [45]. The occurrence frequencies for the core words and candidate words are derived from search engines such as AltaVista [45] or Yahoo [74]. For example, the core words for the *praise* affect would include "acclaim", "praise", "congratulations", "homage", and "approval" [45]. The approach in [26, 113] is coupled with a point-wise mutual information (PMI) scoring mechanism for assigning candidate word intensity scores. Traditional PMI assigns each word a score based on how often it occurs in proximity with *positive* and *negative* paradigm words, and has been adapted to affect categories [45]. Sentence level averaging is then performed based on the generated word-level PMI scores [34, 113]. The authors in [81] perform subject favorability determination by evaluating the syntactic dependencies among the phrases and subject term modifiers. A similar approach in [80] proposes a hybrid model combining PMI [112] with syntactic similarity features [81], and focuses on adjectives while disregarding other part-of-speech groups. The main limitation of this group of methods is the need for a large and expressive textual corpus on which to perform statistical analysis, which is not always available.

Semantic scoring is based on the premise that semantically close words share similar sentiment bearings [93], and consists in evaluating the semantic distance between the meanings of words in a reference KB. Most semantic scoring LSA methods, e.g., [24, 57, 61], utilize WordNet [73] as a widely used lexical KB made of a set of word concepts (synsets) and their semantic relationships (e.g., *synonymy*, *hyponymy*, etc., [18, 106], cf. Section 3.1.1). In this context, the authors in [61] expand the seed words associated with an affect category by comparing each

[6] Overtraining the supervised algorithm to adjust to specific features of the training data, where it becomes exceedingly effective in categorizing the trained data, yet not so effective in generalizing to classify new (untrained) data due the model excessive complexity/specificity [125].

candidate word and its synonymous terms with the seed word list [74]. The intensity for a candidate word is proportional to the number of times the word and its synonyms appears in the seed list [61]:

$$score(w_i, cat_j) = \frac{count(cat_j)}{count(\text{words})} \times \sum_{k=1}^{n} count(syn_k^i, cat_j) \tag{1}$$

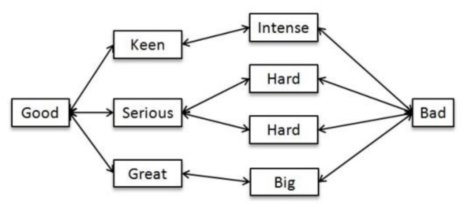
where w_i is a candidate word, cat_j is an affect category, syn_k^i is one of the n synonyms of word w_i, $count(cat_j)$ is the number of words in cat_j divided by the total number of words considered. In [57], the authors identify the polarity of an input (source) word by measuring its distance from two reference (destination) concepts: *good* and *bad* in the WordNet graph. Distance is evaluated by counting the number of *synonymy* relationships (links) connecting the source word concept with either one of the destination concepts, such as (normalized) distances from concepts *good/bad* designate *positive/negative* intensity scores respectively:

$$score(w_i) = \frac{dist(w_i, \text{bad}) - dist(w_i, \text{good})}{dist(\text{good}, \text{bad})} \tag{2}$$

Similar approaches were introduced in [24, 44, 68, 102, 115], which consider a set of seed concepts (instead of two concepts only: *good* and *bad*) as references for their distance computations. The authors also extend distance evaluation to consider other WordNet relations which can carry sentiments such as *hypernymy*/*hyponymy*, *derivation*, and *pertainym* [24, 102, 115].

Note that applying the semantic scoring LSA approach requires *word sense disambiguation* (WSD) [107, 108], a computationally expensive pre-processing step to assign the word targeted for LSA with its semantic concept (i.e., identifying the meaning of the word in its context) [70], so that the latter concept can then be processed for semantic scoring. Another common pitfall of this category of methods is the semantic connectivity between reference concepts, which might not always be accurate. For instance, one can traverse the WordNet graph from concept *good* to concept *bad* in only three hops using the *synonymy* relationship (cf. Fig. 1). This seems “weird” since *good* and *bad* are opposing sentiments, and thus one tends to think they should be farther away from each other in the KB graph. This problem is shared among other lexical knowledge references such as ConceptNet [110] and Yago [51], where concepts are defined following their lexical meanings, rather than their affective expressiveness. In this context, there is a crucial need to distinguish between lexical and semantic relationships between concepts in the lexical knowledge graph on the one hand, and affective relationships between affect categories on the other hand.

We address the latter in our approach by: i) introducing a heuristic WSD method (to reduce processing time), and ii) utilizing a separate structure to describe affective relationships: WNAH [103], a self-contained and independent hierarchy of affect categories, which we use to navigate between affective categories.



- Nodes represent words, and node links represent *synonymy* relationships.
- *Synonymy* relationships are bidirectional and thus are represented as edges having opposite directions.

Fig. 1. Extract of *synonymy* relationship connectivity between words *good* and *bad* in WordNet [44]

2.6. Sentiment Lexicon Creation

Sentiment lexicon creation is attracting increasing attention, starting from an initial list of seed words with user-defined sentiments, and making use of (semi)supervised corpus-based or lexicon-based techniques to extend the seed list. Early approaches like SentiSense [24] and Emolex [77] were developed via crowd-sourcing, relying on the wisdom of a crowd of users providing manual sentiment annotations (using Amazon’s Mechanical Turk[7]). SentiWordNet [6] is a semi-supervised lexical resource associating word concepts from WordNet with *positive*, *negative*, and *objective* scores ranging from 0.0 to 1.0. The authors evaluate the polarity of concepts by mining their glosses, considering that the glosses of *positive* (*negative*) concepts would contain terms that mostly belong to other

[7] Mechanical Turk: https://www.mturk.com/mturk/welcome

positive (*negative*) concepts. The authors utilize this binary relation between concepts (gloss of concept c_i contains a term belonging to c_k) to build a directed graph between concepts. A random-walk process [33] is then executed on the graph to disambiguate the glosses' words and associate them with existing WordNet concepts. Consequently, starting from two small seed sets of manually chosen *positive* and *negative* concepts with their disambiguated glosses, the authors train a set of ternary classifiers to recognize concept polarity based on their glosses. Another polarity lexicon is SenticNet [16] which integrates several knowledge bases, namely ConceptNet [110], DBPedia [9], and WordNet [73], to provide common-sense LSA capability. Initial versions of lexicon: SenticNet 1 [17], 2 [13], and 3 [15], i) integrate different pieces of knowledge from the reference knowledge bases, translate them into RDF (Resource Description Framework) triples (e.g., *Pablo Picasso-IsA-Artist*), and insert them into a graph through the energy-based knowledge representation (EBKR) formalism [12], ii) utilize semi-supervised learning to evaluate the affective associations between concepts, by iii) plotting them into a multi-dimensional vector space using AffectiveSpace [11]. More recent versions: SenticNet 4 [15] and 5 [16], i) apply principal component analysis to perform dimension reduction on the vector space, ii) use *k*-nearest neighbor with *k*-medoids classification to determine semantically related concepts that correspond to the same polarity, and iii) determine sentiment intensity levels by using specially designed discrete and continuous neural networks (DNNs and CNNs) allowing multi-dimensional scaling [13, 14, 87]. A commonsense ontology, OntoSenticNet, was recently built on top of SenticNet, allowing to associate concepts, word embeddings, domain information, and external resources with sentiment values, including the definition of a formal conceptual hierarchy and its properties [28].

AffectiveSpace [11] was introduced in as an n-dimensional vector space produced by matching concepts from ConceptNet [110] with those of WNA [115]. The authors applied singular value decomposition on the resulting concept space in order to select the 100 principal components representing common sense concepts and their emotions. SenticNet was later extended in [88] to produce EmoSenticNet, by integrating SenticNet with AffectiveSpace. The approach assigned one of the six emotion labels of WNA (i.e., *anger*, *fear*, *disgust*, *sadness*, *surprise*, and *joy*) to each concept of SenticNet, along with their intensity scores. The latter was conducted in two stages. The first stage applied fuzzy *c*-means clustering (FCM) in 16 ISEAR data columns [95] using WordNet similarity measures and co-occurrence frequencies in ISEAR to obtain membership values for every concept w.r.t. the six emotion categories. In the second stage, membership values belonging to the six emotion categories were evaluated using supervised SVM-based classification.

2.7. Discussion

To wrap up, we highlight the main issues and limitations facing existing LSA methods. On the one hand, *supervised* and *corpus-based* methods match words or textual features with sentiments represented as labeled categories, using machine learning techniques applied on text corpus statistics. They usually require extensive training data, training time, manual effort, and large statistical corpora which are not always available or practical. In addition, most methods (except for a few approaches in [48, 75, 83]) usually produce discrete sentiment labels (e.g., *joy*, *surprise*) without however evaluating the intensity (valence) scores of different sentiments and their relationships in describing the target word or text (e.g., 20% *joy*, 35% *surprise*). On the other hand, *unsupervised* and *lexicon-based* approaches match source words with seed words in a reference sentiment lexicon, by evaluating their semantic distance (w.r.t. corresponding word concept glosses, or the number of semantic links separating them) in a reference lexical KB (e.g., WordNet, ConceptNet). They usually suffer from the limited coverage of manually created sentiment lexicons (e.g., LEW list [37], or WNA list [115]) as well as the limited or inconsistent connectivity of affective concepts in the lexical KB (i.e., concepts *good* and *bad* are only three hops away in WordNet, despite describing opposing sentiments). Recent efforts have focused on the automatic creation of sentiment corpora in order to address the above limitations. Yet most rely on (semi)supervised processes for their construction, thus sharing the latter's limitations mentioned above.

3. LISA Framework

To address most of the limitations above, we introduce UWKG-LISA (or LISA for short), an Unsupervised Word-level Knowledge Graph-based Lexical Sentiment Analysis framework. It uses different variants of shortest path graph navigation techniques to compute and propagate affective scores in a LAG. LISA's overall architecture is depicted in Fig. 2. It is designed in two separate yet interconnected modules: LISA 1.0 for affect navigation, and LISA 2.0 for affect propagation and lookup. We develop LISA's modules in the following subsections.

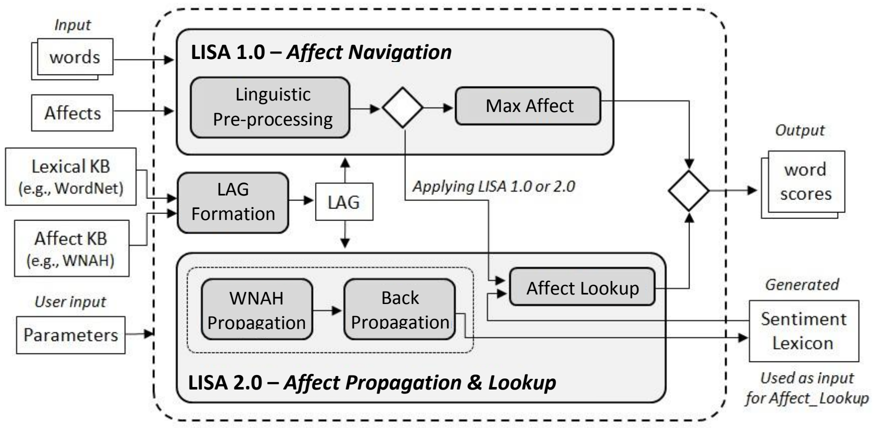


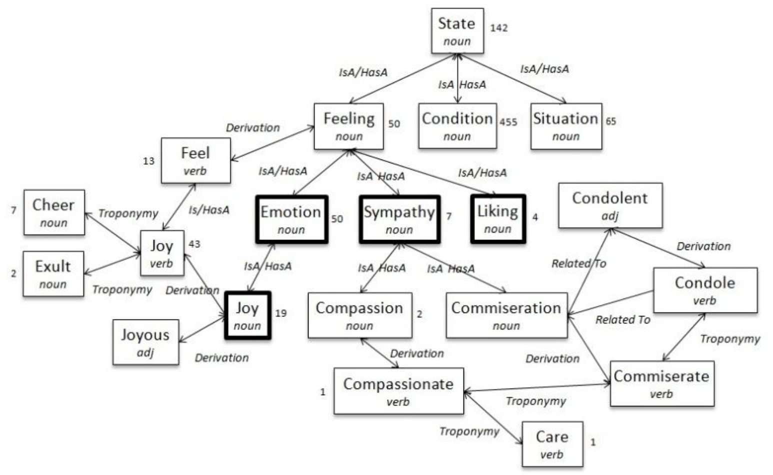
Fig. 2. Simplified activity diagram describing LISA's overall architecture

3.1. Lexical Affective Graph (LAG)

The LAG is created by connecting a typical lexical KB graph like WordNet [73], with a reliable affect reference like WordNet-Affect Hierarchy (WNAH) [103] (although any other lexical or affective references sharing similar properties can be utilized). WNAH is a commonly used affect knowledge base consisting of 294 different affect categories (e.g., *positive emotion*, *joy*, *love*, *apathy*, *euphoria*, etc.), hierarchically organized following a hypernymy/hyponymy (*IsA*/*HasA*) inheritance structure, where every affect category matches a lexical concept (synset) in WordNet. We briefly describe our usage of WordNet and WNAH in the following sub-sections.

3.1.1. WordNet Lexical Knowledge Graph

WordNet [73] it is a widely known machine-readable lexical KB that can be easily represented and processed as a graph made of a set of concepts (nodes) representing word senses (i.e., synsets), and a set of labeled links (edges) connecting the concepts, representing semantic relationships (*hypernymy*, *meronymy*, etc., cf. Fig. 3). A concept encompasses words sharing the same meaning (synonyms) as well as gloss definition (e.g., concept *car* encompasses words "*car*", "*auto*", "*automobile*", etc., and is described by gloss "*a motor vehicle with four wheels propelled by an internal combustion engine*"). Concepts are identified using unique IDs (to distinguish between polysemous terms) and are organized in four separate and interconnected groups, to describe: nouns, verbs, adjectives, and adverbs.



- Numbers next to concepts represent concept frequencies (computed based on the Brown corpus [36]).
- Since all semantic and lexical relations considered in our study are symmetrical (*hyponymy*/*hypernymy*) or bidirectional (e.g., *related to*, *derivation*, *troponymy*), we represent them as edges having opposite directions.
- Concepts highlighted with thick contours are connected to WNAH, and are mapped to the affect categories[8] highlighted in Fig. 4.

Fig. 3. Extract of the WordNet lexical KB graph

[8] Note that *compassion* and *commiseration* are also connected to WNAH. Nonetheless, we do not highlight them here since they do not appear in the WNAH extract in Fig. 4, which is utilized in our computation examples later-on.

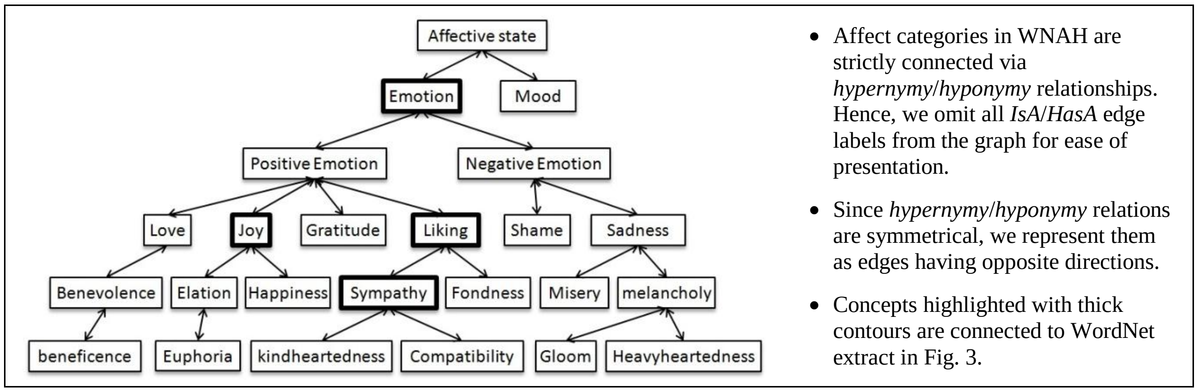
WordNet has more than 18 different semantic relationships connecting concepts. Yet, in our current study, we only consider the relations that were shown to carry sentiments or emotions [102, 115], i.e., for noun concepts: *hypernymy* (*HasA*), *hyponymy* (*IsA*), *instance hypernymy*, *instance hyponymy*, and *usage*; for verb concepts: *hypernymy*, *entailment*, *verb group*, and *troponymy*; for adjective concepts: *attribute* (values that describe the current concept), *similar to*, and *related to*, and for adverb concepts: *usage*.

In addition to semantic relationships, WordNet contains lexical relationships between words, and that can also carry sentiments or emotions, including: *pertainymy* (a lexical relationship that allows deriving the adjective of a noun), and *derivation* (a lexical relationship that allows to derive any word type, if it exists) [72, 73]. The previous mentioned relationships are used in LISA to navigate the LAG, in order to compute word concept affect scores w.r.t. target affect categories (cf. LISA 1.0 in Section 3.2), or back-propagate the latter's scores throughout the LAG (cf. LISA 2.0 in Section 3.3).

Note that in our current study, we utilize WordNet as the lexical reference to describe word meanings, since it easily connects to WNAH via their matching word concept/affect category pairs (cf. Fig. 3 and Fig. 4). Yet, our approach is general and can be adapted to any other lexical KB graphs such as Yago [51] or ConceptNet [110], as long as at least one of their lexical concepts matches one of the affect categories in WNAH.

3.1.2. *WordNet Affect Hierarchy (WNAH)*

WNAH consists of a set of concepts representing affective categories (e.g., *joy*, *elation*, *euphoria*, *positive emotion*, etc.), hierarchically organized following a *hypernymy*/*hyponymy* (*IsA*/*HasA*) structure highlighting the relationships between the categories (e.g., *euphoria-IsA-elation*, *elation-IsA-joy*, and *joy-IsA-positive emotion*, cf. Fig. 4). Most categories in WNAH are mapped to corresponding word concepts in WordNet (e.g., *love* in WNAH is mapped to the concept describing the first meaning of *joy* in Wordnet 3.0, encompassing synonymous terms, "*joy*", "*joyousness*", and, "*joyfulness*", and described as "*the emotion of great happiness*"), except for certain composite affect categories (like *self-pride*, *general-dislike*, and *neutral-languor*) which are not connected to any WordNet concept (synset). Fig. 4 illustrates an extract of WNAH, highlighting the affect categories that match lexical concepts in the WordNet extract in Fig. 3.



- Affect categories in WNAH are strictly connected via *hypernymy*/*hyponymy* relationships. Hence, we omit all *IsA*/*HasA* edge labels from the graph for ease of presentation.
- Since *hypernymy*/*hyponymy* relations are symmetrical, we represent them as edges having opposite directions.
- Concepts highlighted with thick contours are connected to WordNet extract in Fig. 3.

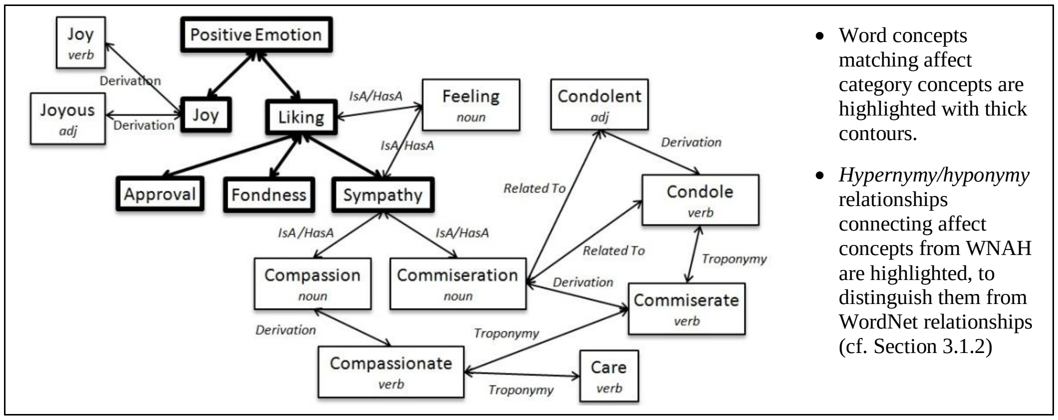
Fig. 4. Extract of WNAH

3.1.3. *Sample LAG Structure*

Consequently, the LAG comes down to a lexical KB graph (e.g., WordNet) and an affect hierarchy (WNAH) connected together and processed as one single graph structure. As sample LAG extract, based on the WordNet and WNAH extracts in Fig. 3 and Fig. 4 respectively, is shown in Fig. 5.

