SenticNet 6: Ensemble Application of Symbolic and Subsymbolic AI for Sentiment Analysis

Erik Cambria
Nanyang Technological University
Singapore
cambria@ntu.edu.sg

Yang Li
Nanyang Technological University
Singapore
yang.li@ntu.edu.sg

Frank Z. Xing
Nanyang Technological University
Singapore
zhutian.xing@ntu.edu.sg

Soujanya Poria
Singapore University of Technology and Design
Singapore
sporia@sutd.edu.sg

Kenneth Kwok
Agency for Science, Technology and Research (A*STAR)
Singapore
kenkwok@ihpc.a-star.edu.sg

ABSTRACT

Deep learning has unlocked new paths towards the emulation of the peculiarly-human capability of learning from examples. While this kind of bottom-up learning works well for tasks such as image classification or object detection, it is not as effective when it comes to natural language processing. Communication is much more than learning a sequence of letters and words: it requires a basic understanding of the world and social norms, cultural awareness, commonsense knowledge, etc.; all things that we mostly learn in a top-down manner. In this work, we integrate top-down and bottom-up learning via an ensemble of symbolic and subsymbolic AI tools, which we apply to the interesting problem of polarity detection from text. In particular, we integrate logical reasoning within deep learning architectures to build a new version of SenticNet, a commonsense knowledge base for sentiment analysis.

KEYWORDS

Knowledge representation and reasoning; Sentiment analysis

ACM Reference format:
Erik Cambria, Yang Li, Frank Z. Xing, Soujanya Poria, and Kenneth Kwok. 2020. SenticNet 6: Ensemble Application of Symbolic and Subsymbolic AI for Sentiment Analysis. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19–23, 2020 (CIKM '20),* 10 pages.
https://doi.org/10.1145/3340531.3412003

1 INTRODUCTION

The AI gold rush has become increasingly intense for the huge potential AI offers for human development and growth. Most of what is considered AI today is actually subsymbolic AI, i.e., machine learning: an extremely powerful tool for exploring large amounts of data and, for instance, making predictions, suggestions, and categorizations based on them. All such classifications are made by transforming real items that need to be classified into numbers or features in order to later calculate distances between them. While this is good for making comparison between such items and cluster them accordingly, it does not tell us much about the items themselves. Thanks to machine learning, we may find out that apples are similar to oranges but this information is only useful to cluster oranges and apples together: it does not actually tell us what an apple is, what it is usually used for, where it is usually found, how does it taste, etc. Throughout the span of our lives, we learn a lot of things by example but many others are learnt via our own personal (kinaesthetic) experience of the world and taught to us by our parents, mentors, and friends. If we want to replicate human intelligence into a machine, we cannot avoid implementing this kind of top-down learning.

Integrating logical reasoning within deep learning architectures has been a major goal of modern AI systems [19, 61, 65]. Most of such systems, however, merely transform symbolic logic into a high-dimensional vector space using neural networks. In this work, instead, we do the opposite: we employ subsymbolic AI for recognizing meaningful patterns in natural language text and, hence, represent these in a knowledge base, termed SenticNet 6, using symbolic logic. In particular, we use deep learning to generalize words and multiword expressions into primitives, which are later defined in terms of superprimitives. For example, expressions like `shop_for_iphone11`, `purchase_samsung_galaxy_S20` or `buy_huawei_mate` are all generalized as `BUY(PHONE)` and later reduced to smaller units thanks to definitions such as `BUY(x)=GET(x)` $\wedge$ `GIVE($)`, where `GET(x)` for example is defined in terms of the superprimitive `HAVE` as `!HAVE(x)`$\rightarrow$ `HAVE(x)`.

While this does not solve the symbol grounding problem, it helps reducing it to a great degree and, hence, improves the accuracy of natural language processing (NLP) tasks for which statistical analysis alone is usually not enough, e.g., narrative understanding, dialogue systems and sentiment analysis. In this work, we focus on sentiment analysis where this ensemble application of symbolic and subsymbolic AI is superior to both symbolic representations and subsymbolic approaches, respectively.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM '20, October 19–23, 2020, Virtual Event, Ireland
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412003

Figure 1: An example of sentic algebra.