3.2. LISA 1.0 – Affect Navigation module

The LISA 1.0 affect navigation module accepts as input a set of user words (provided separately, or extracted from an input sentence provided by the user) and a set of target affect categories in a LAG, and produces as output the target affect scores (intensity weights) for every input word located in the LAG. It consists of two main components: i) *linguistic pre-processing*, to process input words, identifying their proper word meanings (concepts) in the lexical KB (e.g., WordNet), i.e., locating the concepts (nodes) in the LAG, and ii) *Max_Affect* which navigates the LAG from the input word concepts to the target affect categories, using an adaptation of the shortest path problem to compute word affect weights. We describe both components in the following sub-sections.



- Word concepts matching affect category concepts are highlighted with thick contours.
- *Hypernymy/hyponymy* relationships connecting affect concepts from WNAH are highlighted, to distinguish them from WordNet relationships (cf. Section 3.1.2)

Fig. 5. Sample LAG based on a mapping of WordNet and WNAH extracts in Fig. 3 and Fig. 4 respectively

3.2.1. Linguistic Pre-Processing component

Linguistic pre-processing consists of four main phases: i) *tokenization* (identifying the separate tokens in an input sentence), ii) *stop word removal* (removing prepositions and semantically meaningless words such as: *the*, *a*, *of*, *to*, etc.), iii) *stemming* (reducing inflected or derived words to their stem, i.e., base or root, e.g., *games* → *game*, *laughed* → *laugh*). For instance, considering user input sentence: "The man's words were full of dryness that they gave us all the creeps", the stemmer would produce as output the following sequence of words: "*man word be dryness give creeps*". The latter will then be processed for iv) *word sense disambiguation* (WSD), identifying the proper word meanings (concepts) in the lexical KB (in our case, WordNet). Once located in the LAG, word concepts are then provided as input to *Max_Affect* in order to compute their affective weights.

Note that our current study handles sentences as sequences of stemmed words, where the context of every word consisting of the words surrounding it in the sentence, is used as the main feature for semantic disambiguation, and subsequently for sentiment analysis (cf. Background Section 2.3).

```
Algorithm: WSD_Heuristic
Input: Target word: w          // target for WSD
       Context words: C[]      // context of w, consisting of a sequence of words
       Flag: LESK              // user preference regarding using simplified LESK, or not
Output: concept: c             // describing the meaning of the word
Begin
  S[] = list of concepts describing all meanings of w   // following their WordNet ordering   1
  For i=0 to |S|                                         // processing concepts in-order       2
     If gloss(S[i]) contains "feeling" or "emotion"      // from most to least common          3
        Then c = [i]                                                                           4
  If c = ∅                                       // if no concept is identified yet               5
     If LESK = true                                                                            6
        Then c = simplified_LESK(w, C[])         // apply simplified LESK                      7
         Else c = S[0]                           // or select the most common concept          8
  Return c                                                                                     9
End
```

Fig. 6. Pseudo-code of *WSD_Heuristic* process

As for the WSD process, we combine: i) the well known *simplified LESK* algorithm [56], with ii) a simple and efficient heuristic method to handle the special case of direct affective words (i.e., words which directly carry sentiments or emotions). The *simplified LESK* algorithm compares the target word's context (its surrounding words in a sentence) with the contexts of its different possible meanings (concepts) in the lexical KB (where the context of a given concept consists of the set of synonymous terms and gloss descriptions of its surrounding concepts in the KB graph), and chooses the concept whose context is most similar to the target word context as its proper (disambiguated) meaning [56]. *Simplified LESK* is one of the most efficient WSD algorithms [116], requiring linear time w.r.t. the number of possible meanings (concepts) for a given word, and their context sizes. However, after conducting various experiments and closely analyzing the results, we realized that most sentiment-carrying concepts in WordNet contain the words "feeling" or "emotion" in their gloss definitions (e.g., *elation* is described as "a feeling of joy and pride", whereas *apathy* is described as "an absence of emotion or enthusiasm"). Also, the different possible meanings (concepts) for a given polysemous word are associated a natural ordering in WordNet, following their number of occurrences in a large text corpus (i.e., the Brown corpus [36], cf. Fig. 3), thus inferring their *common usage* in the English language. Based on the previous two observations, we provide a simple and efficient WSD heuristic method in Fig. 6, to further improve on LESK's efficiency. It consists of the following basic steps: i) search the glosses of the target word's polysemous concepts, and identify the one(s) containing terms "feeling" or "emotion" (lines 1-3), ii) if multiple concepts are identified, select the most common concept containing either terms as the proper word meaning (line 3-4), otherwise iii) if no concept glosses contain either terms (line 5), either apply the *simplified LESK* algorithm (lines 6-7) or (exclusively) select the most common concept as the proper meaning of the target word (line 8), following the user's preference (indicated by the *LESK* flag, line 6). The latter option (i.e., using the most common concept) is naturally sub-optimal compared with the former (i.e., running *LESK*), yet allows for fast (constant time) processing.

For instance, when processing direct affective word "creeps" (cf. Fig. 7.a) from sentence "man word be dryness give creeps", its second meaning will be chosen as the proper meaning since it contains word "feeling" in its gloss definition, without having to process its context words through legacy (*simplified LESK*) WSD. As for indirect affective words like "dryness" (cf. Fig. 7.b) which concepts do not contain neither "feeling" nor "emotion" words, it can be: i) either processed for WSD using the *simplified LESK* algorithm, which would identify concept #3 as its proper meaning in the above sentence, or ii) its first concept (the most common one) will be chosen to reflect its meaning (which is not the proper meaning in the above sentence). The latter depends on the user's preference: producing more reliable (*simplified LESK*) or faster (heuristic) disambiguation results.

The noun "creeps" has 2 senses (none from tagged texts)

1. **creeps** -- (a disease of cattle and sheep attributed to a dietary deficiency; characterized by anemia and softening of the ones and a slow still gait)
2. **creeps** -- (a feeling of fear and revulsion; "he gives me the creeps")

a. Meanings (concepts) of noun "*creeps*"

The noun "dryness" has 3 senses (first 1 from tagged texts)

1. (1) **dryness**, waterlessness, xerotes -- (the condition of not containing or being covered by a liquid (especially water))
2. sobriety, **dryness** -- (moderation in or abstinence from alcohol or other drugs)
3. dispassion, dispassionateness, **dryness** -- (objectivity and detachment, "her manner assumed a dispassion and dryness very unlike her usual tone")

b. Meanings (concepts) of noun "*dryness*"

Fig. 7. Concepts describing the meanings of words "creeps" and "dryness" in WordNet

3.2.2. Max_Affect component

The *Max_Affect* component's pseudo-code is described in Fig. 8. It accepts as input the users' disambiguated word concepts as well as their target affective categories (i.e., the emotions they are interested in evaluating in the input words), and then produces as output the corresponding sentiment scores in the form of a sentiment vector whose dimensions correspond to the user-chosen affective categories. To do so, it utilizes an adaptation of *Dijkstra*'s shortest path distance computations [21], applied on the LAG.

Max_Affect explores the LAG starting from one or multiple word concept nodes. From every starting concept node, it attempts to identify the closest path to every target affective category node, highlighting the target affect's expressiveness w.r.t. the source concept(s). However, we altered *Dijkstra*'s original premise: instead of identifying the minimum weight path between two nodes, *Max_Affect* seeks to identify the maximum sentiment weight of a source concept node c_i w.r.t. a target affect node a_j. To do so, we compute node and edge weights as follows:

i. The weight of a source concept node c_i w.r.t. a target affect node a_j, noted $w(c_i, a_j)$ or $w_{a_j}(c_i)$, is $\in [0, 1]$, where 0 means that affect category a_j is not expressed in c_i, whereas 1 means that a_j is totally expressed in c_i, Hence, the weight of c_i w.r.t. a set of target affect categories $A=\{a_1,\ldots,a_J\}$, consists of a vector of affect weights $V_i = < w(c_i, a_1), \ldots, w(c_i, a_J) >$, of J dimensions, where dimension j corresponds to a target affect category $a_j \in A$, and its vector coordinate $w(c_i, a_j)$ represents the affective weight of a_j w.r.t. c_i.

ii. The weight of an edge outgoing from node c_i and incoming into node c_r, noted $w(c_i, c_r)$, is $\in [0, 1]$ and reflects sentiment "conductance" where 0 means that the edge does not carry any sentiment expressiveness from c_i to c_j, whereas 1 means that the edge carries all the sentiment expressiveness from c_i to c_j. The edge weight is determined firstly based on the edge label (i.e., semantic relationship connecting the two nodes, e.g., *hypernymy*, *related to*, etc.), and secondly based on the out-degree of c_i (depending on the semantic relationship being processed). Following [102, 115], certain semantic relations are considered to be "reliable" in carrying emotions (namely: *hyponymy*, *similar to*, *related to*, *attribute*, *usage*, *derivation*, and *pertainymy*) while other relations are not completely reliable in that they only partially preserve affective meaning (namely: *hypernymy*, *entailment*, *verb group*, and *troponymy*). We mathematically concretize the latter using the following simple weight function:

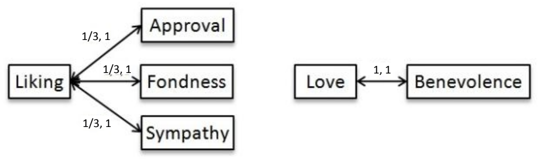
$$w(c_i, c_r) = \begin{cases} \dfrac{1}{\textit{out-degree}_{rel}(\mathrm{c_i})} & \text{if } rel \notin R_{reliable} \\ 1 & \text{otherwise} \end{cases} \quad \textbf{(3)}$$

where *rel* designates the edge's label (semantic relationship), and $R_{reliable}$ the set of sentiment reliable relationships (mentioned above). In other words, $w(c_i, c_r) = 1$ (maximum score) if its edge label corresponds to a sentiment reliable relationship, otherwise, it is determined by the out-degree of incoming node c_i. The rationale behind the latter is that an edge designates a stronger connection between two (word concept or affective category) nodes when it carries most of the descriptive power from the source node to the destination node, such that the source node has few other out-going connections (if any, cf. Fig. 9). Note that Formula 3 provides one possible cost model. Other more elaborate cost functions could be utilized later (e.g., assigning individual scores to individual relationships, following the user's needs).

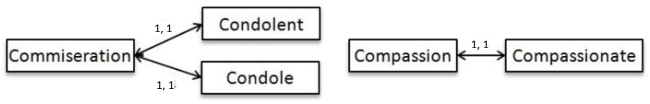
iii. Finally, instead of starting from an initial weight =0 assigned to the source lexical node c_i, *Max_Affect* starts with an initial weight =1 (maximum sentiment expressiveness), and then multiplies (instead of summing) the source node's weight by the weights of every edge on the maximum weight path leading to a_j. If all edges on the path between c_i and a_j are of maximum sentiment conductance (i.e., they carry all of the sentiment expressiveness), then $w(c_i, a_j) = w_{aj}(c_i) = 1$ where affect a_j is fully expressed in c_i. Otherwise, if edges have diminishing sentiment conductance, then $w_{aj}(c_i)$ will decrease accordingly.

```
Algorithm: Max_Affect
Input: LAG graph: G                          // lexical affective graph, connecting in our case: WordNet with WNAH
       Set of source word concept nodes: C   // corresponding to input word concepts (synsets), $C \subset G$
       Set of destination affect nodes: A    // corresponding to target affect categories (emotions), $A \subset G$
Ouput: Set of affect vectors: $\nabla$       // affect vectors associated to every source concept node in C w.r.t. affect nodes in A
Begin
  Initialize $\nabla = \{V_i\}_{i=1\ldots|C|}$ where $V_i = <w(c_i, a_1)=0, \ldots, w(c_i, a_{|A|})=0>$   // affect vector $V_i$ describing $c_i$ w.r.t. affects of A
  Initialize weights of nodes $\in$ G to 0
  Processed = $\varnothing$                  // Set of nodes that have been processed, i.e., for which affect vectors have been computed
  For every $c_i$ in C
  {
     Frontier = $c_i$, Explored = $\varnothing$          // Frontier is defined as a priority queue based on node weight
     Initialize $w(c_i) = 1$, remove from Frontier and add to Explored
     While (Explored $\neq$ A)                 // while not all destination affect nodes have been reached
     {
        For each node $c_j \in$ Explored
           For each node $c_m$ outgoing from $c_j$
              Add $c_m$ to Frontier
              Compute weight vector of $c_m$
                 $w(c_m) = max(w(c_m), w(c_m, c_j) \times w(c_j))$   // edge weight $w(c_m, c_j)$ is computed following Formula 3
           Compute maximum weight $w_{max}$ for all nodes in Frontier
           For each node $c_n$ in Frontier having $w(c_n) == w_{max}$
              If $c_n \in$ Processed Then
                 Compute $V_i = V_n \times w(c_n, c_i)$   // using affective vector computed in previous iteration
                 Add $c_i$ to Processed              // $c_i$ has been processed, so no need continue processing it
                 Goto Exit                          // break-out of the $c_i$ loop and go to the next input lexical concept node
              Else Remove $c_n$ from Frontier and add it to Explored
     }
     Exit:
     If ($c_i \notin$ Processed) Then
        Compute $V_i = <w(a_1), \ldots, w(a_{|A|})>$   // where $w(a_i)$ designates the affective weight of $a_i$ w.r.t. $c_i$
        Add $c_i$ to Processed
  }
  Return $\nabla$
End
```

Fig. 8. Pseudo-code of algorithm *Max_Affect*



a. Considering *partly reliable* relationships: *hypernymy/hyponymy*



b. Considering *reliable* relationship: *related* to

- Recall that edges in our LAG represent symmetric (*hyponymy*/*hypernymy*) or bidirectional relationships (e.g, *related to*, derivation, etc.).
- Hence, we represent the weight of an edge in the graph as a pair of weights separated by the "," symbol, representing the weight of the edge navigated in either directions.
- For instance, given the partly reliable *hyponymy*/*hypernymy* relationships in example *a*, and following Formula 3, w(*Liking*, *Approval*) = 1/3, whereas w(*Fondness*, *Liking*) = 1
- Likewise in example *b*, w(*Commiseration*, *Condolent*) = w(*Condolent*, *Commiseration*) = 1 since relationship *related to* is reliable (conductive) in carrying emotions

Fig. 9. LAG edge weights computed on extracts from 3.1.3, following semantic relationship *reliability* in carrying affects

Consider for instance the sample LAG in Fig. 10, where edge weights are computed following their semantic relationship reliability using Formula 3. Fig. 11 describes a sample run of *Max_Affect*, considering as source: word concept nodes *Compassionate* and *Care*, and as destination: affect nodes *Sympathy* and *Liking*.

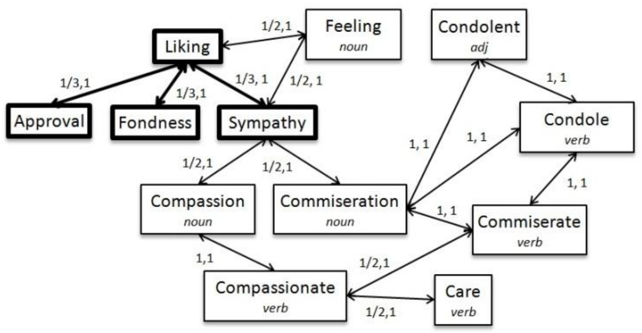


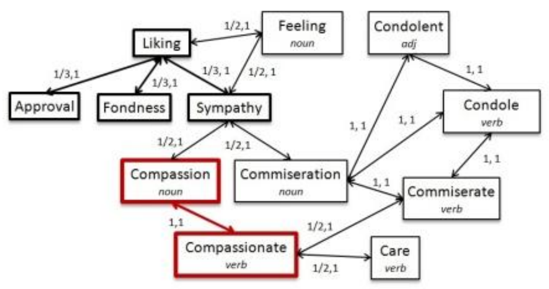
Fig. 10. Sample LAG (lexical affective graph) extracted from Fig. 5

One can realize from Fig. 11 that the *processed* nodes carry their optimal (maximum) affective vector weights throughout the different iterations of the algorithm, such that once computed for the same target affect dimensions, an affective vector can be directly utilized in computing the vectors of the node' neighbors (e.g., V_{Care} was computed based on $V_{Compassionate}$ produced in the previous iteration, without having to run an additional iteration of the algorithm navigating the graph from concept node *Care* to target nodes *Sympathy* and *Liking*).

3.2.3.Issues with Max_Affect

While *Max_Affect* provides a solution to perform LSA in a completely unsupervised manner, nonetheless, it suffers from two main drawbacks regarding: i) effectiveness and ii) efficiency.

In terms of effectiveness, we realized that semantic connectivity between affect concepts in the LAG does not always accurately portray their affective expressiveness. For instance, considering the LAG extract of Fig. 10, we can reach affect node *liking* from affect node *sympathy* through concept node *feeling* with a higher weight compared with the direct link between *sympathy* and *liking*, i.e., *w*(*feeling*, *sympathy*)=1/2 > and *w*(*liking*, *sympathy*)=1/3. In the example in Fig. 11, this led to $V_{Compassion}$ = <1/2, 1/4> and V_{Care} = <1/4, 1/8> (let us refer to this as result #1). Had we disregarded concept node *feeling* which connects *liking* with *sympathy*, and only used the direct affective connection between the latter two, we would have obtained $V_{Compassion}$ = <1/2, 1/6> and V_{Care} = <1/4, 1/12> (let's refer to this as result #2). At first glance, both results sound reasonable, and one cannot really judge which is better and which is worse. Yet, after empirically testing *Max_Affect* on the manually annotated ANEW word dataset [10, 100] (cf. Section 4), investigating *Max_Affect*'s produced scores, and carefully tweaking the algorithm, we realized that connections between affect concepts from WNAH are more reliable in carrying sentiment expressiveness compared with lexical and semantic connections from WordNet. In other words, when compared with human judgments in ANEW, results obtained when navigating the affective connections between affect nodes (similarly to result #2) were deemed more relevant following human sentiment ratings, compared with those obtained when navigating between affect nodes using otherwise lexical-or-semantic connections (similarly to result #1).



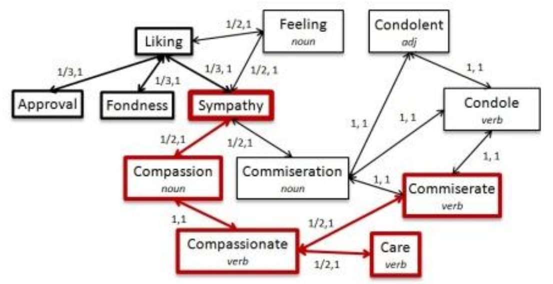
Considering the **first source** node ***Compassionate***:

– Initialize weight of source node to: $w(Compassionate) = 1$

– Initialize affective vector of source node w.r.t. both target affect nodes: $V_{Compassionate} = <0, 0>$

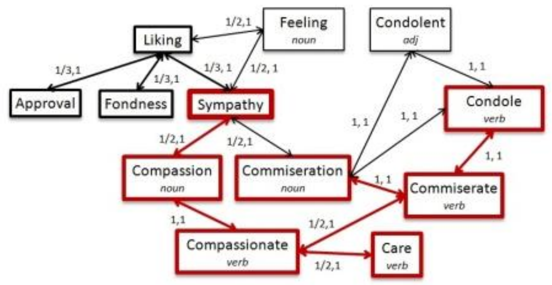
– Iteration #1: Fill neighbors in *Frontier* and identify maximum weight: $w(Compassion) = 1$

⇨ Include *Compassion* in *Explored* set



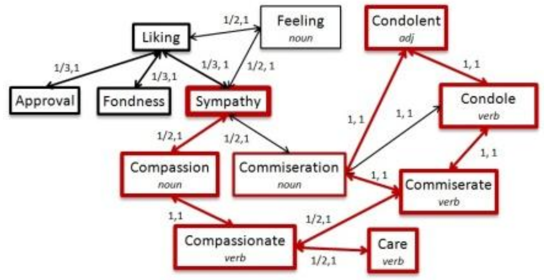
– Iteration #2: Fill neighbors in *Frontier* and identify maximum weight:
$w(Sympathy) = w(Care) = w(Commiserate)$
$= 1\times 1/2 = 1/2$

⇨ Remove them from *Frontier* and include them in *Explored*



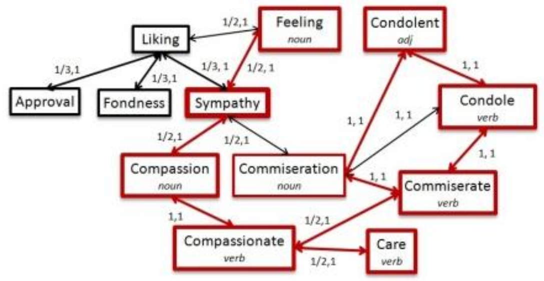
– Iteration #3: Fill neighbors in *Frontier* and identify maximum weight: $w(Commiseration) = w(Condole) = 1/2 \times 1 = 1/2$

⇨ Remove them from *Frontier* and include them in *Explored*



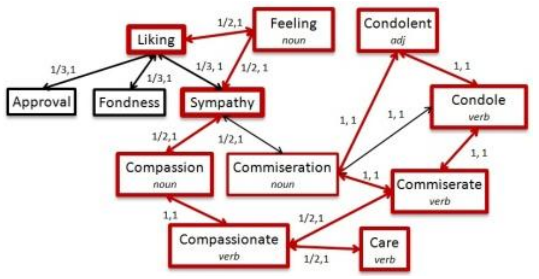
– Iteration #4: Fill neighbors in *Frontier* and identify maximum weight: $w(Condolent) = 1/2 \times 1 = 1/2$

⇨ Remove it from *Frontier* and include it in *Explored*



– Iteration #5: Fill neighbors in *Frontier* and identify maximum weight: $w(Feeling) = 1/2 \times 1/2 = 1/4$

⇨ Remove it from *Frontier* and include it in *Explored*



– Iteration #6: Fill neighbors in *Frontier* and identify maximum weight: $w(Liking) = 1/4 \times 1 = 1/4$
- Include it from *Frontier* and include it in *Explored*

Both targets are now in *Explored*:
- $V_{Compassionate} = < w(Sympathy), w(Liking)> = <1/2, 1/4>$

⇨ End processing of first source node

Considering the **second source** node ***Care*:**

– Initialize weight of source node to: $w(Care) = 1$

– Initialize affective vector of source node w.r.t. both target affect nodes: $V_{Care} = <0, 0>$

Iteration #1: Fill neighbors in *Frontier* and identify maximum weight: $w(Compassionate) = 1/2$
- Since *Compassionate* $\in$ *Processed*, we can directly compute $V_{Care} = w(Compassionate) \times V_{Compassionate}$
$= 1/2 \times <1/2, 1/6>$
$= <1/4, 1/12>$

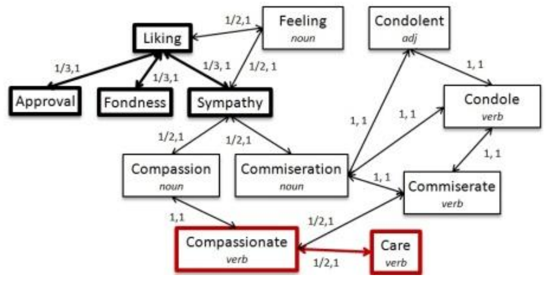


Fig. 11. Sample run of *Max_Affect*, from source words: *Compassionate* and *Care*, to destination affects: *Sympathy* and *Liking*

Also pertaining to effectiveness, another issue is the nature and direction of the edge being navigated. Following *Max_Affect*, if we disregard concept node *feeling* from the LAG in Fig. 10 (removing it along with its connections), navigating from *compassionate* to *liking* would produce $w_{liking}(compassionate) = w(compassion, compassionate) \times w(sympathy, compassion) \times w(liking, sympathy) = 1/2\times1\times1/2\times1/3 = 1/12$. While it does not look peculiar at first glance, yet a careful analysis of the result highlights an interesting observation regarding affect node weight propagation. Following *Max_Affect*'s logic, we cross from affect node *sympathy* to *liking* by considering the weight of *liking* carried toward *sympathy*: $w(liking, sympathy) = 1/3$. This is exactly similar to the logic applied when navigating from a concept node toward an affect node (e.g., crossing from concept node *compassion* to affect node *sympathy* requires the weight of *sympathy* carried toward *compassion*, $w(sympathy, compassion) = 1/2$). While the logic seems sound when navigating from concept nodes, nonetheless, it tends to break down when propagating weight scores between the affect nodes themselves. For instance, crossing from *sympathy* to *liking* should carry the whole weight of *sympathy* toward *liking*, $w(liking, sympathy)=1$, and not the other way around (as described above), since *sympathy-IsA-liking* where *IsA* (*hyponymy*) is a reliable (sentiment conductive) relationship. In other words, reaching affect node *sympathy* from any concept node c_i should be enough to identify c_i's sentiment weight w.r.t. affect *liking*, i.e., $w_{liking}(c_i) = w_{sympathy}(c_i)$ (e.g., considering concept node *compassionate* in Fig. 10, we would expect $w_{liking}(compassionate) = w_{sympathy}(compassionate) = 1/2\times1\times1/2\times1=1/4$).

As for efficiency, *Max_Affect* requires average polynomial (quadratic) time w.r.t. the size of the LAG covered in the navigation process (from source concept nodes to target affect nodes) which, despite LAG navigation optimizations and parallelization (cf. Section 4), remained relatively time consuming. This led us to provide an improved solution, considering the above mentioned effectiveness and efficiency issues in designing LISA 2.0.

3.3. LISA 2.0 - Affect Propagation and Lookup module

To address the issues mentioned above, LISA 2.0 includes three main components: i) *WNAH_Propagation* to handle affect score computation between affect nodes themselves considering their affective connections only, while disregarding word concepts and their lexical/semantic connections in the LAG (this allowed solving the LAG lexical/semantic connectivity problem of LISA 1.0), ii) *Back_Propagation* which propagates affect scores from user chosen affect nodes to all connected concept nodes in the LAG[9]. The set of affect-scored concepts form a *sentiment lexicon* which can be efficiently searched by iii) *Affect_Lookup* to identify word concept affect scores (handling LISA 1.0's efficiency problem). We describe LISA 2.0's components in following sub-sections.

3.3.1. WNAH_Propagation component

This component computes the sentiment scores of every affect node w.r.t. every other affect node in WNAH, such that each affect category becomes fully representative of all of the others. In other words, every affect node a_j in WNAH will be associated with a sentiment vector V_j consisting of 294 dimensions, where every dimension represents every other affect node in WNAH with its corresponding affect score w.r.t. a_j. On the one hand, this allows disregarding all lexical and sentiment concepts and connections when navigating between affect nodes in the LAG. On the other hand, instead of computing the maximum weight path between a word concept node c_i and all 294 affect nodes to get their sentiment scores (following *MaxAffect*, cf. Section 3.2), we only need to compute the path from c_i to the closest affect node a_j, where a_j would provide through its sentiment vector V_j all the scores for all other WNAH affect categories. The sentiment vector of c_i, V_i would be equal to V_j multiplied by the maximum path weight from c_i to a_j, i.e., $V_j = w(c_i, a_j)\times V_{i.}$

The pseudo-code of *WNAH_Propagation* is provided in Fig. 12. It takes as the input an affect hierarchy (namely: WNAH in our case, yet any other affect hierarchy can be utilized), and produces as output the set of affect vectors describing every affect category node in the hierarchy. After initializing all affect vectors to unit vectors following their reference affect nodes (cf. lines 1-3), the algorithm updates the vectors by including the weights of every node's parent and children nodes in the *hypernymy/hyponymy* hierarchy following Formula 3 (Step 1, lines 4-9). Affect nodes are processed in parallel (with every node assigned a dedicated thread[10]), where affect vectors are computed independently in every iteration. Then, the algorithm iterates once for every inner node in the hierarchy, processing all vectors in parallel in order to update their weights w.r.t. inner node connectivity (Step 2, lines 10-18).

[9] Recall that our approach is different from existing graph-based LSA methods in that we distinguish the affect concept hierarchy from the lexical KB, in order to process affective concepts separately following their affective relationships, before mapping them with their lexical counterparts with their lexical and semantic connections. To do so, we consider affective and lexical/semantic relationships and their weight combinations differently as discussed in Sections 3.2 and 3.3.

[10] The physical implementation of the algorithm is configured to run as many threads as necessary to process the different affect nodes, where thread scheduling and parallel execution are left to the operating system.

For instance, a node a_i having node a_j as its parent (or child), will have its affect vector updated w.r.t. the latter's, by multiplying their weights while preserving the maximum weight following every vector dimension (lines 15-17).

```
Algorithm: WNAH_Propagation
Input:  Affect hierarchy: WNAH              // Any hypernymy/hyponymy affect category could be used as input
Ouput: Set of affect vectors: $\nabla$      // Affect vectors associated to every affect node in WNAH
Begin
  M = |WNAH|                                                                                   1
  N = number of non-leaf nodes in WNAH                                                         2
  Initialize $\nabla^{(t0)} = \{V_i^{(t0)}\}_{i = 1 \ldots M}$  where  $V_i = <w(a_i, a_1)=0, \ldots, w(a_i, a_i)=1, w(a_i, a_M)=0>$    // Iteration $t_0$   3
  Step 1: Create Thread for each $V_i$ in $\nabla$                                   // Iteration $t_1$   4
  {                                                   // processing all nodes simultaneously   5
     For m=1 to M                                     // applying  Formula 3 on  every  node   6
        If  $a_m$ is parent of $a_i$   Then  $w(a_i, a_m)^{(t1)} = 1$                         7
        Else If $a_m$ is child of $a_i$  Then $w(a_i,a_m)^{(t1)} = \frac{1}{out\text{-}degree(a_i)}$   8
  }                                                                                            9
  Step 2: For n=2 to N                                // Iterations $t_2$ to $t_N$            10
  {                                                   // processing all nodes simultaneously  11
     Create Thread for each $V_i$ in $\nabla$                                                 12
        For m = 1 to M                     // for every affective dimension $a_m$ in $V_i$    13
           If  $w(a_i,a_m)^{(t_{n-1})} \neq w(a_i,a_m)^{(t_{n-2})}$   // test if weight of dimension $a_m$ was computed in previous iteration   14
           Then For j=1 to M               // update vector $V_i$ w.r.t. computed affect dimension $a_m$   15
                 If  $w(a_i,a_j)^{(t_{n-1})} == 0$                                            16
                 Then  $w(a_i,a_j)^{(t_n)} = max\left(w(a_i,a_j)^{(t_n)}, w(a_i,a_m)^{(t_{n-1})} \times w(a_m,a_j)^{(t_{n-1})}\right)$   17
  }                                                                                           18
  Return $\nabla^{(t_N)}$                  // final affect vectors are computed in N iterations   19
End
```

Fig. 12. Pseudo-code of *WNAH_Propagation* algorithm

Consider for instance the sample affective hierarchy in Fig. 13, extracted from Fig. 4, where edge weights are computed following *hypernymy/hyponymy* affective reliability (conductance) following Formula 3. Fig. 14 describes a sample run of *WNAH_Propagation*.

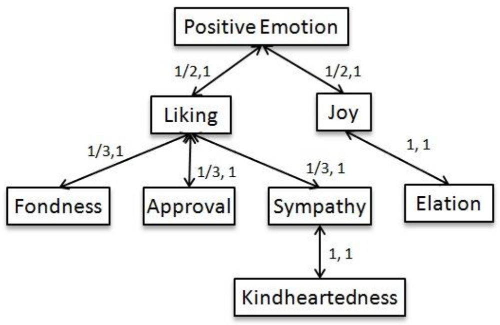


Fig. 13. Extract of the WNAH hierarchy from Fig. 4

Having computed all vectors for all affect nodes in WNAH, every affect node becomes fully descriptive of the affective scores of all other nodes in WNAH, such that accessing any affect node would give away all of WNAH's sentiment descriptiveness. In other words, reaching one single affective node a_j by navigating the LAG from an input user word concept c_i would instantly provide c_i with an affective vector V_i containing the affective scores of all 294 WNAH affects, without having to navigate the rest of the affective node hierarchy.

$$\nabla = \begin{array}{c|cccccccc} & PosEm & Liking & Joy & Fond & Appr & Symp & Elation & Kind \\ \hline V_{PosEm} & 1 & & & & & & & \\ V_{Liking} & & 1 & & & & & & \\ V_{Joy} & & & 1 & & & & & \\ V_{Fond} & & & & 1 & & & & \\ V_{Appr} & & & & & 1 & & & \\ V_{Symp} & & & & & & 1 & & \\ V_{Elation} & & & & & & & 1 & \\ V_{Kind} & & & & & & & & 1 \end{array}$$

a. Iteration t$_0$: Initializing all affect vector weights to 0, except for the dimension representing the affect itself, which is assigned maximum weight = 1. We do not show the zeros in the matrix for ease of presentation.

$$\nabla = \begin{array}{c|cccccccc} & PosEm & Liking & Joy & Fond & Appr & Symp & Elation & Kind \\ \hline V_{PosEm} & 1 & 1/2 & 1/2 & & & & & \\ V_{Liking} & 1 & 1 & & 1/3 & 1/3 & 1/3 & & \\ V_{Joy} & 1 & & 1 & & & & & \\ V_{Fond} & & 1 & & 1 & & & & \\ V_{Appr} & & 1 & & & 1 & & & \\ V_{Symp} & & 1 & & & & 1 & & 1 \\ V_{Elation} & & & 1 & & & & 1 & \\ V_{Kind} & & & & & & 1 & & 1 \end{array}$$

b. Iteration t$_1$: Updating the vectors by including the weights of every node's parent and children nodes in the *hypernymy/hyponymy* hierarchy following Formula 3. Vectors are processed in parallel, using multiple threads running simultaneously.

$$\nabla = \begin{array}{c|cccccccc} & PosEm & Liking & Joy & Fond & Appr & Symp & Elation & Kind \\ \hline V_{PosEm} & 1 & 1/2 & 1/2 & 1/6 & 1/6 & 1/6 & 1/2 & \\ V_{Liking} & 1 & 1 & 1/2 & 1/3 & 1/3 & 1/3 & 1/2 & 1/3 \\ V_{Joy} & 1 & 1/2 & 1 & 1/6 & 1/6 & 1/6 & 1 & \\ V_{Fond} & 1 & 1 & 1/2 & 1 & 1/3 & 1/3 & 1/2 & 1/3 \\ V_{Appr} & 1 & 1 & 1/2 & 1/3 & 1 & 1/3 & 1/2 & 1/3 \\ V_{Symp} & 1 & 1 & 1/2 & 1/3 & 1/3 & 1 & 1/2 & 1 \\ V_{Elation} & 1 & 1/2 & 1 & 1/6 & 1/6 & 1/6 & 1 & \\ V_{Kind} & 1 & 1 & 1/2 & 1/3 & 1/3 & 1 & 1/2 & 1 \end{array}$$

c. Iteration t$_2$: Updating the vectors w.r.t. weights computed in the previous iteration. Consider node *PosEm* (representing *positive emotion*) with vector V_{PosEm} for instance, $w(_{PosEm,\ Liking}) = 1/2$ was computed in iteration t$_1$. Hence we utilize it to update all weights of vector V_{PosEm} accordingly:
- $w(_{PosEm,\ Fond}) = w(_{PosEm,\ Liking}) \times w(_{Liking,\ Fond}) = 1/2 \times 1/3 = 1/6$
- $w(_{PosEm,\ Appr}) = w(_{PosEm,\ Liking}) \times w(_{Liking,\ Appr}) = 1/2 \times 1/3 = 1/6$
- $w(_{PosEm,\ Symp}) = w(_{PosEm,\ Liking}) \times w(_{Liking,\ Symp}) = 1/2 \times 1/3 = 1/6$

Likewise, $w(_{PosEm,\ Joy}) = 1/2$ computed in the previous iteration t$_1$, allows to update V_{PosEm} accordingly:
- $w(_{PosEm,\ Elation}) = w(_{PosEm,\ Joy}) \times w(_{Joy,\ Elation}) = 1/2 \times 1 = 1/2$

The same process is applied simultaneously (in parallel) to every other node vector in the hierarchy.

$$\nabla = \begin{array}{c|cccccccc} & PosEm & Liking & Joy & Fond & Appr & Symp & Elation & Kind \\ \hline V_{PosEm} & 1 & 1/2 & 1/2 & 1/6 & 1/6 & 1/6 & 1/2 & 1/6 \\ V_{Liking} & 1 & 1 & 1/2 & 1/3 & 1/3 & 1/3 & 1/2 & 1/3 \\ V_{Joy} & 1 & 1/2 & 1 & 1/6 & 1/6 & 1/6 & 1 & 1/6 \\ V_{Fond} & 1 & 1 & 1/2 & 1 & 1/3 & 1/3 & 1/2 & 1/3 \\ V_{Appr} & 1 & 1 & 1/2 & 1/3 & 1 & 1/3 & 1/2 & 1/3 \\ V_{Symp} & 1 & 1 & 1/2 & 1/3 & 1/3 & 1 & 1/2 & 1 \\ V_{Elation} & 1 & 1/2 & 1 & 1/6 & 1/6 & 1/6 & 1 & 1/6 \\ V_{Kind} & 1 & 1 & 1/2 & 1/3 & 1/3 & 1 & 1/2 & 1 \end{array}$$

c. Iteration t$_3$: The same process is repeated until covering all non-leaf nodes in the hierarchy. Consider node *PosEm* with vector V_{PosEm} for instance for instance, $w(_{PosEm,\ Fond}) = 1/6$ was computed in iteration t$_2$, hence we utilize it to update all weights of vector V_{PosEm} accordingly. Yet, the only weight which will be updated here is:
- $w(_{PosEm,\ Kind}) = w(_{PosEm,\ Fond}) \times w(_{Fond,\ Kind}) = 1/2 \times 1/3 = 1/6$

The other weights of vector V_{PosEm} will not be updated in this iteration (considering the $w(_{PosEm,\ Appr})$, $w(_{PosEm,\ Symp})$, and $w(_{PosEm,\ Elation})$ computed in iteration t$_2$) since maximum weights for all dimensions of V_{PosEm} have been reached already.

The algorithm ends in this iteration (i.e., t$_3$) which number corresponds to the number of inner (non-leaf) nodes in the considered hierarchy (cf. Fig. 13), after having computed all affect vectors w.r.t. all considered affect nodes in the hierarchy.