By deconstructing multiword expressions into primitives and superprimitives, in fact, there is no need to build a lexicon that assigns polarity to thousands of words and multiword expressions: all we need is the polarity of superprimitives. For example, expressions like `grow_profit`, `enhance_reward` or `intensify_benefit` are all generalized as `INCREASE(GAIN)` and, hence, classified as positive (Fig. 1). Likewise, this approach is also superior to most subsymbolic approaches that simply classify text based on word occurrence frequencies. For example, a purely statistical approach would classify expressions like `lessen_agony`, `reduce_affliction` or `diminish_suffering` as negative because of the statistically negative words that compose them. In SenticNet 6, however, such expressions are all generalized as `DECREASE(PAIN)` and thus correctly classified (Fig. 1).

The remainder of the paper is organized as follows: Section 2 briefly discusses related works in the field of sentiment analysis; Section 3 describes in detail how to discover affect-bearing primitives for this task; Section 4 explains how to define such primitives in terms of denotative and connotative information; Section 5 proposes experimental results on 9 different datasets; finally, Section 6 provides concluding remarks.

2 RELATED WORK

Sentiment analysis is an NLP task that has raised growing interest within both the scientific community, for the many exciting open challenges, as well as the business world, due to the remarkable benefits to be had from marketing and financial prediction. While most works approach it as a simple categorization problem, sentiment analysis is actually a complex research problem that requires tackling many NLP tasks, including subjectivity detection, anaphora resolution, word sense disambiguation, sarcasm detection, aspect extraction, and more.

Sentiment analysis research can be broadly categorized into symbolic approaches (i.e., ontologies and lexica) and subsymbolic approaches (i.e., statistical NLP). The former school of thought focuses on the construction of knowledge bases for the identification of polarity in text, e.g., WordNet-Affect [55], SentiWordNet [3], and SenticNet [10]. The latter school of thought leverages statistics-based approaches for the same task, with a special focus on supervised statistical methods. Pang et al. [43] pioneered this trend by comparing the performance of different machine learning algorithms on a movie review dataset and obtained 82% accuracy for polarity detection. Later, Socher et al. [53] obtained 85% accuracy on the same dataset using a recursive neural tensor network (NTN).

With the advent of Web 2.0, researchers started exploiting microblogging text or Twitter-specific features such as emoticons, hashtags, URLs, @symbols, capitalizations, and elongations to enhance the accuracy of social media sentiment analysis. For example, Tang et al. [58] used a convolutional neural network (CNN) to obtain word embeddings for words frequently used in tweets and dos Santos and Gatti [17] employed a deep CNN for sentiment detection in short texts. More recent approaches have been focusing on the development of sentiment-specific word embeddings [44], which are able to encode more affective clues than regular word vectors, and on the use of context-aware subsymbolic approaches such as attention modeling [32, 33] and capsule networks [13, 66].

3 PRIMITIVE DISCOVERY

While the bag-of-words model is good enough for simple NLP tasks such as autocategorization of documents, it does not work well for complex NLP tasks such as sentiment analysis, for which context awareness is often required. Extracting concepts or multiword expressions from text has always been a "pain in the neck for NLP" [49]. Semantic parsing and n-gram models have taken a bottom-up approach to solve this issue by automatically extracting concepts from raw data. The resulting multiword expressions, however, are prone to errors due to both richness and ambiguity of natural language. A more effective way to overcome this hurdle is to take a top-down approach by generalizing semantically-related concepts (e.g., `sell_pizza`, `offer_noodles_for_sale` and `vend_ice_cream` and) via a set of primitives, i.e., a set of ontological parents or more general terms (e.g., `SELL_FOOD`). In this way, most concept inflections can be captured by SenticNet 6: noun concepts like `pasta`, `cheese_cake`, `steak` are replaced with the primitive `FOOD` while verb concepts like `offer_for_sale`, `put_on_sale`, and `vend` are all represented as the primitive `SELL`, which is later deconstructed into simpler primitives, e.g., `SELL(x)= BARTER(x,$)`, where `BARTER(x,y)= GIVE(x)` ∧ `GET(y)`.