Fig. 14. Sample run of algorithm *WNA_Propagation* on the WNAH extract in Fig. 13

3.3.2. Back_Propagation component

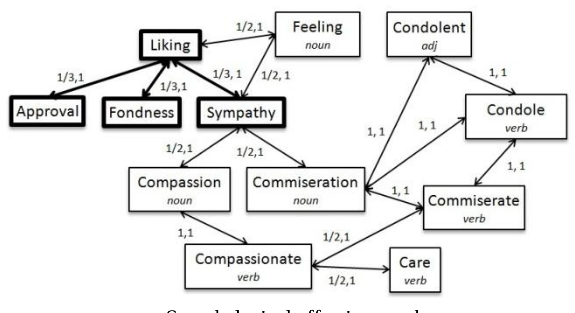
Having computed the affect scores of all affect nodes in WNAH (using *WNA_Propagation*), the *Back_Propagation* component propagates the produced affect scores from user chosen affect nodes to all connected concept nodes in the LAG. As a result, all lexical concepts connected (directly through an edge, or indirectly through a path) with any affect node acquires an affect score, forming a *sentiment lexicon*. The latter can then be utilized to perform sentiment analysis by simply looking-up the affect vectors of the target lexical concepts from the lexicon.

The *Back_Propagation* pseudo-code is shown in Fig. 15. It is a variation of *Dijkstra*'s maximum weight process utilized in *Max_Affect*, with the following modifications (highlighted in the pseudo-code): i) a set of source affect nodes $A \in$ G along with their affect vectors ∇_A (pre-computed using *WNA_Propagation*); it does not require a set of lexical concept nodes as input since it will process all of them $\in G$ (line 1), ii) the algorithm navigates the LAG starting from all source affect nodes in parallel (with a dedicated thread assigned to every source node), where affect vectors are computed independently in every iteration (lines 4-22), iii) it navigates from every source affect node toward its surrounding concept nodes (lines 4-11) and beyond, back-propagating toward all connected concept nodes (lines 12-16), iv) affect vectors of lexical nodes are computed directly from those of their connected affect node vectors (lines 17-20, instead of computing their weights individually such as with *Max_Affect*), and v) the maximum affect vector weights for all concept nodes produced from every source affect node (i.e., from every thread) are finally retained (line 23).

```
Algorithm: Back_Propagation
Input: Lexical affective graph (LAG): G
        Set of source affect nodes: A                 // target affect nodes (emotions), $A \subset G$
        Set of affect vectors for A: $\nabla_A$       // affect vectors pre-computed for every affect node in A
Ouput: Set of concept affect vectors: $\nabla_C$      // affect vectors associated with every concept node in C
Begin
  C = set of all word concept nodes in G
  Initialize weights of nodes $\in$ G to 0
  Frontier = $\emptyset$, Explored = $\emptyset$      // Frontier is defined as a priority queue based on node weight
  Create Thread for every $a_i$ in A                  // starting from affect nodes and navigating toward concept nodes
  {
     $\nabla_i$ = set of concept affect vectors when navigating from $a_i$
     Initialize $\nabla_i = \{V_j\}_{j=1\ldots|C|}$ where $V_j = <w(c_j, a_1)=0, \ldots, w(c_j, a_{|A|})=0>$   // affect vector $V_j$ describing $c_j$ w.r.t. affects of A
     Add $a_i$ to Frontier                            // any affect node could be a starting node
     Initialize $w(a_i)=1$, remove from Frontier and add to Explored
     While (Explored $\neq$ C)                        // while not all concept nodes have been reached
     {
        For each node $x_j \in$ Explored              // $x_j$ could be an affect node or a concept node
          For each node $c_m$ outgoing from $x_j$     // $c_m$ should be strictly a concept node
             Add $c_m$ to Frontier
             Compute weight vector of $c_m$
                 $w(c_m) = max(w(c_m), w(x_j) \times w(x_j, c_m))$   // edge weight $w(x_j, c_m)$ is computed following Formula 3
          Compute maximum weight $w_{max}$ for all nodes in Frontier
          For each node $c_n$ in Frontier having $w(c_n) == w_{max}$
             Compute $V_n = max(V_n, w(a_i, c_n) \times V_i)$   // using affective vector computed in previous iteration
             Remove $c_n$ from Frontier and add it to Explored
     }
  }
  $\nabla_C = max(\nabla_i)_{i=1\ldots|A|}$           // retaining maximum affective vector weights
  Return $\nabla_C$
End
```

Fig. 15. Pseudo-code of *Back_Propagation* algorithm

Consider the same sample LAG example in Fig. 10, reported in Fig. 16.a for ease of presentation. Fig. 17 describes a sample run of the *Back_Propagation* algorithm, starting from affective node *Sympathy* and propagating its affective scores toward all concept nodes in the graph. Remaining threads[11] starting from other affective nodes are computed similarly in Fig. 18. Note that if no lexical or semantic relationships exist between affect nodes from outside of the affect hierarchy (e.g, if concept node *feeling* was not connected to affect nodes *liking* and *sympathy*), then one single thread, starting from any affective node in the LAG, would suffice to run *Back_Propagation*.



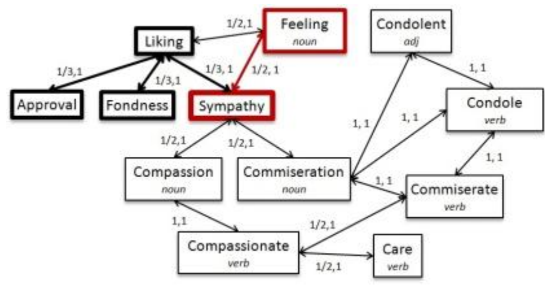
a. Sample lexical affective graph

$$\nabla_A = \begin{array}{c|cccc} & \textit{Liking} & \textit{Fond} & \textit{Symp} & \textit{Appr} \\ \hline V_{Liking} & 1 & 1/3 & 1/3 & 1/3 \\ V_{Fond} & 1 & 1 & 1/3 & 1/3 \\ V_{Symp} & 1 & 1/3 & 1 & 1/3 \\ V_{Appr} & 1 & 1/3 & 1/3 & 1 \end{array}$$

b. Input affect vectors for all affect nodes, pre-computed by *WNA_Propagation* algorithm:

Fig. 16. Sample LAG, reported from Fig. 10, along with its input affective node vectors

[11] In the physical implementation of the algorithm, thread scheduling and parallel execution are left to the operating system.

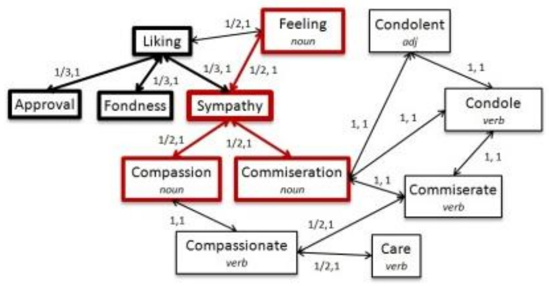


a. <u>**Thread #1**</u>: Considering **source** node ***Sympathy***:

- Initialize weight of source node: *w*(*Sympathy*) = 1
- Affective vector of sympathy w.r.t. all considered affect categories has been (pre-computed) provided: $V_{Sympathy} = <1 \quad 1/3 \quad 1 \quad 1/3>$ following dimensions *Liking*, *Fondness*, *Sympathy*, and *Approval* respectively
- <u>Iteration #1</u>: Fill neighbors in *Frontier* and identify maximum weight: $w_{Symp}(Feeling) = 1$
    - Compute affective vectors:

$$\nabla_{Symp} = \begin{array}{c|cccc} & Liking & Fond & Symp & Appr \\ V_{Feeling} & 1 & 1/3 & 1 & 1/3 \\ V_{Compassion} & & & & \\ V_{Commiseration} & & & & \\ V_{Compassionate} & & & & \\ V_{Condolent} & & & & \\ V_{Condole} & & & & \\ V_{Commiserate} & & & & \\ V_{Care} & & & & \end{array}$$

⇨ Remove node *Feeling* from *Frontier* and include in *Explored* set

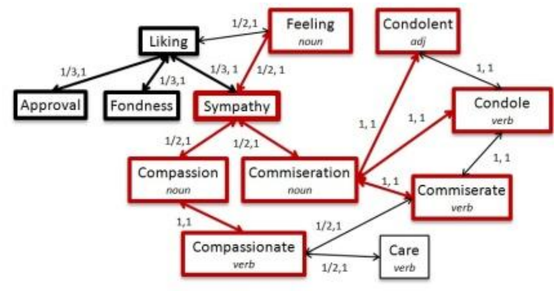


b. <u>Iteration #2</u>: Fill neighbors in *Frontier* and identify maximum weight: w_{Symp} (*Compassion*) = w_{Symp}(*Commiseration*) = 1/2

- Compute affective vectors:

$$\nabla_{Symp} = \begin{array}{c|cccc} & Liking & Fond & Symp & Appr \\ V_{Feeling} & 1 & 1/3 & 1 & 1/3 \\ V_{Compassion} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiseration} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Compassionate} & & & & \\ V_{Condolent} & & & & \\ V_{Condole} & & & & \\ V_{Commiserate} & & & & \\ V_{Care} & & & & \end{array}$$

⇨ Remove both nodes from *Frontier* and include in *Explored* set



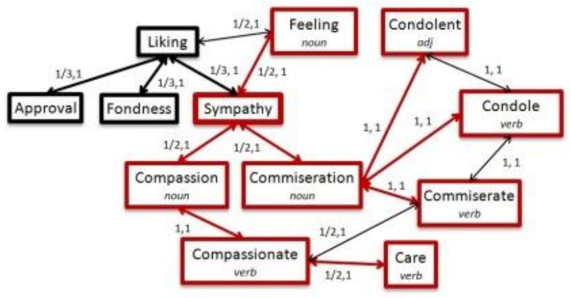
c. <u>Iter. #3</u>: Fill neighbors in *Frontier* and identify maximum weight:

$$w_{Symp}(Compassionate) = w_{Symp}(Condolent) = w_{Symp}(Condole) = w_{Symp}(Commiserate) = 1/2 \times 1 = 1/2$$

- Compute affective vectors:
  $V_{Compassionate} = V_{Condolent} = V_{Condole} = V_{Commiserate} = <1/2 \quad 1/6 \quad 1/2 \quad 1/6>$

$$\nabla_{Symp} = \begin{array}{c|cccc} & Liking & Fond & Symp & Appr \\ V_{Feeling} & 1 & 1/3 & 1 & 1/3 \\ V_{Compassion} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiseration} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Compassionate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condolent} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condole} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiserate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Care} & & & & \end{array}$$

⇨ Remove all latter nodes from *Frontier* and include them in *Explored*



d. <u>Iteration #4</u>: Fill neighbors in *Frontier* and identify maximum weight: *w*(*Care*) = 1/2 × 1/2 = 1/4

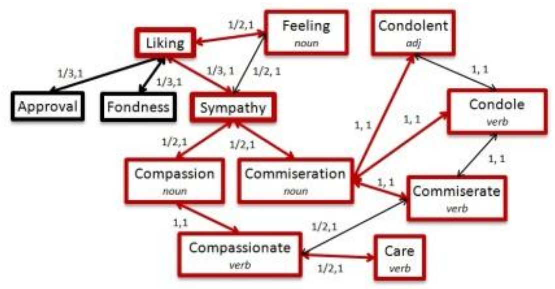
- Compute affective vector: $V_{Care} = <1/4 \quad 1/12 \quad 1/4 \quad 1/12>$

$$\nabla_{Symp} = \begin{array}{c|cccc} & Liking & Fond & Symp & Appr \\ V_{Feeling} & 1 & 1/3 & 1 & 1/3 \\ V_{Compassion} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiseration} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Compassionate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condolent} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condole} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiserate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Care} & 1/4 & 1/12 & 1/4 & 1/12 \end{array}$$

⇨ Remove *Care* from *Frontier* and include in *Explored*

End of Thread #1 since all words have been reached from *Sympathy*

Fig. 17. Sample run of one of the thread of the *Back_Propagation* algorithm on the LAG extract in Fig. 16. Threads 2-to-4 are described in Fig. 18. Recall that threads are executed in parallel. We only number them in this example for ease of presentation



a. **Thread #2:** Considering **source** node ***Liking***:

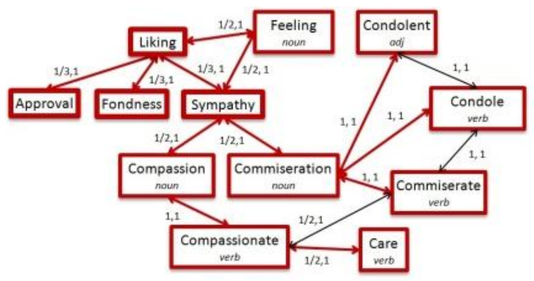
– Produces the same results obtained from *Sympathy* except for concept node *Feeling* which is reached from *Liking* through $w(Feeling) = 1$

- Hence, $V_{Feeling} = 1 \times V_{Liking} = \langle 1 \quad 1/3 \quad 1/3 \quad 1/3 \rangle$

$$\nabla_{Liking} = \begin{array}{c|cccc} & Liking & Fond & Symp & Appr \\ V_{Feeling} & 1 & 1/3 & 1/3 & 1/3 \\ V_{Compassion} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiseration} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Compassionate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condolent} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condole} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiserate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Care} & 1/4 & 1/12 & 1/4 & 1/12 \end{array}$$

– **Thread #3 and 4:** Considering **source** nodes ***Approval*** and ***Fondness*** produce the same result obtained with *Liking*

⇨ $\nabla_{Liking} = \nabla_{Appr} = \nabla_{Fond}$



b. Final affective weight vectors:

$$\nabla_C = \max(\nabla_{Symp}, \nabla_{Liking}, \nabla_{Appr}, \nabla_{fond})$$

$$= \begin{array}{c|cccc} & Liking & Fond & Symp & Appr \\ V_{Feeling} & 1 & 1/3 & 1 & 1/3 \\ V_{Compassion} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiseration} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Compassionate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condolent} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Condole} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Commiserate} & 1/2 & 1/6 & 1/2 & 1/6 \\ V_{Care} & 1/4 & 1/12 & 1/4 & 1/12 \end{array}$$

Fig. 18. Sample run of the remaining threads of the *Back_Propagation* algorithm executed on the LAG extract in Fig. 16. Recall that threads are executed in parallel and are only numbered here for ease of presentation

For instance, results in Fig. 17 show that word concepts *compassion* and *commiserate* express the same amount of *sympathy*, despite being at different locations in the LAG. On the one hand, *compassion* is linked directly with *sympathy* via a partly reliable *hypernymy* (*HasA*) relationships, resulting in $w_{sympathy}(compassion) = 1/2$. On the other hand, *commiserate* is linked with *sympathy* through a path made of two relationships: one partly reliable (*sympathy-HasA-commiseration*) and one reliable (*commiseration-Derivation-commiserate*), yielding $w_{sympathy}(commiserate) = 1/2 \times 1 = 1/2$. Yet, concept *compassion* expresses more *sympathy* compared with concept *care* which is farther away in the LAG, where *care* is connected with *sympathy* via two partly reliable semantic relations (i.e., *sympathy-HasA-compassion* and *compassionate-Troponymy-care*) and one reliable relation in-between the latter (i.e., *compassion-Derivation-compassionate*), thus yielding: $w_{Sympathy}(Care) = 1/2 \times 1 \times 1/2 = 1/4$.

The final concept affect scores are computed as the maximum weights among all affect vectors produced from every affect source node (cf. ∇_C in the final result shown in Fig. 18.b).

3.3.3. Affect_Lookup component

The resulting set of affect-scored concepts (i.e., ∇_C) forms a *sentiment lexicon* which can be efficiently searched to lookup any word concept affect score. For instance, the affect score of concept *care* w.r.t. affect category *approval* can be directly identified as =1/12 by looking it up from ∇_C. This is handled by the *Affect_Lookup* component, which makes use of legacy indexing techniques (e.g., B+ Tree [31]) to access and efficiently search ∇_C. We do not describe *Affect_Lookup* further here since it comes down to a typical data lookup process.

To sum-up, the LISA 2.0 module, through its *Affect_Lookup* component (which makes use of the sentiment lexicon produced by *Back-Popagation* and *WNA_Propagation*), allows to transform the problem of LSA from a (polynomial) graph navigation problem (with LISA 1.0) into a fast (logarithmic) data (lexicon) lookup problem. At the same time, LISA 2.0's lexicon construction process (through *Back-Popagation* and *WNA_Propagation*) is fully automated and does not require any training or manual effort.

3.4. Complexity Analysis

The overall time complexity of LISA 1.0 simplifies to $O(|w|\times|G^*|^2)$ where $|w|$ represents the number of input words to be processed for LSA, G the LAG (lexical affective graph) graph utilized for affect navigation, G^* the part of G which is navigated before reaching the destination affect nodes (where the execution process ends), and $|G^*|$ its number of nodes (concepts). It is evaluated as the sum of the complexities of LISA 1.0's main *WSD_Heuristic* and *Max_Affect* components. Considering one single target word w:

- *WSD_Heuristic* simplifies to best case $O(1)$ when the most common meaning of the word is chosen, and worst case $O(|senses(w)| \times (|context(w)|+|context(c_i)|)$ when the *simplified LESK* process is executed, where $|senses(w)|$ designates the total number of possible senses (concepts) of the source word w in the reference lexical KB (e.g., WordNet), $|context(w)|$ the cardinality (in number of terms) of the source word's context within its surrounding input text (if available), $c_i \in senses(w)$ a possible sense (concept) for source word w, and $|context(c_i)|$ the cardinality of the context of c_i in the reference KB.
- *Max_Affect* simplifies to $O(|G^*|^2)$ time, as it explores for every node $x_i \in G^*$ all its neighbor nodes $neighbors(x_i)$ in order to identify the neighbors' maximum edge weights incoming from x_i. This process is repeated iteratively until reaching the target (affect) nodes.

As for LISA 2.0, we analyze its complexity separately for: i) affect propagation, and i) affect lookup:

- In terms of *affect propagation* (leading to the creation of a sentiment lexicon), LISA 2.0 simplifies to $O(|G|^2)$, as the sum of the complexities of its *WNAH_Propagation* and *Back_Propagation* components:
    - *WNAH_Propagation* requires $O(|WNAH|^2)$ where $|WNAH|$ designates the number of nodes in *WNAH*. While the latter is constant, one could deduce that this component would require $O(1)$ constant time. Nonetheless, our approach is not limited to *WNAH* in its current form, and can be applied to any variation of it or to any other affect reference sharing similar hierarchical properties, thus emphasizing its dynamic complexity.
    - *Back_Propagation* simplifies to $O(|G|^2)$ as it follows a computational process comparable to that of *Max_Affect*, where back-propagation is executed on the whole of graph G.
- In terms of *Affect_Lookup* (to perform word-level LSA based on the sentiment lexicon generated in the previous phase), LISA 2.0 requires $O(|w|\times log(|G|))$ to perform data lookup for every input word on the affect-scored concepts of G (which form the sentiment lexicon) where legacy indexing techniques (e.g., B+ Tree [31]) are used to speed-up the lookup process.

4. Experimental Evaluation

We have implemented LISA 1.0 and 2.0 to test and evaluate their performance, and compare them with recent alternatives in the literature. Written in Java, our implementation comprises LISA's main modules and components, in addition to a linguistic pre-processing component to perform *tokenization*[23], *stop word removal*[22], and *stemming*[24]. WordNet 3.0 was utilized as the reference lexical KB, and WNAH as the affect KB. The experimental prototype, test data, and test results are available online[12].

In the following, we first describe the experimental data and pre-processing steps in Sections 4.1 and 4.2. Evaluation metrics are described in Section 4.3. Polarity and affect evaluation results are developed in Section 4.4 and 4.5 respectively, before presenting performance evaluation results in Section 4.6. In summary, results show that LISA 2.0 outperforms LISA 1.0 in both LSA quality and performance, while being on a par with existing (semi)supervised approaches (without the need for training or manual effort).

4.1. Experimental Data and Pre-Processing

We utilized the ANEW (Affective Norms for English Words) dataset [10, 100] to evaluate our approach. Developed at the University of Florida's Center for Emotion and Attention Studies, ANEW consists of 1024 words in the English language, manually rated in terms of *pleasure*, *arousal*, *dominance* in [10] as well as *happiness*, *anger*, *sadness*, *fear*, and *dislike/disgust* in [100]. Ratings were conducted by a large number of psychology students equally distributed between female and male candidates. Ratings for every dimension were provided on a 9-point scale in [10] and on a 5-point scale in [100] following the SAM (Self-Assessment Manikin) system [64], which can be translated into integers ($\in[1, 9]$ or $\in[1, 5]$) designating [*min*, *max*] expressiveness. For instance, *pleasure* was rated from *no pleasure* (=1) to *extreme pleasure* (=9), and arousal from *not aroused* (=1) to *extremely aroused* (=9).

[12] Available online at: http://sigappfr.acm.org/Projects/LISA

Hence, we normalized ANEW's ratings using Formulas 4, 5, and 6 to obtain scores $\in$[0, 1], representing them in a common referential which would be easier to compare with LISA and other existing LSA methods. As for the afore mentioned dimensions, we considered *pleasure* to describe word polarity (*negative*-to-*positive*), and *happiness*, *anger*, *sadness*, *fear*, and *dislike/disgust* to describe their respective affect categories[13].

$$pos_{Norm} = \frac{pleasure-1}{\max(pleasure)} \in [0,1] \quad \text{and} \quad neg_{Norm} = 1\text{-}\left(\frac{pleasure-1}{\max(pleasure)}\right) \in [0,1] \tag{4}$$

$$happy_{Norm} = \frac{happy-1}{\max(happy)} \in [0,1] \quad \text{likewise for } anger, fear, sadness, \text{ and } dislike \tag{5}$$

Note that certain existing LSA methods, e.g., [6, 14, 53], produce polarity scores $\in$[-1, 1], varying from absolutely *negative* (score=-1) to absolutely *positive* (score=1). The latter were also normalized to the [0, 1] interval using Formula 6:

$$pos_{Norm} = \frac{polarity+1}{2} \in [0,1] \quad \text{and} \quad neg_{Norm} = 1\text{-}\left(\frac{polarity+1}{2}\right) \in [0,1] \tag{6}$$

4.2. LISA Score Normalization

While sentiment scores produced by LISA are $\in$[0, 1] following the weight cost model and navigation processes adopted in our approach, nonetheless, the latter need to be further processed to allow a more accurate evaluation w.r.t. existing methods. On the one hand, polarity scores need to be normalized to obtain *pos* + *neg* = 1 while preserving the variation in the original values. To do so, we applied a geometric translation process on both scores as follows. Considering pos_{Raw} and neg_{Raw} to be the raw (original) scores produced by LISA[14], we start by computing corresponding variation ratios (Step 1). Then we compute relative variations as the averages of raw and variation ratio scores (Step 2). The median of the relative variation scores is then computed (Step 3), followed by average score distances from the median (Step 4). Consequently, normalized scores are obtained by adding the obtained distance scores to the median (Step 5).

$$\begin{aligned}
&\text{pos}_{\text{Ratio}} = \frac{\text{pos}_{\text{Raw}}}{\text{pos}_{\text{Raw}} + \text{neg}_{\text{Raw}}} \quad \text{and} \quad \text{neg}_{\text{Ratio}} = \frac{\text{neg}_{\text{Raw}}}{\text{pos}_{\text{Raw}} + \text{neg}_{\text{Raw}}} && \text{// Step 1}\\
&\text{pos}_{\text{Rel}} = \frac{\text{pos}_{\text{Raw}} + \text{pos}_{\text{Ratio}}}{2} \quad \text{and} \quad \text{neg}_{\text{Rel}} = \frac{\text{neg}_{\text{Raw}} + \text{neg}_{\text{Ratio}}}{2} && \text{// Step 2}\\
&\text{Median} = \frac{\text{pos}_{\text{Avg}} + (1-\text{neg}_{\text{Avg}})}{2} && \text{// Step 3}\\
&\text{pos}_{\text{Dist2Med}} = |\text{pos}_{\text{Avg}} - \text{Median}| \quad \text{and} \quad \text{neg}_{\text{Dist2Med}} = |\text{Median} - (1-\text{neg}_{\text{Avg}})| && \text{// Step 4}\\
&\text{pos}_{\text{Norm}} = \text{pos}_{\text{Avg}} + \text{pos}_{\text{Dist2Med}} \quad \text{and} \quad \text{neg}_{\text{Norm}} = \text{neg}_{\text{Avg}} + \text{neg}_{\text{Dist2Med}} && \text{// Step 5}
\end{aligned} \tag{7}$$

For instance, $pos_{Raw} = 0.1$ and neg_{Raw} = 0.3 would produce normalized scores pos_{Norm}=0.325 and neg_{Norm}=0.675, where pos_{Norm+} neg_{Norm} =1.

As for affect category scores, we normalize them w.r.t. the number of user chosen target affect categories such that their sum becomes equal to 1:

$$\text{For i=1 to } |A| \quad \left\{ \text{aff}_{i_{\text{Norm}}} = \frac{\text{aff}_{j_{\text{Raw}}}}{\sum_{j=1..|A|} \text{aff}_{j_{\text{Raw}}}} \right\} \quad \text{where} \sum_{j=1..|A|} \text{aff}_{j_{\text{Norm}}} = 1 \tag{8}$$

where A is the set of affect categories chosen by the user as targets for LSA[15].

[13] We disregard *arousal* and *dominance* in our current experiments since they reflect *behavioral* rather than *affective* dimensions [10].

[14] The latter are obtained from nodes *positive emotion* and *negative emotion*, either i) by reaching them through LAG navigating when using LISA 1.0, or ii) by looking them up from the sentiment lexicon when using LISA 2.0.

[15] Note that certain affects can be grouped in pairs of opposite categories (e.g., *love-hate*, *like-dislike*) [24], and can thus be normalized pair-wise similarly to polarity scores (cf. Formula 6). Yet, we adopt a more general normalization function (cf. Formula 8) in our current study in order to evaluate the set of non-opposite affect categories provided in ANEW.

4.3. Evaluation Metrics

To evaluate our approach, we ran LISA 1.0 and 2.0 on the ANEW dataset and compared our results with ANEW's normalized word ratings using *Pearson Correlation Coefficient* (*PCC*) and *Mean Squared Error* (*MSE*):

$$PCC(\mathrm{X},\mathrm{Y}) = \frac{\delta \mathrm{XY}}{\delta \mathrm{X} \times \delta \mathrm{Y}} \quad \in [\text{-}1,\ 1] \tag{9}$$

$$MSE(\mathrm{X},\mathrm{Y}) = \frac{1}{\mathrm{J}} \times \sum_{\mathrm{i}=1\ldots\mathrm{J}} (\mathrm{x_j} - \mathrm{y_j})^2 \quad \in [0,\ max_{Err}] \tag{10}$$

where: $x \in X$ and $y \in Y$ designate ANEW (human) and system (LISA) generated sentiment scores respectively, δX and δY denote the standard deviations of X and Y respectively, δXY denotes the covariance between X and Y, and $J = max(|X|, |Y|)$ is the maximum vector size[16]. The values of *PCC* are $\in$ [-1, 1], i.e., 1 for maximum correlation, 0 for no correlation, and -1 for negative correlation[17]. *MSE* is a distance measure evaluating the separation between X and Y as the average of the squares of their difference errors. *MSE* $\in[0,\ max_{Err}]$, i.e., 0 for minimum error distance and max_{Err} for maximum error distance, where max_{Err}=1 in our case since both X and Y values $\in[0, 1]$.

A high quality LSA method would naturally produce: i) high *PCC* scores: which means that the system generated sentiment scores are closely correlated with the user (ANEW) ratings; ii) and low *MSE* scores: meaning that the system generated sentiment scores are not distant from the user ratings. In addition to comparing one approach's *PCC* improvement to another's *MSE* distance, and given the matching boundaries of *PCC* and *MSE* in our experimental scenario (with max_{Err}=1), we combine them into a single measure which we designate as: *closeness*, computed as the linear combination of the absolute value of *PCC* with the inverse of *MSE*:

$$closeness(X,Y) = \alpha \times \left|PCC(\mathrm{X},\mathrm{Y})\right| + (1-\alpha) \times \left(1 - MSE(\mathrm{X},\mathrm{Y})\right) \qquad \in [\text{-}1,\ 1]$$
$$\text{where } \alpha \in [0,\ 1] \tag{11}$$

Therefore, high absolute *PCC* (considering both positive and negative correlation) and low *MSE* would produce high *closeness*, indicating (in the case of positive *PCC*) excellent LSA quality.

4.4. Polarity Evaluation Quality

We compared LISA 1.0 and 2.0 with ANEW as well as four recent polarity (opinion) detection methods available online: SentiWordNet 3 [6], SenticNet 3 [14], SenticNet 5 [16][18], and AlchemyAPI [53]. The results of alternative solutions were produced based on the sentiment scores extracted from their original studies (available online). For each of the latter methods, we identified the ANEW words matching with the corresponding lexicon entries to produce the corresponding polarity scores. A snapshot of the results is provided in Fig. 19 and Fig. 20. The complete set of empirical graphs and corresponding data is provided online[19]. Results are also summarized Table 1.

Fig. 19 shows *positive* polarity scores w.r.t. ANEW, where words have been ranked following ANEW's *positive* intensities (from highest to lowest). Similar graphs were produced for *negative* polarity scores and are provided online. Three main observations can be made here. First, one can realize that LISA 2.0 produced results which are more consistently distributed following ANEW's ratings compared with LISA 1.0. Second, LISA 2.0's results show concentrations of score points around the ANEW reference score line, with clusters of points forming around *positive* polarity scores = 0.8, 0.64, 0.5, 0.37, and 0.18 (highlighted in Fig. 19.b) following ANEW's slope. This highlights LISA 2.0's quality in producing scores which correlate more closely with ANEW's manual ratings compared with LISA 1.0. Third, most alternative solutions which are supervised produce polarity scores which are

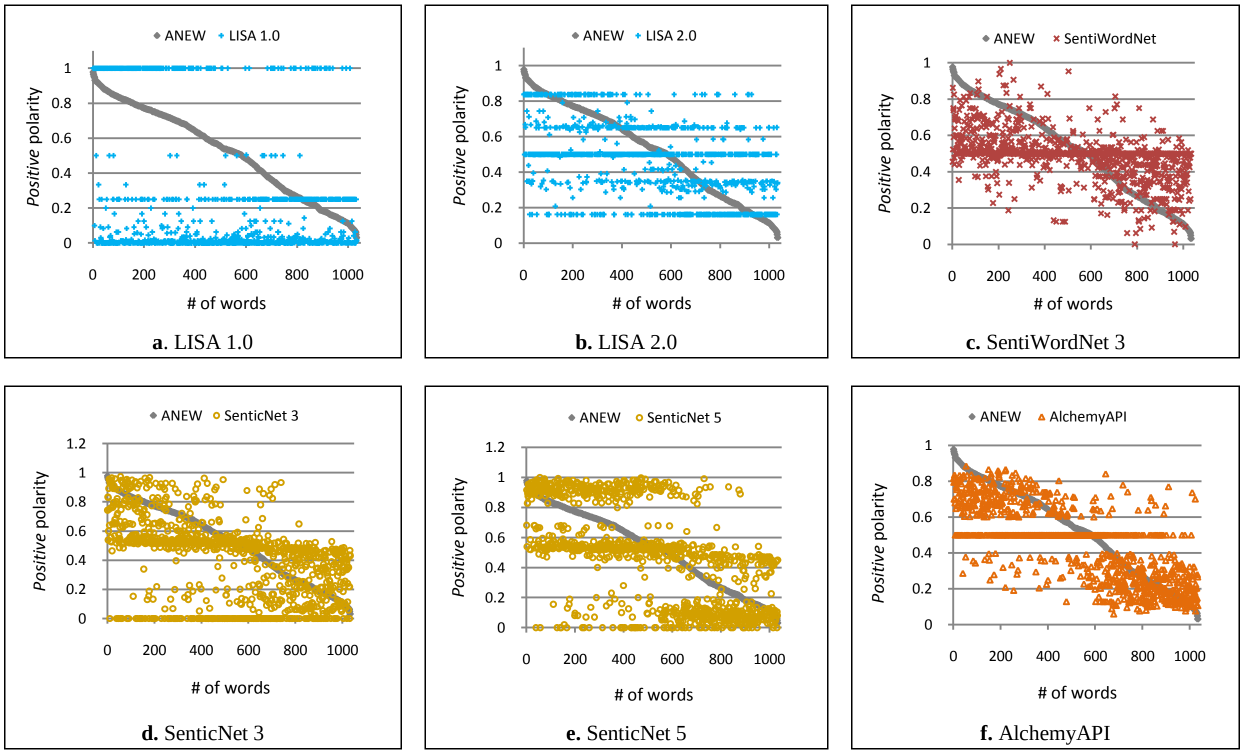
[16] *X* and *Y* are expected to have the same size, i.e., producing sentiment scores for the same number of words in the test dataset. Nonetheless, different sizes for *X* and *Y* can be accounted for in both *PCC* and *MSE* formulas, when rating LSA methods which do not produce sentiment scores for certain words. In this case, the scores of words missing from *X* (*Y*) are considered to be as different as possible from the available word scores in *Y* (*X*), such that x=1-y, and vice versa for *Y*, where x and y remain $\in$[0, 1]. The latter simulates the concept of "maximum distance" for an unprocessed word, which in our case will always be bounded in the [0, 1] interval.

[17] The values of *PCC* $\in$ [-1, 1] such that: -1 designates that one of the variables is a decreasing function of the other variable (i.e., words deemed sentiment-expressive by human testers are deemed non-expressive of that sentiment by the system, and vice versa), 1 designates that one of the variables is an increasing function of the other variable (i.e., words are deemed expressive/non-expressive of a given sentiment by human testers and the system alike), and 0 means that the variables are not correlated.

[18] SenticNet 5 is an improved version of SenticNet 3 which performs dimension reduction and multi-dimensional scaling of the lexicon's feature vector space, using specially designed recurrent neural networks to improve sentiment scoring (cf. Section 2).

[19] http://sigappfr.acm.org/Projects/LISA

relatively dispersed in the polarity space (show in Fig. c, d, and e). This reflects their supervised learning nature, which produces results that are varied and reflective of the diversity of their training data, compared with LISA's less dispersed and more rigorously structured (clustered) results, reflecting the structured nature of its LAG reference and graph computation process.



a. LISA 1.0 **b.** LISA 2.0 **c.** SentiWordNet 3 **d.** SenticNet 3 **e.** SenticNet 5 **f.** AlchemyAPI

Fig. 19. *Positive* polarity scores w.r.t. the ANEW dataset

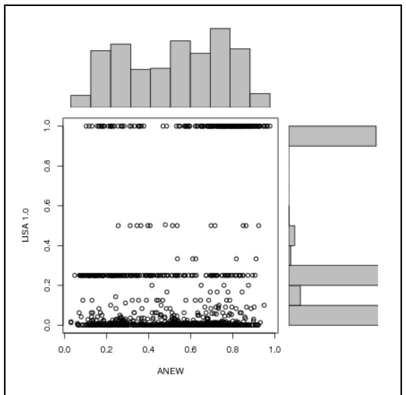
The above results are also highlighted in the *PCC* graphs in Fig. 20, where LISA 1.0 and 2.0 show a more structured result organization, compared with loosely structured and more diverse polarity scoring behavior with alternative supervised solutions.

Results compiled in Table 1 show that LISA 2.0's performance, in terms of both *PCC* and *MSE*, is on a par with existing (supervised learning) approaches, namely SentiWordNet and SenticNet 3. SenticNet 5, which performs dimension reduction and multi-dimensional scaling of SenticNet's feature vector space, produced improved results, highlighting the impact of feature vector dimensionality on LSA quality. IBM's AlchemyAPI opinion mining engine produced the best results, distinctively surpassing all other approaches including LISA.

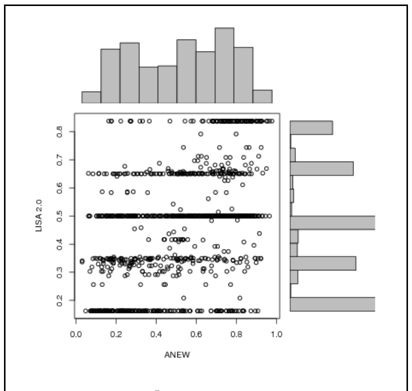
Table 1. Average *PCC*, *MSE*, and combined *closeness* scores for *positive* and *negative* polarity

	PCC			***MSE***			***Closeness*** [20]
	Positive	*Negative*	*Avg.*	*Positive*	*Negative*	*Avg.*	
AlchemyAPI	0.7476	0.7477	0.7477	0.0322	0.0321	0.0322	0.8578
SenticNet 5	0.6370	0.6249	0.63095	0.0368	0.0282	0.0325	0.7992
SentiWordNet 3	0.4934	0.4934	0.4934	0.0482	0.0482	0.0482	0.7226
LISA 2.0	0.4496	0.4496	0.4496	0.0605	0.0606	0.0606	0.6945
SenticNet 3	0.4504	0.4343	0.4424	0.0864	0.0807	0.0836	0.6794
LISA 1.0	0.2497	0.1872	0.2185	0.2247	0.2281	0.2264	0.4961

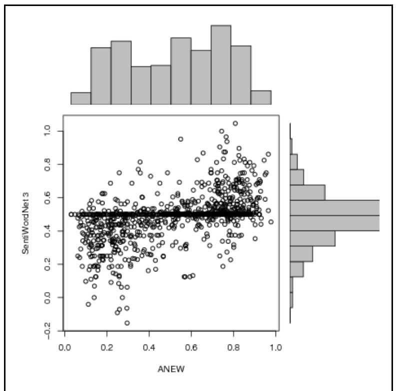
[20] We consider α = 0.5, assigning equal weight to *PCC* and *MSE*



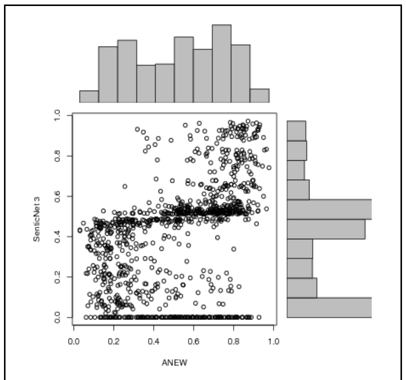
a. LISA 1.0



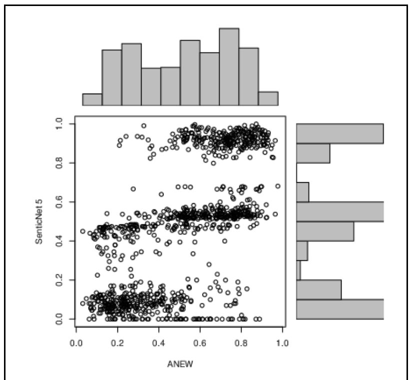
b. LISA 2.0



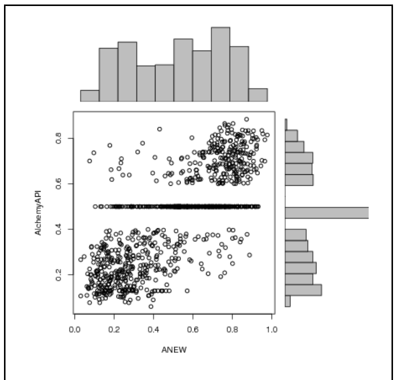
c. SentiWordNet 3



d. SenticNet 3



d. SenticNet 5



f. AlchemyAPI

Fig. 20. *PCC* graphs for *positive* polarity scores

4.5. Affect Evaluation Quality

We also compared LISA 1.0 and 2.0 with ANEW as well as two alternative methods available online: EmoSenticNet [88], and Tone Analyzer [54]. A snapshot of the results is provided in Fig. 21 and Fig. 22. The complete set of empirical graphs and corresponding data is provided online[21]. Results are summarized Table 2.