The main goal of this generalization is to get away from associating polarity to a static list of affect keywords or multiword expressions by letting SenticNet 6 figure out such polarity on the fly based on the building blocks of meaning. This way, SenticNet 6 reduces the symbol grounding problem and, hence, gets one step closer to natural language understanding. As preached by the field of semiotics, in fact, words are "completely arbitrary signs" [18] that we automatically and almost instinctively connect to semantic representations in our mind. Such process is far from being automatic for an AI, since it never got the chance to learn a language or experience the world the way we did during the first years of our existence. In order to bridge this huge gap between symbols and meaning, we need to ground words (and their associations) into some form of semantic representation, e.g., a structure of semantic features in the Katz-Fodor semantics [28] or in Jackendoff's conceptual structure [26].

While this would be a formidable task for NLP research, it is still manageable in the context of sentiment analysis because, in this domain, the description of such features would be more connotative than denotative. In other words, we do not need define what a concept really is but simply what kind of emotions it generates or evokes.

While the set of mental primitives and the principles of mental combination governing their interaction are potentially infinite for NLP, in the context of sentiment analysis these are bounded by a finite set of emotion categories and much simpler interaction principles that lead to an either positive or negative outcome. Thus, in this work, we leverage subsymbolic AI to automatically discover the primitives that can better generalize SenticNet's commonsense knowledge. This generalization is inspired by different theories on conceptual primitives, including Roger Schank's conceptual dependency theory [51], Ray Jackendoff's work on explanatory semantic representation [25], and Anna Wierzbicka's book on primes and universals [62], but also theoretical studies on knowledge representation [37, 48]. All such theories claim that a decompositional method is necessary to explore conceptualization.

In the same manner as a physical scientist understands matter by breaking it down into progressively smaller parts, a scientific study of conceptualization proceeds by decomposing meaning into smaller parts. Clearly, this decomposition cannot go on forever: at some point we must find semantic atoms that cannot be further decomposed. In SenticNet 6, this 'decomposition' translates into the generalization of words and multiword expressions into primitives and subsequently superprimitives, from which they inherit a specific set of emotions and, hence, a particular polarity.

One of the main reasons why conceptual dependency theory, and many other symbolic methods, were abandoned in favor of subsymbolic techniques was the amount of time and effort required to come up with a comprehensive set of rules. Subsymbolic techniques do not require much time nor effort to perform classification but they are data-dependent and function in a black-box manner (i.e., we do not really know how and why classification labels are produced). In this work, we leverage the representation learning power of long short-term memory (LSTM) networks to automatically discover primitives for sentiment analysis. The deconstruction of primitives into superprimitives is currently a manual process: we leave the automatic (or semi-automatic) discovery of superprimitives to future work.

A sentence S can be represented as a sequence of words, i.e., $S = [w_1, w_2, ...w_n]$ where n is the number of words in the sentence. The sentence can be split into sections such that the prefix: $[w_1, ...w_{i-1}]$ form the left context sentence with l words and the suffix: $[w_{i+1}, ...w_n]$ form the right context sentence with r words. Here, $c = w_i$ is the target word. In the first step, we represent these words in a low-dimensional distributed representation, i.e., word embeddings. Specifically, we use the pre-trained 300-dimensional word2vec embeddings [36] trained on the 3-billion-word Google News corpus. The context sentences and target concept can now be represented as a sequence of word vectors, thus constituting matrices, $L \in \mathcal{R}^{d_w \times l}$, $R \in \mathcal{R}^{d_w \times r}$ and $C \in \mathcal{R}^{d_w \times 1}$ ($d_w = 300$) for left context, right context and target word, respectively.