Fig. 21 shows *dislike/disgust* polarity scores w.r.t. ANEW, where words have been ranked following ANEW's *dislike* intensities (from highest to lowest). Fig. 22 provides corresponding *PCC* graphs. Similar graphs were produced for the other four affective categories (i.e., *happiness*, *anger*, *sadness*, *fear*) and are provided online. Results here reflect observations similar to the ones made earlier with polarity scores: i) LISA 2.0 produced results which are more evenly distributed along ANEW's ratings compared with LISA 1.0, ii) LISA 2.0's results show concentrations of score points around the ANEW score line, with clusters of points forming around *dislike* intensity scores = 0.67, 0.46, 0.32, 0.18, and 0.09 (highlighted in Fig. 21.b), iii) IBM's Tone Analyzer, which is a supervised learning solution, produced affect scores that are relatively dispersed in the affective space (cf. Fig. 21.d), compared with LISA's clustered results, reflecting the former's supervised learning nature and the diversity of the training data, iv) EmoSenticNet, which is a semi-supervised sentiment lexicon, produced discrete affect category labels (in the form of scores $\in\{0, 1\}$ for every affect category, where the score of a word that belongs to the category =1, otherwise, it is =0). It does not produce affective intensity levels as clearly reflected in the binary nature of its results (in Fig. 21.c).

Results compiled in Table 2 show that LISA 2.0's performance, in terms of both *PCC* and *MSE,* is on a par with existing (semi)supervised approaches. IBM's Tone Analyzer results, while more varied and dispersed than LISA's, slightly surpassed the latter's effectiveness w.r.t. the ANEW experimental dataset. This highlights LISA's potential as an unsupervised word-level LSA method capable of contending with existing supervised solutions. Nonetheless, we clarify that LISA only performs word-level analysis at this stage, while Tone Analyzer is capable of sentence and document-level analyses.

[21] http://sigappfr.acm.org/Projects/LISA

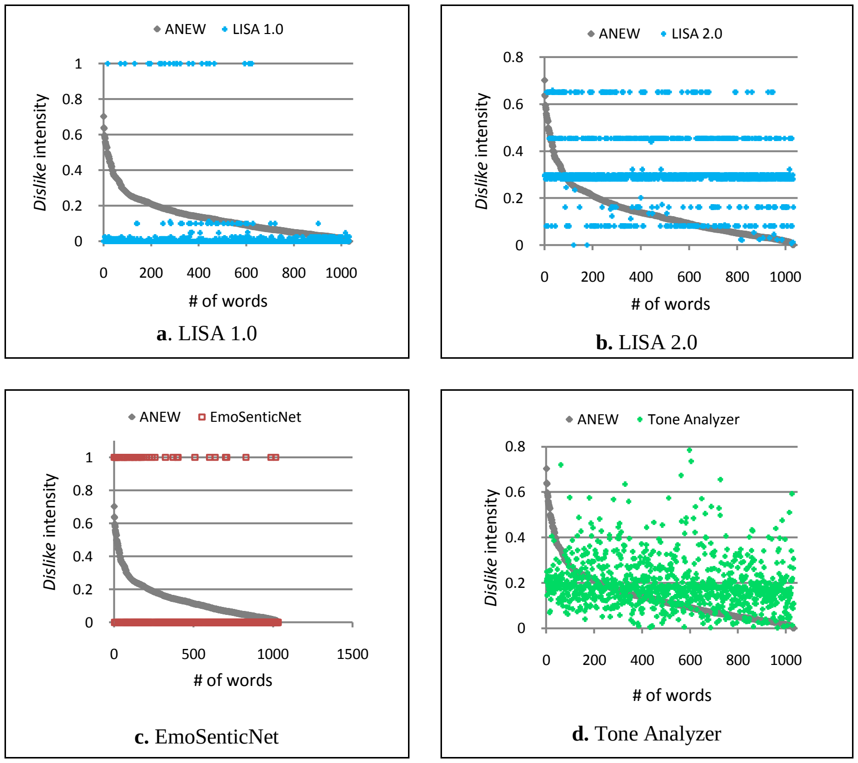


Fig. 21. *Dislike/disgust* affective scores w.r.t. the ANEW dataset

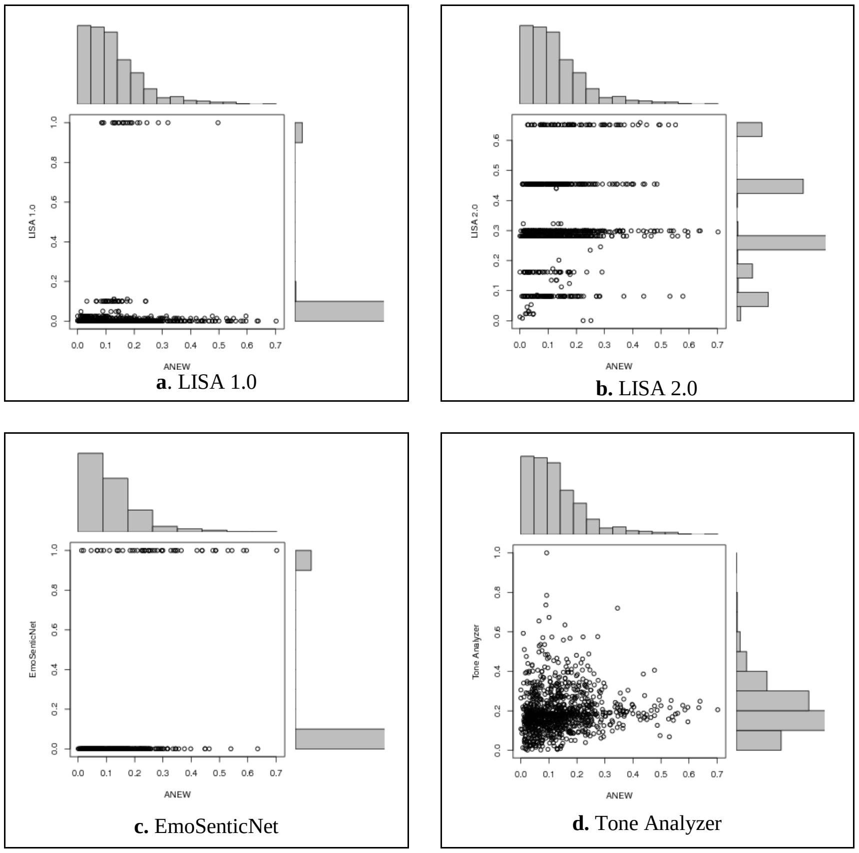


Fig. 22. *PCC* graphs for *dislike/disgust* affect scores

Table 2. Average *PCC*, *MSE*, and *closeness* scores for *happiness*, *anger*, *sadness*, *fear*, and *dislike/disgust* affective categories

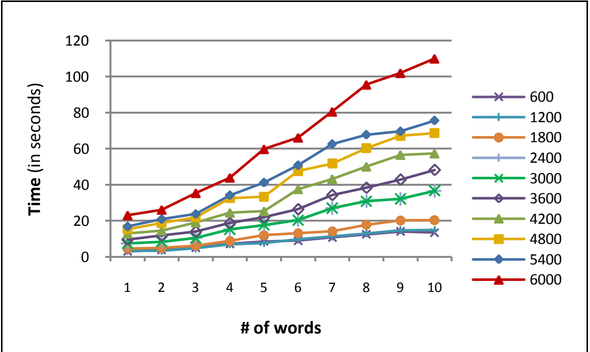
	PCC						***MSE***						***Closeness***
	Happiness	*Anger*	*Sadness*	*Fear*	*Dislike*	*Avg.*	*Happiness*	*Anger*	*Sadness*	*Fear*	*Dislike*	*Avg.*	
Tone Analyzer	0.1997	0.1488	0.1299	0.0756	0.1513	0.14106	0.1146	0.0458	0.0543	0.0458	0.0542	0.06294	0.60959
LISA 2.0	0.2251	0.1667	0.0108	0.0807	0.1669	0.13004	0.1935	0.0293	0.0609	0.0269	0.0608	0.07428	0.5929
EmoSenticNet	0.1512	-0.0369	0.0394	0.0838	0.0671	0.06092	0.2932	0.2943	0.3208	0.2849	0.3207	0.30278	0.40953
LISA 1.0	0.1257	0.0697	0.0045	0.0338	0.0698	0.0607	0.2322	0.0343	0.2292	0.0404	0.2286	0.15294	0.48423

We are currently investigating phrase-level and sentence-level LSA, combining LISA's functionality with additional features such as word associations, valence shifters, and a dedicated emoji affect lexicon [40], to perform unsupervised LSA on short social media texts (such as Facebook comments and Twitter messages).

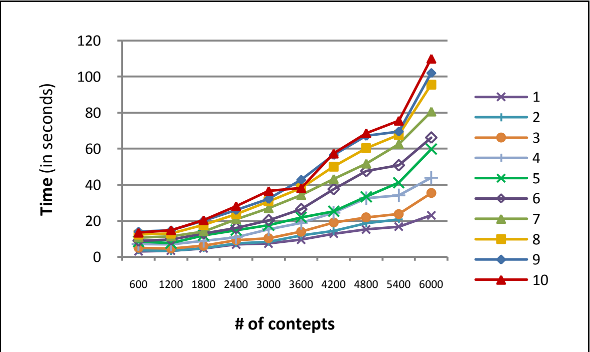
4.6. Time Performance

We also conducted efficiency tests to assess the running times of both LISA 1.0 and 2.0 modules. Following our complexity analysis in Section 3.4, LISA 1.0's time complexity simplifies to $O(|w|\times|G^{*}|^{2})$ where $|w|$ represents the number of input words to be processed, and $|G^{*}|$ the number of navigated nodes (concepts) in the LAG (lexical affective graph) G processed before reaching the destination affect nodes. As for LISA 2.0, its *affect propagation* process in the LAG (generating the *sentiment lexicon*) requires $O(|G|^{2})$, whereas its *affect lookup* process (acquiring affect scores from the generated *sentiment lexicon*) simplifies to $O(|w|\times log(|G|))$.

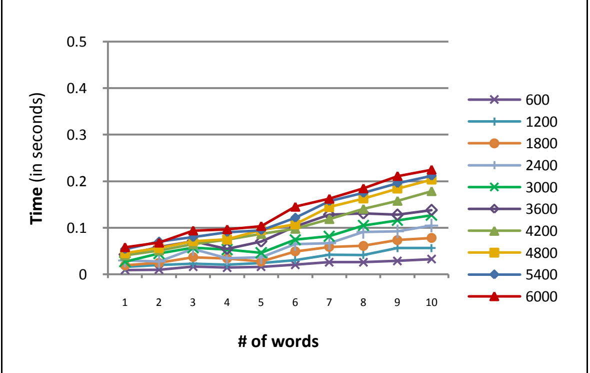
We varied the number of user input words, as well as the size of the LAG by generating different extracts w.r.t. its total size (considering for instance: 10%, 20%, …, or 100% of the LAG). Every extract contained all affective nodes to allow LSA processing. The total size of our LAG (mapping WordNet 3.0 with WNAH) was around 30 Mbytes, including more than 117k concepts. The characteristics of a set of sample LAG chunks used in our experiments are summarized in Table 3. Experiments were carried out on an HP ProLiant ML350 Generation 5 (G5) Dual-Core Intel ® XeonTM 5000 processor with 2.66 GHz processing speed and 16 GB of RAM.



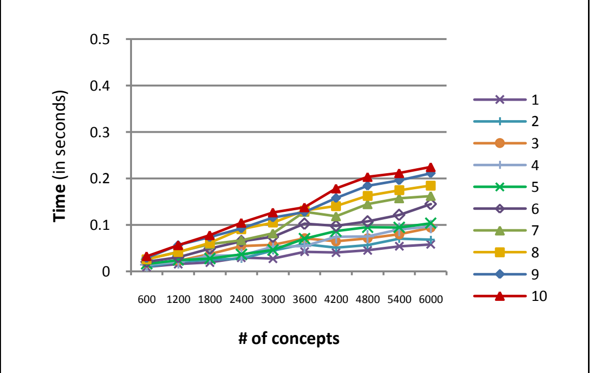
a. LISA 1.0, varying the number of inout source words



b. LISA 1.0, varying the number of concepts in the LAG



c. LISA 2.0, varying the number of input source words

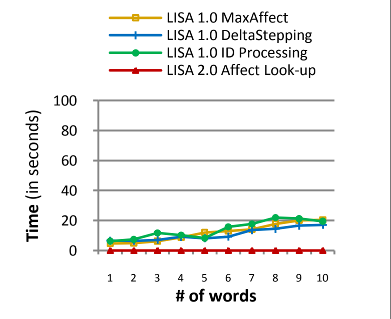


d. LISA 2.0, varying the number of concepts in the LAG

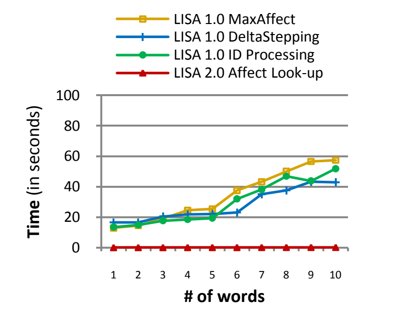
Fig. 23. Time performance w.r.t. the number of input words, and the number of concepts processed in the LAG

Table 3. Characteristics of sample LAG chunks utilized in our experiments

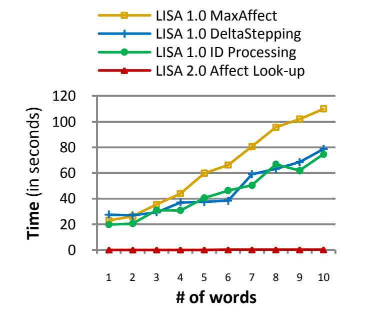
N# of Concepts (Synsets)	11,738	23,475	35,212	46,949	58,686	70,423	82,160	93,897	105,634	117,659
Chunk %	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Size (in MBs)	2.7707	3.9466	7.6498	9.5691	12.1641	13.8941	18.2191	19.9491	23.4091	26.0041
Avg. Branch[22]	1.4533	1.6257	1.7553	1.9236	2.0697	2.2259	2.3736	2.5285	2.6677	2.8223
Avg. Span[23]	2.1035	2.2299	2.3665	2.5213	2.8849	3.5362	3.7411	4.1947	5.9852	7.5119



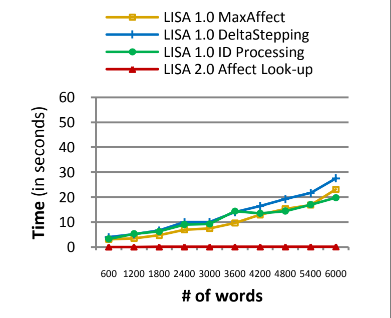
a. Number of navigated concepts = 1800



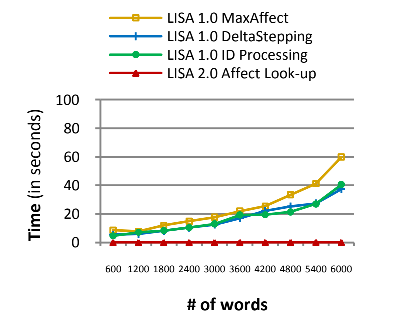
b. Number of navigated concepts = 4200



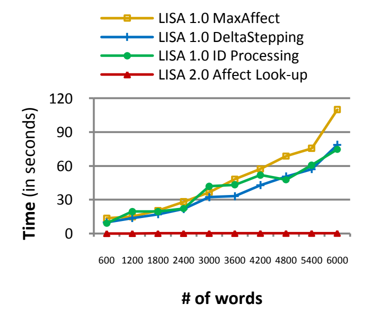
c. Number of navigated concepts = 6000



a. Number of input words = 1



b. Number of input words = 5



c. Number of input words = 10

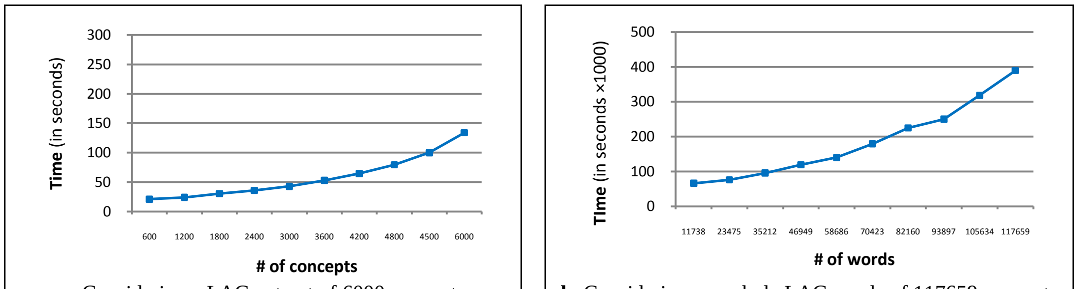
Fig. 24. Comparing the time performance of LISA 1.0 and LISA 2.0 (*affect lookup*)

Results in Fig. 23 and Fig. 24 show sentiment analysis time w.r.t. the number of user input words and the number of navigated concepts in the LAG, comparing LISA 1.0's main navigation process, with LISA 2.0's affect lookup process, as well as two variants of LISA 1.0: i) LISA 1.0_*IDProcessing*, an enhanced version of LISA 1.0 which allows for faster LAG navigation by processing the concepts' IDs from the WordNet KB (cf. Appendix I), and ii) LISA 1.0_*DeltaStepping*, a parallelized version of LISA 1.0 following the Delta Stepping shortest path parallel processing paradigm (cf. Appendix I).

Results in Fig. 23.a and c highlight LISA 1.0 and LISA 2.0 (*affect lookup*)'s linear dependencies on the number of input words processed for sentiment analysis. Fig. 23.b and d respectively show the latter's quadratic and logarithmic dependencies on the number of concepts processed in the LAG. Comparative time results in Fig. 24 clearly show the major difference between LISA 1.0's execution time, including its *DeltaStepping* and *IDProcessing* variants on the one hand, and LISA 2.0's *affect lookup* time on the other hand. LISA 2.0's *affect lookup* process requires on average 0.18 ‰ of LISA 1.0's time to produce affective scores.

Time results in Fig. 25 highlight LISA 2.0 *affective propagation*'s quadratic dependency on the size (in number of concepts) of the LAG. It took around 4 days to process our complete LAG which maps the whole of WordNet with WNAH (cf. Fig. 25.b). Note that LISA 2.0's *affective propagation* is done offline prior to performing user LSA operations (which are handled by LISA 2.0's *affect lookup* process).