3.1 biLSTM

To extract the contextual features from these subsentences, we use the biLSTM model on $\mathcal{L}$ and C independently. Given that we represent the word vector for the t^{th} word in a sentence as x_t, the LSTM transformation can be performed as:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \tag{1}$$

$$f_t = \sigma(W_f.X + b_f) \tag{2}$$

$$i_t = \sigma(W_i.X + b_i) \tag{3}$$

$$o_t = \sigma(W_o.X + b_o) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c.X + b_c) \tag{5}$$

$$h_t = o_t \odot tanh(c_t) \tag{6}$$

where d is the dimension of the hidden representations and $W_i, W_f, W_o, W_c \in \mathbb{R}^{d \times (d+d_w)}$, $b_i, b_f, b_o \in \mathbb{R}^d$ are parameters to be learnt during the training (Table 1). σ is the sigmoid function and $\odot$ is element-wise multiplication. The optimal values of the d and k were set to 300 and 100, respectively (based on experiment results on the validation dataset). We used 10 negative samples.

When a biLSTM is employed, these operations are applied in both directions of the sequence and the outputs for each timestep are merged to form the overall representation for that word. Thus, for each sentence matrix, after applying biLSTM, we get the recurrent representation feature matrix as $H_{LC} \in \mathcal{R}^{2d \times l}$, and $H_{RC} \in \mathcal{R}^{2d \times r}$.

3.2 Target Word Representation

The final feature vector $\mathbf{c}$ for target word c is generated by passing C through a multilayer neural network. The equations are as follows:

$$C^* = tanh(W_a.\mathbf{c} + b_a) \tag{7}$$

$$\mathbf{c} = tanh(W_b.C^* + b_b) \tag{8}$$

where $W_a \in \mathcal{R}^{d \times d_w}, W_b \in \mathcal{R}^{k \times d}, b_a \in \mathcal{R}^d$ and $b_b \in \mathcal{R}^k$ are parameters (Table 1) and $\mathbf{c} \in \mathcal{R}^k$ is the final target word vector.

3.3 Sentential Context Representation

For our model to be able to attend to subphrases which are important in providing contexts, we incorporate an attention module on top of our biLSTM for our context sentences. The attention module consists of an augmented neural network having a hidden layer followed by a softmax output (Fig. 2).

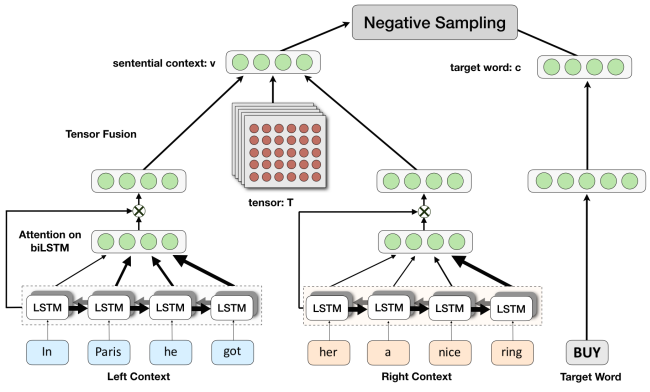

Figure 2: Overall framework for context and word embedding generation.

It generates a vector which provides weights corresponding to the relevance of the underlying context across the sentence. Below, we describe the attention formulation applied on the left context sentence. H_{LC} can be represented as a sequence of $[h_t]$ where $t \in [1, l]$. Let A denote the attention network for this sentence. The attention mechanism of A produces an attention weight vector α and a weighted hidden representation r as follows:

$$P = tanh(W_h.H_{LC}) \tag{9}$$

$$\alpha = softmax(w^T.P) \tag{10}$$

$$r = H_{LC}.\alpha^T \tag{11}$$

where $P \in \mathbb{R}^{d\times l}, \alpha \in \mathbb{R}^l, r \in \mathbb{R}^{2d}$. And, $W_h \in \mathbb{R}^{d\times 2d}, w \in \mathbb{R}^d$ are projection parameters (Table 1). Finally, the sentence representation is generated as:

$$r^* = tanh(W_p.r) \tag{12}$$

Here, $r^* \in \mathbb{R}^{2d}$ and $W_p \in \mathbb{R}^{d\times 2d}$ is the weight to be learnt while training. This generates the overall sentential context representation for the left context sentence: $E_{LC} = r^*$. Similarly, attention is also applied to the right context sentence to get the right context sentence E_{RC}. To get a comprehensive feature representation of the context for a particular concept, we fuse the two sentential context representations, E_{LC} and E_{RC}, using a NTN [52]. It involves a neural tensor $T \in \mathcal{R}^{2d\times 2d\times k}$ which performs a bilinear fusion across k dimensions. Along with a single layer neural model, the overall fusion can be shown as:

$$\mathbf{v} = tanh(E_{LC}^T.T^{[1:k]}.E_{RC} + W.\begin{bmatrix} E_{LC} \\ E_{RC} \end{bmatrix} + b) \tag{13}$$

Here, the tensor product $E_{LC}^T.T^{[1:k]}.E_{RC}$ is calculated to get a vector $\mathbf{v}^* \in \mathcal{R}^k$ such that each entry in the vector $\mathbf{v}^*$ is calculated as $\mathbf{v}_i^* = E_{LC}^T.T^{[i]}.E_{RC}$, where $T^{[i]}$ is the i^{th} slice of the tensor T. $W \in \mathcal{R}^{k\times 4d}$ and $b \in \mathcal{R}^k$ are the parameters (Table 1). The tensor fusion network thus finally provides the sentential context representation $\mathbf{v}$.

3.4 Negative Sampling

To learn the appropriate representation of sentential context and target word, we use word2vec's negative sampling objective function. Here, a positive pair is described as a valid context and word pair and the negative pairs are created by sampling random words from a unigram distribution. Formally, our aim is to maximize the following objective function:

$$Obj = \sum_{\mathbf{c},\mathbf{v}} (log(\sigma(\mathbf{c}.\mathbf{v})) + \sum_{i=1}^{z} log(\sigma(-\mathbf{c}_i.\mathbf{v}))) \tag{14}$$

Here, the overall objective is calculated across all the valid word and context pairs. We choose z invalid word-context pairs where each $-\mathbf{c}_i$ refers to an invalid word with respect to a context.

3.5 Context embedding using BERT

We leverage the BERT architecture [16] to obtain the sentential context embedding of a word. BERT utilizes a transformer network to pre-train a language model for extracting contextual word embeddings. Unlike ELMo and OpenAI-GPT, BERT uses different pre-training tasks for language modeling.