[22] Average number of outgoing edges per node, i.e., node fan-out.
[23] Length of the path from a root (most abstract) concept node to a leaf (most specific) node in the LAG (considering hierarchical relations only, e.g., *hypernymy*/*hyponymy*, to avoid loops).



a. Considering a LAG extract of 6000 concepts **b.** Considering our whole LAG made of 117659 concepts

Fig. 25. Time performance of LISA 2.0 *affective propagation*

5. Conclusion

This paper introduces LISA, an unsupervised word-level knowledge graph-based LSA framework, which uses different variants of shortest path graph navigation techniques to compute and propagate affective scores in a lexical-affective graph (LAG). LISA was designed in two consecutive iterations, producing two modules: i) LISA 1.0 for affect navigation, and ii) LISA 2.0 for affect propagation and lookup. LISA 1.0 suffered from the semantic connectivity problem shared by some existing lexicon-based methods, and required polynomial execution time. This led to the development of LISA 2.0, which i) processes affective relationships separately from lexical/semantic connections (solving the semantic connectivity problem of LISA 1.0), and ii) produces a *sentiment lexicon* which can be searched in logarithmic time to perform word-level LSA (handling LISA 1.0's efficiency problem). Experiments on the ANEW dataset show that LISA 2.0 outperforms LISA 1.0 in both LSA quality and performance, while being on a par with existing (semi)supervised approaches (without the need for training or manual effort).

We are currently investigating phrase-level and sentence-level LSA, combining LISA's functionality with additional features such as word associations, valence shifters, and a dedicated emoji affect lexicon [40], to perform unsupervised LSA on short social media texts (such as Facebook comments and Twitter messages). We are also considering author stylistic features [131] including different text writing attributes which can emphasize sentiments, such as i) *punctuation* (e.g., "!", "-", "?") which can intensify or weaken a sentiment, iii) *abbreviations* (e.g., "US", "UK" instead of mentioning the full names of countries) which, coined with context features, might highlight familiarity and thus intensity certain sentiments, and iii) *unusual word spellings* such as the use of capital case letters (e.g., "COOL") as well as character repetitions (e.g., "greeeeat") which can intensify sentiments [42]. We are also investigating the extension of our LAG structure to consider unambiguous word concepts mined form concept glosses in the lexical KB, following the intuition that the gloss of a concept describing an affect category would contain words that might be related (with different intensities) to that category [6]. In the near future, we aim to explore *implicit semantics* (a.k.a. *latent semantics*) [107], i.e., semantics which can be inferred from the statistical analysis of word/phrase embeddings (feature vectors), following their co-occurrence in a corpus [50]. This is different from conventional concept-based LSA, which utilizes *explicit concepts* representing *real-life* entities (e.g., concepts within a conventional KB such as WordNet or Wikipedia) [118]. Latent feature representations, combined with dimension reduction and scaling [16], allow capturing certain syntactic and semantic regularities and relationships between words and phrases represented as feature vectors [50] (e.g., identifying that "failure" is *related to* "sadness" following their feature vector offsets), which we aim to investigate and possibly combine with our LAG structure toward unsupervised knowledge-based and corpus-based LSA.

Acknowledgements

This study is partly funded by the National Council for Scientific Research - Lebanon (CNRS-L), by the Lebanese American University (LAU), and by the Fulbright Visiting Scholar program (sponsored by the US Department of State). Special thanks go to LAU computer engineering graduates Ms. Souad Charafeddine and Mr. Ralph Abboud who partly contributed to LISA's implementation.

References

[1] Abbasi A., Chen H., Thoms S., and Fu T., *Affect Analysis of Web Forums and Blogs Using Correlation Ensembles.* IEEE Transactions on Knowledge and Data Engineering, 2008. 20(9):1168-1180.

[2] Abbasi A., France S.L., Zhang Z., and Chen H., *Selecting Attributes for Sentiment Classification using Feature Relation Networks.* IEEE Trans. Knowl. Data Eng., 2011. 23(3): 447-462.

[3] Akhtar M.S., Gupta D.K., Ekbal A., Bhattacharyya P.,, *Feature Selection and Ensemble Construction: A Two-step Method for Aspect based Sentiment Analysis.* Knowledge-based Systems, 2017. 125: 116-135.

[4] Almatarneh S. and Gamallo P., *Automatic Construction of Domain-Specific Sentiment Lexicons for Polarity Classification.* Practical Applications of Agents and Multi-Agent Systems (PAAMS'2017), 2017. pp. 175-182.

[5] Appel O., Chiclana F., Carter J., and Fujita H., *A Hybrid Approach to the Sentiment Analysis Problem at the Sentence Level.* Knowledge-Based Systems, 2016. 108: 110-124.

[6] Baccianella S., Esuli A., and Sebastiani F., *SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining.* Language Resources and Evaluation (LREC'10), 2010. pp. 2200-2004.

[7] Balahur A., Hermida J.M., and Montoyo A., *Detecting Implicit Expressions of Emotion in Text: a Comparative Analysis* Decis. Support Syst. , 2012. 53:742–753.

[8] Bautin M., Vijayarenu L., and Skiena S., *International Sentiment Analysis for News and Blogs.* International Conference on Weblogs and Social Media (ICWSM'08), 2008. pp. 19-26.

[9] Bizer C., Lehmann J., Kobilarov G., Auer S., Becker C., Cyganiak R., and Hellmann S., *DBpedia – a crystallization point for the web of data.* Elsevier Journal of Web Semantics (JWS), 2009. 7:154–165.

[10] Bradley M. and Lang P., *Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings.* Technical Report C-1, 1999. University of Florida: Center for Research in Psychophysiology.

[11] Cambria E., Fu J., Bisio F., Poria S., *AffectiveSpace 2: Enabling Affective Intuition for Concept-level Sentiment Analysis.* AAAI Conference on Artificial Intelligence (AAAI'15), 2015. pp. 508–514.

[12] Cambria E., Cambria E., Gastaldo P., Federica Bisio, Rodolfo Zunino:, *An ELM-based Model for Affective Analogical Reasoning.* Neurocomputing, 2015. 149: 443–455.

[13] Cambria E., Havasi C., and Hussain A., *SenticNet 2: a Semantic and Affective Resource for Opinion Mining and Sentiment Analysis.* Proc. 25th Int'l Florida Artificial Intelligence Research Society Conf. (AAAI'12) 2012. pp. 202–207.

[14] Cambria E., Olsher D., and Rajagopal D., *SenticNet 3: A Common and Common-Sense Knowledge Base for Cognition-Driven Sentiment Analysis.* AAAI Conference on Artificial Intelligence (AAAI'14) 2014. pp. 1515-1521.

[15] Cambria E., Poria S., Bajpai R., and Schuller B.W., *SenticNet 4: A Semantic Resource for Sentiment Analysis Based on Conceptual Primitives.* International Conference on Computational Linguistics 5COLING'16) 2016. pp. 2666-2677.

[16] Cambria E., Poria S., Hazarika D., and Kwok K., *SenticNet 5: Discovering Conceptual Primitives for Sentiment Analysis by Means of Context Embeddings.* Association for the Advancement of Artificial Intelligence (AAAI'18) 2018. pp. 1795-1802.

[17] Cambria E., Speer R., Havasi C., and Hussain A., *SenticNet: A Publicly Available Semantic Resource for Opinion Mining.* Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium: Commonsense Knowledge 2010.

[18] Charbel N., Joe Tekli Richard Chbeir, and Gilbert Tekli, *Resolving XML Semantic Ambiguity.* International Conference on Extending Database Technology (EDBT'15), 2015. Brussels, Belgium, pp 277-288.

[19] Cho H., Kim S., Lee J., and Lee J.S., *Data-Driven Integration of Multiple Sentiment Dictionaries for Lexicon-based Sentiment Classification of Product Reviews.* Knowl.-Based Syst., 2014. 71: 61-71.

[20] Chuang Z. and Wu C., *Multi-Modal Emotion Recognition from Speech and Text.* Computational Linguistics and Chinese Language Processing, 2004. 9(2): 45-62.

[21] Cormen T.H., Leiserson C.E., Rivest R.L., and Stein C., *Introduction to Algorithms (3rd Ed.).* MIT Press and McGraw-Hill. , 2009.

[22] Cowie R., Esposito A.M., & Vogel C., *Emotion Recognition in Human-Computer Interaction.* In IEEE Sig. Proc. Mag., 2001. 18(1):32-80.

[23] da Silva N., Coletta L., and Hruschka E., *A Survey and Comparative Study of Tweet Sentiment Analysis via Semi-Supervised Learning.* ACM Comput. Surv. , 2016. 49(1): 15:1-15:26.

[24] de Albornoz J., Plaza L., and Gervás P., *SentiSense: An easily scalable concept-based affective lexicon for sentiment analysis.* Language Resources and Evaluation (LREC'12) 2012. pp. 3562-3567.

[25] Demartini G. and Siersdorfer S., *Dear Search Engine: What's your opinion about...? Sentiment Analysis for Semantic Enrichment of Web Search Results.* International World Wide Web Conference (WWW'10), 2010. p. 7.

[26] Donath J., Karahalio K., and Viegas F., *Visualizing Conversation.* J. Computer-Mediated Communication (1999), 1999. 4(4).

[27] Dragoni M. and Petrucci G., *A Neural Word Embeddings Approach for Multi-Domain Sentiment Analysis.* IEEE Trans. Affective Computing, 2017. 8(4): 457-470.

[28] Dragoni M., Poria S., and Cambria E., *OntoSenticNet: A Commonsense Ontology for Sentiment Analysis.* IEEE Intelligent Systems, 2018. 33(3): 77-85.

[29] Ebrahimi M., Yazdavar A.H., and Sheth A.P., *Challenges of Sentiment Analysis for Dynamic Events.* IEEE Intelligent Systems, 2017. 32(5):70-75.

[30] Eliaçik A.B. and Erdogan N., *Influential User Weighted Sentiment Analysis on Topic based Microblogging Community.* Expert Syst. Appl. 92: 403-418, 2018.

[31] Elmasri R. and Navathe S.B., *Fundamentals of Database Systems (6th ed.).* Upper Saddle River, N.J.: Pearson Education, 2010, 652–660.

[32] Esparza G.G. de Luna A., Ochoa-Zezzatti A., Hernandez A., Ponce J., Álvarez M., Cossio E., and de Jesus Nava J., *A Sentiment Analysis Model to Analyze Students Reviews of Teacher Performance Using Support Vector Machines.* Distributed Computing and Artificial Intelligence (DCAI'17) 2017. pp. 157-164.

[33] Esuli A. and Sebastiani F., *PageRanking WordNet Synsets: An Application to Opinion Mining.* Meeting of the Association for Computational Linguistics (ACL'07), 2007. pp. 424-431.

[34] Fahrni A and Klenner M., *Old Wine or Warm Beer: Target-Specific Sentiment Analysis of Adjectives.* Symposium on Affective Language in Human and Machine (AISB'08), 2008. pp. 60-63.

[35] Ferilli S., De Carolis B., Redavid D., and Esposito F., *Towards Sentiment and Emotion Analysis of User Feedback for Digital Libraries.* Italian Research Conference on Digital Libraries (IRCDL'16), 2016. pp. 137-149.

[36] Francis W. N. and Kucera H., *Frequency Analysis of English Usage.* Houghton Mifflin, Boston, 1982.

[37] Francisco V., Gervás O., and Peinado F., *Ontological Reasoning for Improving the Treatment of Emotions in Text.* Knowl. Inf. Syst. , 2010. 25(3): 421-443.

[38] Fu X., Liu W., Xu Y., and Cui L., *Combine HowNet Lexicon to Train Phrase Recursive Autoencoder for Sentence-Level Sentiment Analysis*, Neurocomputing 241: 18-27,

[39] Ganesan K. and Zhai C.X., *Opinion-based Entity Ranking.* Information Retrrieval 2012. 15(2): 116-150.

[40] Gavilanes M.F., Juncal-Martínez J., García-Méndez S., Costa-Montenegro E., and González-Castaño F.J., *Creating Emoji Lexica from Unsupervised Sentiment Analysis of their Descriptions.* Expert Syst. Appl., 2018. 103: 74-91.

[41] Ghiassi M. and Lee S., *A Domain Transferable Lexicon Set for Twitter Sentiment Analysis using a Supervised Machine Learning Approach.* Expert Syst. Appl. , 2018. 106: 197-216.

[42] Giachanou A. & Crestani F., *Like It or Not: A Survey of Twitter Sentiment Analysis Methods.* ACM Comput. Surv., 2016. 49(2): 28:1-28:41

[43] Gill A.J., Robert M. French, Darren Gergle, and Jon Oberlander, *Identifying Emotional Characteristics from Short Blog Texts.* Proceedings for the 30th Annual Meeting of the Cognitive Science Society, 2008. pp. 2237-2242.

[44] Godbole N., Srinivasaiah M., and Skiena S., *Large-Scale Sentiment Analysis for News and Blogs.* International Conference on Weblogs and Social Media (ICWSM'07), 2007. p. 4.

[45] Grefenstette G., Yan Qu, David A. Evans, and James G. Shanahan:, *Validating the Coverage of Lexical Resources for Affect Analysis and Automatically Classifying New Words Along Semantic Axes.* Proc. AAAI Spring Symp. Exploring Attitude and Affect in Text: Theories and Applications (AAAI-EAAT '04), Y. Qu, J. Shanahan, and J. Wiebe, eds., 2004. pp. 71-78.

[46] Grefenstette G., Qu Y., Shanahan J.G., and Evans D.A., *Coupling Niche Browsers and Affect Analysis for an Opinion Mining Application,".* Proc. of 12th Inter. Conf. Recherche d'Information Assistee par Ordinateur (RIAO '04), 2004. pp. 186-194.

[47] Hancock J.T., Landrigan C., and Silver C., *Expressing Emotion in Text-based Communication.* Proc. of ACM Conference on Human Factors in Computing Systems (CHI'07) 2007. pp. 929-932.

[48] Hatzivassiloglou V. and McKeown K.R., *Predicting the Semantic Orientation of Adjectives.* Meeting of the Association for Computational Linguistics (ACL'97), 1997. pp. 174-181.

[49] Hearst M.A., *Direction-Based Text Interpretation as an Information Access Refinement.* Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval 1992. P. Jacobs, ed., Lawrence Erlbaum Assoc.

[50] Hill F., Kyunghyun C., and Korhonen A., *Learning Distributed Representations of Sentences from Unlabelled Data.* Inter. Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016. pp. 1367-1377.

[51] Hoffart J., Suchanek F.M., Berberich K., and Weikum G., *YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia.* Artif. Intell., 2013. 194: 28-61.

[52] Hovy, E., *What are Sentiment, Affect, and Emotion? Applying the Methodology of Michael Zock to Sentiment Analysis.* N. Gala et al. (eds.), Language Production, Cognition, and the Lexicon, Text, Speech and Language Technology 48, 2015. pp. 13-24.

[53] IBM, *AlchemyAPI.* 2005. Available at https://www.ibm.com/watson/alchemy-api.html (accessed June 2018).

[54] IBM, *Tone Analyzer.* Accessed June 2018. Available at: https://www.ibm.com/watson/services/tone-analyzer/.

[55] Jaggi M., Uzdilli F., and Cieliebak M., *Swiss-Chocolate: Sentiment Detection using Sparse SVMs and Part-Of-Speech n-Grams.* SemEval at International Conference on Computational Linguistics (COLING'14), 2014. pp. 601-604.

[56] Jiménez F.V., Gelbukh A.F., and Sidorov G., *Simple Window Selection Strategies for the Simplified Lesk Algorithm for Word Sense Disambiguation.* Mexican International Conference on Artificial Intelligence (MICAI'13), 2013. pp. 217-227.

[57] Kamps J., Marx M., Mokken R.J., and de Rijke M., *Using WordNet to Measure Semantic Orientations of Adjectives.* Language Resources and Evaluation (LREC'04), 2004. p. 4.

[58] Kang H., Yoo S.J., and Han D., *Senti-Lexicon and Improved Naïve Bayes Algorithms for Sentiment Analysis of Restaurant Reviews* Expert Syst. Appl., 2012. 39:6000–6010.

[59] Khan E.F., Qamar U., and Bashir S., *A Semi-Supervised Approach to Sentiment Analysis using Revised Sentiment Strength based on SentiWordNet.* Knowl. Inf. Syst. , 2017. 51(3): 851-872.

[60] Khan F.H., Qamar U., and Bashir S., *Lexicon-based Semantic Detection of Sentiments using Expected Likelihood Estimate Smoothed Odds Ratio.* Artif. Intell. Rev., 2017. 48(1): 113-138

[61] Kim S. and Hovy E., *Determining the Sentiment of Opinions.* Proc. 20th Int'l Conf. Computational Linguistics (COLING '04), 2004. pp. 1367-1373.

[62] Koc S.S., et al., *Triadic Co-Clustering of Users, Issues and Sentiments in Political Tweets.* Expert Syst. Appl., 2018. 100: 79-94.

[63] Korayem M., AlJadda K., and Crandall D.J., *Sentiment/subjectivity analysis survey for languages other than English.* Social Netw. Analys. Mining, 2016. 6(1): 75:1-75:17.

[64] Lang P. J., *Behavioral Treatment and Bio-Behavioral Assessment: Computer Applications.* In J. B. Sidowski, J. H. Johnson, & T. A. Williams (Eds.), Technology in Mental Health Care Delivery Systems. Norwood, NJ Ablex Publishing., 1980. pp. 119-137.

[65] Lee G., Jeong J., Seo S., Kim C.Y., and Kang P., *Sentiment Classification with Word Localization based on Weakly Supervised Learning with a Convolutional Neural Network* Knowl.-Based Syst. , 2018. 152: 70-82.

[66] Li S., Hao J., Jiang Y., and Jing Q., *Exploiting Co-occurrence Opinion Words for Semi-supervised Sentiment Classification.* Advanced Data Mining and Applications (ADMA'13), 2013. pp. 36-47.

[67] Liu H., Lieberman H., and Selker T., *A Model of Textual Affect Sensing Using Real-World Knowledge.* Proc. Eighth Int'l Conf. Intelligent User Interfaces (IUI'03), 2003. pp. 125-132.

[68] Lloyd L., Kechagias D., and Skiena S., *Lydia: A System for Large-Scale News Analysis.* String Processing and Information Retrieval (SPIRE'05) 2005. pp. 161-166.

[69] Ma C., Wang M., and Chen X., *Topic and Sentiment Unification Maximum Entropy Model for Online Review Analysis.* International World Wide Web Conferences (WWW'15), 2015. pp. 649-654.

[70] Martín-Wanton T., et al., *Opinion Polarity Detection - Using Word Sense Disambiguation to Determine the Polarity of Opinions.* International Conference on Agents and Artificial Intelligence (ICAART'10), 2010. 1: 483-486.

[71] Meyer U. and Sanders P., *[Delta]-Stepping: a Parallelizable Shortest Path Algorithm.* J. Algorithms 2003. 49(1): 114-152.

[72] Miller G., *WordNet: An On-Line Lexical Database.* International Journal of Lexicography, 1990. 3(4).

[73] Miller G.A. and Fellbaum C., *WordNet Then and Now.* Language Resources and Evaluation, 2007. 41(2): 209-214.

[74] Mishne G., *Experiments with Mood Classification.* First Workshop on Stylistic Analysis of Text for Info. Access, 2005. p. 8.

[75] Mishne G. and de Rijke M., *Capturing Global Mood Levels Using Blog Posts.* Proc. AAAI Spring Symp. Computational Approaches to Analysing Weblogs (AAAI-CAAW), 2006. pp. 145-152.

[76] Moghaddam S., *Beyond Sentiment Analysis: Mining Defects and Improvements from Customer Feedback.* European Conference on Information Retrieval (ECIR'15), 2015. pp. 400-410.

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

[78] Mohammad S.M. and Turney P.D., *Emotions Evoked by CommonWords and Phrases: Using Mechanical Turk to Create an Emotion Lexicon.* NAACL HLT 2010 Workshop on Computational Approaches to Analysis & Generation of Emotion in Text, 2010. pp. 26–34.