Algorithm 1 Context and target word embedding generation

procedure TrainEmbeddings
  Given sentence $S = [w_1, w_2, ...w_n]$ s.t. w_i is target word.
  $\mathcal{L} \leftarrow E([w_1, w_2, ...w_{i-1}])$ ▷ E() : word2vec embedding
  $\mathcal{R} \leftarrow E([w_{i+1}, w_2, ...w_n])$
  $C \leftarrow E(w_i)$
  $\mathbf{c} \leftarrow$TargetWordEmbedding(C)
  $\mathbf{v} \leftarrow$ContextEmbedding($\mathcal{L}, \mathcal{R}$)
  NegativeSampling($\mathbf{c}$, $\mathbf{v}$)
procedure TargetWordEmbedding(C)
  $C^* = tanh(W_a.\mathbf{c} + b_a)$
  $\mathbf{c} = tanh(W_b.C^* + b_b)$
  return c
procedure ContextEmbedding($\mathcal{L}, \mathcal{R}$)
  $H_{LC} \leftarrow \phi$
  $h_{t-1} \leftarrow 0$
  for t:[1, $i-1$] **do**
    $h_t \leftarrow LSTM(h_{t-1}, L_t)$
    $H_{LC} \leftarrow H_{LC} \cup h_t$
    $h_{t-1} \leftarrow h_t$
  $H_{RC} \leftarrow \phi$
  $h_{t-1} \leftarrow 0$
  for t:[$i+1$, n] **do**
    $h_t \leftarrow LSTM(h_{t-1}, R_t)$
    $H_{RC} \leftarrow H_{RC} \cup h_t$
    $h_{t-1} \leftarrow h_t$
  $E_{LC} \leftarrow$Attention(H_{LC})
  $E_{RC} \leftarrow$Attention(H_{RC})
  $\mathbf{v} \leftarrow$NTN(E_{LC}, E_{RC})
  return v
procedure LSTM(h_{t-1},x_t)
  $X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}$
  $f_t = \sigma(W_f.X + b_f)$
  $i_t = \sigma(W_i.X + b_i)$
  $o_t = \sigma(W_o.X + b_o)$
  $c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c.X + b_c)$
  $h_t = o_t \odot tanh(c_t)$
  return h_t
procedure Attention(H)
  $P = tanh(W_h.H)$
  $\alpha = softmax(w^T.P)$
  $r = H.\alpha^T$
  return r
procedure NTN(E_{LC}, E_{RC})
  $\mathbf{v} = tanh(E_{LC}^T.T^{[1:k]}.E_{RC} + W.\begin{bmatrix} E_{LC} \\ E_{RC} \end{bmatrix} + b)$
  return v

In one of the tasks, BERT randomly masks a percentage of words in the sentences and only predicts those masked words. In the other task, BERT predicts the next sentence given a sentence. This task, in particular, tries to model the relationship among two sentences which is supposedly not captured by traditional bidirectional language models.

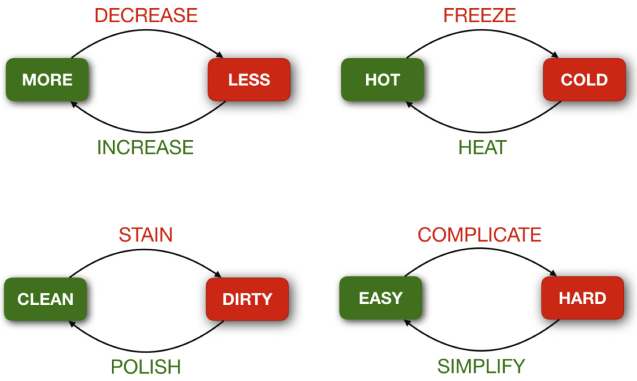

Figure 3: An example of primitive specification.

Consequently, this particular pre-training scheme helps BERT to outperform state-of-the-art techniques by a large margin on key NLP tasks such as question answering and natural language inference where understanding the relation among two sentences is very important. In SenticNet 6, we utilize BERT as follows:

- First, we fine-tune the pre-trained BERT network on the ukWaC corpus [4].
- Next, we calculate the embedding for the context **v**. For this, we first remove the target word **c**, i.e., either the verb or noun from the sentence. The remainder of the sentence is then fed to the BERT architecture which returns the context embedding.
- Finally, we adopt a new similarity measure in order to find the replacement of the word. For this, we need the embedding of the target word which we obtain by simply feeding the word to BERT pre-trained network. Given a target word **c** and its sentential context **v**, we calculate the cosine distance of all the other words in the embedding hyperspace with both **c** and **v**. If **b** is a candidate word, the distance is then calculated as:

$$dist(\mathbf{b},(\mathbf{c},\mathbf{v})) = cos(\mathbf{b},\mathbf{c}) + cos(\mathbf{b},\mathbf{v}) + cos(BERT(\mathbf{v},\mathbf{b}),BERT(\mathbf{v},\mathbf{c})) \quad (15)$$

where $BERT(\mathbf{v},\mathbf{b})$ is the BERT-produced embedding of the sentence formed by replacing word **c** with the candidate word **b** in the sentence. Similarly, $BERT(\mathbf{v},\mathbf{c})$ is the embedding of the original sentence which consists of word **c**.
A stricter rule to ensure high similarity between the target and candidate word is to apply multiplication instead of addition:

$$dist(\mathbf{b},(\mathbf{c},\mathbf{v})) = cos(\mathbf{b},\mathbf{c}) \cdot cos(\mathbf{b},\mathbf{v}) \cdot cos(BERT(\mathbf{v},\mathbf{b}),BERT(\mathbf{v},\mathbf{c})) \quad (16)$$

We rank the candidates as per their cosine distance and generate the list of possible lexical substitutes.

First, we extract all the concepts of the form verb-noun and adjective-noun present in ConceptNet 5 [54]. An example sentence for each of these concepts is also extracted. Then, we take one word from the concept (either a verb/adjective or a noun) to be the target word and the remaining sentence serves as the context.

The goal now is to find a substitute for the target word having the same parts of speech in the given context. To achieve this, we obtain the context and target word embeddings (**v** and **c**) from the joint hyperspace of the network. For all possible substitute words **b**, we then calculate the cosine similarity using equation 16 and rank them using this metric for possible substitutes. This substitution leads to new verb-noun or adjective-noun pairs which bear the same conceptual meaning in the given context. The context2vec code for primitive discovery is available on our github[1].

4 PRIMITIVE SPECIFICATION

The deep learning framework described in the previous section allows for the automatic discovery of concept clusters that are semantically related and share a similar lexical function. The label of each of such cluster is a primitive and it is assigned by selecting the most typical of the terms. In the verb cluster {`increase, enlarge, intensify, grow, expand, strengthen, extend, widen, build_up, accumulate`...}, for example, the term with the highest occurrence frequency in text (the one people most commonly use in conversation) is `increase`.

Hence, the cluster is named after it, i.e., labeled by the primitive `INCREASE` and later defined either via symbolic logic, e.g., `INCREASE(x) = x + a(x)`, where $a(x)$ is an undefined quantity related to x, or in terms of polar transitions, e.g., `INCREASE: LESS` → `MORE` (Fig. 3). Symbolic logic is usually used to define super-primitives or neutral primitives. Polar transitions are used to define polarity-bearing verb primitives in terms of polar state change (from positive to negative and vice versa) via a ying-yang kind of clustering [64].

In both cases, the goal is to define the connotative information associated with primitives and, hence, associate a polarity to them (explained in the next section). Such a polarity is later transferred to words and multiword expressions via a four-layered knowledge representation (Fig. 4).

[1] http://github.com/senticnet/context2vec

Parameters			
Weights			
W_i, W_f, W_o, W_c	$\in \mathbb{R}^{d\times(d+d_w)}$	W_p	$\in \mathbb{R}^{d\times 2d}$
W_b	$\in \mathcal{R}^{k\times d}$	Bias	
W_a	$\in \mathcal{R}^{d\times d_w}$	b_i, b_f, b_o	$\in \mathbb{R}^d$
T	$\in \mathcal{R}^{2d\times 2d\times k}$	b_a	$\in \mathcal{R}^d$
W_h	$\in \mathbb{R}^{d\times 2d}$	b	$\in \mathcal{R}^k$
W	$\in \mathcal{R}^{k\times 4d}$	b_b	$\in \mathcal{R}^k$
w	$\in \mathbb{R}^d$		
Hyperparameters			
d	dimension of LSTM hidden unit		
k	NTN tensor dimension		
z	negative sampling invalid pairs		

Table 1: Summary of notations used in Algorithm 1. Note: d_w is the word embedding size. All the hyperparameters were set using random search [5].