[79] Mondal A., Cambria E., Das D., and Bandyopadhyay S., *Employing sentiment-based affinity and gravity scores to identify relations of medical concepts.* IEEE Symposium Series on Computational Intelligence (SSCI'17), 2017. pp. 1-7.

[80] Mullen T. and Collier N., *Sentiment Analysis using Support Vector Machines with Diverse Information Sources.* Empirical Methods in Natural Language Processing (EMNLP'04), 2004. pp. 412–418.

[81] Nasukawa T. and Yi J., *Sentiment Analysis: Capturing Favorability using Natural Language Processing.* Proceedings of the 2nd ACM International Conference On Knowledge Capture, 2003. pp. 70–77.

[82] Neviarouskaya A., Prendinger H., and Ishizuka M., *Textual Affect Sensing for Sociable and Expressive Online Communication.* Affective Computing and Intelligent Interaction (ACII'07) 2007. pp. 218-229.

[83] Pang B. and Lee L., *Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales.* Proc. Ann. Meeting on Assoc. for Computational Linguistics (ACL '05), 2005. pp. 115-124.

[84] Parrot, W.G., *Emotions in Social Psychology: Key Readings (Key Readings in Social Psychology).* Psychology Press, 1st Ed., 2001. p. 392.

[85] Pennebaker J., Booth R., and Francias M., *Linguistic Inquiry and Word Count: LIWC.* LIWC Manual, 2007. p. 22.

[86] Pham D.H. and Le A.C., *Learning Multiple Layers of Knowledge Representation for Aspect-based Sentiment Analysis.* Data and Knowledge Engineering, 2018. 114: 26-39.

[87] Poria S., et al., *Enriching SenticNet Polarity Scores through Semi-Supervised Fuzzy Clustering.* IEEE Inter. Conf. on Data Mining (ICDM'12), 2012. pp. 709-716.

[88] Poria S., Gelbukh A.F., Hussain A., Howard N., Das D., and Bandyopadhyay S., *Enhanced SenticNet with Affective Labels for Concept-based Opinion Mining.* IEEE Intelligent Systems, 2013. 28(2): 31–38.

[89] Poria S., et al., *EmoSenticSpace: A Novel Framework for Affective Common-Sense Reasoning.* Knowl.-Based Syst. , 2014. 69: 108-123.

[90] Prabowo R. and Thelwall M., *Sentiment Analysis: a Combined Approach.* J. Informet. , 2009. 3:143–157.

[91] Rao Y., *Contextual Sentiment Topic Model for Adaptive Social Emotion Classification.* IEEE Intelligent Systems, 2016. 31(1): 41-47.

[92] Rao Y., et al., *Intensive Maximum Entropy Model for Sentiment Classification of Short Text.* Database Systems for Advanced Applications (DASFAA'15) Workshops, 2015. pp. 42-51.

[93] Ravi K. and Ravi V., *A survey on opinion mining and sentiment analysis: Tasks, approaches and applications.* Knowledge-Based Systems, 2015. 89:14–46.

[94] Razon B. and C. C., *Word Sense Disambiguation of Opinionated Words using Extended Gloss Overlap.* Proc. of the 8th National Natural Language Processing Research Symposium, 2011. pp. 1-5.

[95] Scherer K.R., *What are Emotions? And how can they be Measured?* Social Science Information, 2005. 44(4):693–727.

[96] Schouten K., et al., *Supervised and Unsupervised Aspect Category Detection for Sentiment Analysis with Co-occurrence Data.* IEEE Trans. Cybernetics, 2018. 48(4): 1263-1275.

[97] Severyn A. and Moschitti A., *On the Automatic Learning of Sentiment Lexicons.* North American Chapter of the Association for Computational Linguistics (HLT-NAACL'15), 2015. pp. 1397-1402.

[98] Sindhwani V. and Melville P., *Document-word Co-Regularization for Semisupervised Sentiment Analysis.* 8th IEEE International Conference on Data Mining (ICDM'08), 2008. pp. 1025–1030.

[99] Singh P., et al., *Commonsense: Knowledge Acquisition from the General Public.* Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems (ODBASE'02), 2002. p. 7.

[100] Stevenson R.A., Mikels J.A., and James T.W., *Manual for the Categorization of the Affective Norms for English Words (ANEW) by Discrete Emotions.* Indiana University Bloomington, 2014. p. 123.

[101] Stone P.J. and Hunt E.B. , *A Computer Qpproach to Content Qnalysis: Studies using the General Inquirer System.* Proceedings of the Spring Joint Computer Conference (AFIPS'63), 1963. pp. 241–256.

[102] Strapparava C. and Valitutti A., *WordNet Affect: an Affective Extension of WordNet.* Language Resources and Evaluation (LREC'04), 2004.

[103] Strapparava C., Valitutti A., & Stock O., *The Affective Weight of Lexicon.* Language Resources and Evaluation (LREC'06), 2006, 423-426.

[104] Subasic P. and Huettner A., *Affect Analysis of Text Using Fuzzy Semantic Typing.* IEEE Trans. Fuzzy Systems, 2001. 9(4):483-496.

[105] Taboada M., et al., *Lexicon-Based Methods for Sentiment Analysis.* Computational Linguistics, 2011. 37(2): 267-307.

[106] Taddesse F.G., et al., *Semantic-based Merging of RSS Items.* World Wide Web Journal: Internet and Web Information Systems Journal Special Issue: Human-Centered Web Science., 2010. Vol. 12 (No. 11280), Springer Netherlands.

[107] Tekli J., *An Overview on XML Semantic Disambiguation from Unstructured Text to Semi-Structured Data: Background, Applications, and Ongoing Challenges.* IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), 2016. 28(6): 1383-1407.

[108] Tekli J., Charbel N., and Chbeir R., *Building Semantic Trees from XML Documents.* Elsevier Journal of Web Semantics (JWS): Science, Services and Agents on the World Wide Web, 2016. 37–38:1–24.

[109] Tetko I., Livingstone D., & Luik A., *Comparison of Overfitting and Overtraining.* J. Chem. Inf. Comput. Sci., 1995. 35 (5): 826–833.

[110] Trainor K.J., Andzulis J., Rapp A., Agnihotri R., *Social Media Technology Usage and Customer Relationship Performance: a Capabilities-based Examination of Social CRM.* J. Bus. Res., 2013. 67(6): 1201-1208.

[111] Tsirakis N., et al., *Large Scale Opinion Mining for Social News and Blog Data.* Journal of Systems and Software, 2017. 127: 237-248.

[112] Turney P.D., *Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews.* Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, 2002. pp. 417-424.

[113] Turney P.D. and Littman M.L., *Measuring Praise and Criticism: Inference of Semantic Orientation from Association.* ACM Trans. Information Systems, 2003. 21(4):315-346.

[114] Valdivia A., Luzón M.V., and Herrera F., *Sentiment Analysis in TripAdvisor.* IEEE Intelligent Systems, 2017. 32(4): 72-77.

[115] Valitutti A., Strapparava C., and Stock O., *Developing Affective Lexical Resources.* PsychNology Journal, 2004. 2(1): 61-83.

[116] Vasilescu F., Langlais P., and Lapalme G., *Evaluating Variants of the Lesk Approach for Disambiguating Words.* Language Resources and Evaluation (LREC'04), 2004. pp. 633-636.

[117] Vilares D., Gómez-Rodríguez C., and Alonso M.A., *Universal, Unsupervised (Rule-based), Uncovered Sentiment Analysis.* Knowl.-Based Syst., 2017. 118: 45-55.

[118] Voorhees E. M., *Using Wordnet to Disambiguate Word Senses for Text Retrieval.* In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1993. pp. 171–180.

[119] Wang G., et al., *POS-RS: A Random Subspace Method for Sentiment Classification based on Part-Of-Speech Analysis.* Inf. Process. Manage. , 2015. 51(4): 458-479.

[120] Wawer A. and Rogozinska D., *How Much Supervision? Corpus-Based Lexeme Sentiment Estimation.* International Conference on Data Mining (ICDM'12) Workshops, 2012. pp. 724-730.

[121] Webb A.R., *Statistical Pattern Recognition.* Second Edition, Academic Press, 2003. p. 592.

[122] Weichselbraun A., Gindl S., Fischer F., Vakulenko S., and Scharl A., *Aspect-Based Extraction and Analysis of Affective Knowledge from Social Media Streams.* IEEE Intelligent Systems, 32(3): 80-88, 2017

[123] Weichselbraun A., Gindl S., and Scharl A., *Enriching Semantic Knowledge Bases for Opinion Mining in Big Data Applications.* Knowl.-Based Syst., 2014. 69: 78-85.

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

[125] Witten I.H. and Frank E., *Data Mining: Practical Machine Learning Tools and Techniques.* Second ed. Morgan Kauffman, 2005. p. 664.

[126] Xia R., Zong C., and Li S., *Ensemble of Feature Sets and Classification Algorithms for Sentiment Classification.* Information Sciences, 2011. 181:1138–1152.

[127] Yeh J. F., Kuang T. Y., Huangv Y. J., Wu M. R., *Dimensional Sentiment Analysis in Valence-Arousal for Chinese Words by Linear Regression.* International Conference on Asian Language Processing (IALP'16) 2016. pp. 328-331.

[128] Zhang S., et al., *Sentiment Analysis of Chinese Micro-blog Text based on Extended Sentiment Dictionary.* Future Generation Comp. Syst., 2018. 81: 395-403.

[129] Zhang Y., *Incorporating Phrase-level Sentiment Analysis on Textual Reviews for Personalized Recommendation* Inter. Conf. on Web Search and Data Mining (WSDM'15), 2015. pp. 435-440.

[130] Zhang Y., Wei Z., Wang Y., Liao T., *Unsupervised Sentiment Analysis of Twitter Posts Using Density Matrix Representation.* European Conference on Information Retrieval (ECIR'18), 2018. pp. 316-329.

[131] Zhao H., Zhang X., and Li K., *A Sentiment Classification Model Using Group Characteristics of Writing Style Features.* International Journal of Pattern Recognition and Artificial Intelligence, 2017. 31(12): 1-19.

[132] Zhao Y., Qin B., Che W., Liu T., *Appraisal Expression Recognition with Syntactic Path for Sentence Sentiment Classification*, Int. J. Comput. Proc. Oriental Lang., 2011. 23(1): 21-37.

Appendices

Appendix I. LISA 1.0 *ID Processing*

LISA 1.0_*IDProcessing* aims to speed-up the process of finding the closest target affect concept w.r.t. a source word concept in the LAG, by making use of the concepts' IDs from WordNet (defined as integer numbers). By analyzing the structure of WordNet, one can realize that the IDs of concepts occurring in the same *hypernymy*/*hyponymy* path in WordNet increase as the corresponding concepts' depths increase in the *hypernymy*/*hyponymy* hierarchy. LISA 1.0_*IDProcessing* makes use of the latter behavior to allow a faster identification of the target affect concept that is closest to the source word concept in the WordNet hierarchy.

```
Algorithm: IDProcessing
Input: LAG graph: G                              // lexical affective graph, connecting in our case: WordNet with WNAH
       Word concepts graph: C                    // corresponding to input word concepts (synsets) in WordNet, $C \subset G$
       Source concept: c
       Set of destination affect nodes A         // corresponding to target affect categories (emotions), $A \subset G$
       Set of affect vectors for: $\nabla_A$          // affect vectors for every target affect node
Output: Source concept affective vector: $V_c$    // affect vectors associated to every affect node in WNAH
Begin
  $V_c = <w(c, a_1)=0, \dots, w(c, a_{|A|})=0>$
  For each $a_i \in A$
  {
      If $ID(c) = ID(a_i)$ where $a \subset A$  Then $Vc = V_a \subset \nabla_A$
      Else
      {
        If $ID(c) \in [minID, maxID]$ Then
        {
            $c_{lb}$ = lowerBound(c, C)          // smallest ID that is higher than the ID of c
            $c_{up}$ = upperBound(c, C)          // largest ID that is lower than the ID of c
        }
        Else
        {
            $c_{lb}$ = highestNbAncestors(c, C)  // following WordNet IsA/HasA hierarchy
            $c_{up}$ = lowestNbAncestors(c, C)   // following WordNet IsA/HasA hierarchy
        }
        Run MaxAffect to compute the weight of $c_{lb}$ and $c_{up}$
        $w(c, a_i)= max(w(c, c_{lb}), w(c, c_{up}))$
      }
  }
  Return $V_c$
End
```

Fig. 26. Summarized pseudo-code of LISA 1.0_*IDProcessing*

The algorithm's pseudo-code is provided in Fig. 26. First, it identifies the ID of the source word concept, and then compares it with the IDs of all affect concepts, which are comprised within a [*minID*, *maxID*] interval, where *minID*=05770995 (representing concept *recognition*) and *maxID*=14565279 (representing concept *insecurity*) in WordNet 3.0. Note that non-affective concepts are also included within the [*minID*, *maxID*] interval in WordNet. Here, we identify three different possibilities:

1. If the word concept maps to an affective concept, then it directly receives the affective vector of the latter.
2. If the ID of the source concept is included within the affective concepts' ID range, the algorithm identifies the two affective concepts which IDs are closest to the source word concept ID: i) the one that has the smallest ID higher than that of the source concept, and ii) the one that has the largest ID lower that the source's. The closest affective concept will be one of the two afore identified concepts, i.e., the one with the highest affective weight w.r.t. the source concept (following Formula 3).
3. If the source concept ID does not fall within the identified interval, then its closest affective concept would be one of the boundary affective concepts. These could be either: i) shallower in the WordNet *hypernymy*/*hyponymy* hierarchy, i.e., having a higher number of ancestors compared with the other affective concepts, or ii) deeper in the WordNet *hypernymy*/*hyponymy* hierarchy, i.e., having a lesser number of ancestors compared with the other affective concepts.

In cases 2 and 3, having identified the closest affective concepts following thier concept IDs, LISA 1.0's *Max_Affect* algorithm is applied between the source concept and each of the boundary concepts in order to identify (the one yielding) the maximum affective intensity score.

Appendix II. LISA 1.0 *Delta Stepping*

LISA 1.0*_DeltaStepping* is a parallelized version of LISA 1.0 following the Delta Stepping (DS) shortest path parallel processing paradigm [71].

```
Algorithm: DeltaStepping
Input: LAG graph: G                               // lexical affective graph, connecting in our case: WordNet with WNAH
       Set of source word concept nodes: C        // corresponding to input word concepts (synsets), $C \subset G$
       Set of destination affect nodes: A         // corresponding to target affect categories (emotions), $A \subset G$
Ouput: Set of affect vectors: $\nabla$            // affect vectors associated to every source concept node in C w.r.t. affect nodes in A
Begin
  Initialize $\nabla = \{V_i\}_{i=1\ldots|C|}$ where $V_i = \langle w(c_i, a_1)=0, \ldots, w(c_i, a_{|A|})=0 \rangle$     1
  Initialize weights of nodes $\in$ G to 0                                                   2

  For each $c_i \in C$                                                                      3
      Create thread for each $a_j \in A$                                                    4
      {                                                                                     5
          If $c_i = a_j$ Then $w(c_i, a_j)=1$                                               6
          Else If $c_i$ isDecendentOf $c_j$ Or $a_j$ isDecendentOf $c_i$ Then               7
          {                                                                                 8
              AllBuckets = fillBuckets($c_i$, $a_j$)                                        9
              Identify $p_j^i = path(c_i, a_j)$                                             10
              Compute $w(c_i, a_j)$ in $p_j^i$ following Formula 3                          11
          } Else                                                                            12
          {                                                                                 13
              s = LCA($c_i$, $a_j$)                  // Lowest common ancestor              14
              AllBuckets = fillBuckets(s, $a_j$)                                            15
              Identify $p_j^s = path(s, a_j)$                                               16
              Compute $w(s, a_j)$ in $p_j^s$ following Formula 3                            17
              $w(w_i, c_j) = w(s, c_j)$                                                     18
          }                                                                                 19

  Return $\nabla$                                                                           20
End
```

Fig. 27. Summarized pseudo-code of algorithm LISA 1.0*_DeltaStepping*

The algorithm's pseudo-code is summarized in Fig. 27 and can be described in four main steps:

1. First, it identifies the nodes that are located on the same path between the source node and the destination node. Here, we considered *hypernymy*/*hyponymy* hierarchical relationships only to simplify processing (maximizing efficiency to the detriment of quality).
2. Second, it sets the weight of every one of these nodes following Formula 3.
3. Third, it divides the nodes into buckets following their weight ranges, where each bucket would contain the nodes having similar weights. Weight ranges are defined as multiples of a user chosen value: Δ (delta).
4. Finally, the buckets are processed in parallel, with every bucket assigned to a dedicated thread in order to identify the nodes with the smallest weights, multiplying the weights in order to get the distance.

Fig. 28 to Fig. 30 describe an example of how LISA 1.0_*DeltaStepping* is processed on a *hypernymy*/*hyponymy* hierarchy. Fig. 28 shows the source and destination nodes where the source is an ancestor of the destination. Fig. 29 shows how nodes are divided into buckets (to be multithreaded), while setting the weight of each node following Formula 3. In Fig. 30, the nodes between the source and the destination are processed for distance computations.

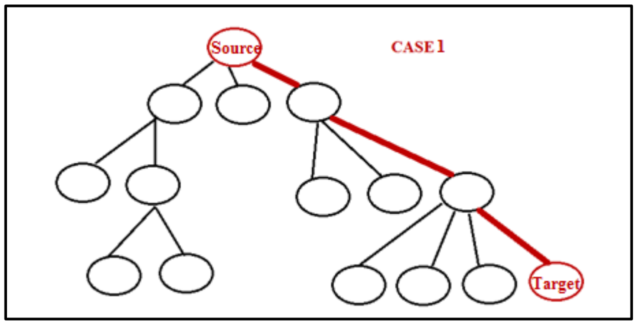


Fig. 28. Nodes on the path between the source and the destination

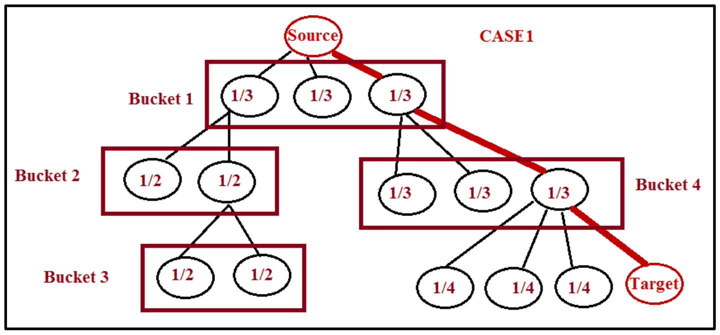


Fig. 29. Nodes are divided into buckets

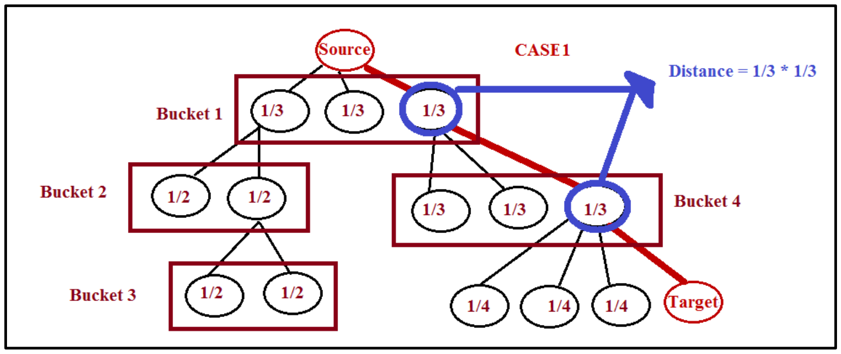


Fig. 30. Distance computation for intermediary nodes