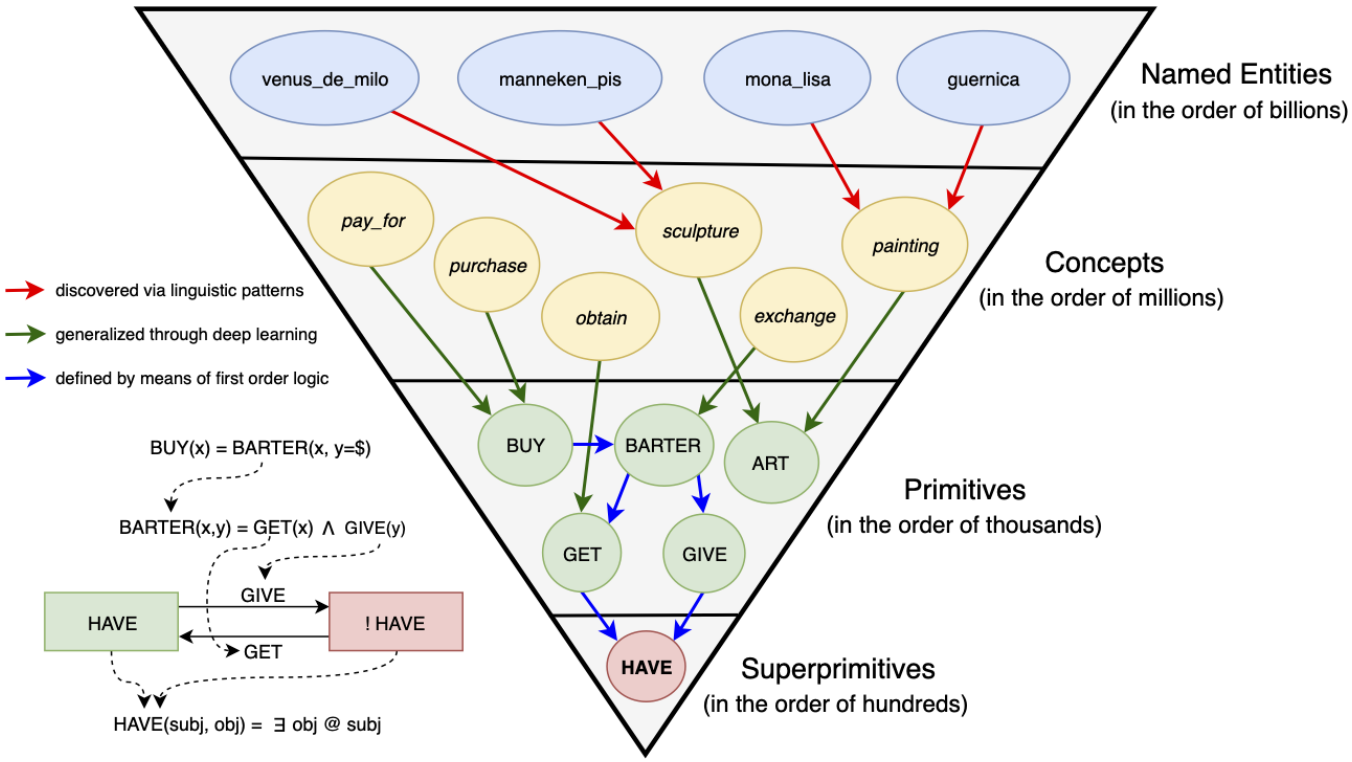

Figure 4: SenticNet 6's dependency graph structure.

In this representation, in particular, named entities are linked to commonsense concepts by IsA relationships from IsaCore [11], a large subsumption knowledge base mined from 1.68 billion webpages. Commonsense concepts are later generalized into primitives by means of deep learning (as explained in the previous section). Primitives are finally deconstructed into superprimitives, basic states and actions that are defined by means of first order logic, e.g., `HAVE(subj,obj)`= ∃ `obj` @ `subj`.

4.1 Key Polar State Specification

In order to automatically discover words and multiword expressions that are both semantically and affectively related to key polar states such as `EASY` versus `HARD` or `STABLE` versus `UNSTABLE`, we use AffectiveSpace [7], a vector space of affective commonsense knowledge built by means of semantic multidimensional scaling.

By exploiting the information sharing property of random projections, AffectiveSpace maps a dataset of high-dimensional semantic and affective features into a much lower-dimensional subspace in which concepts conveying the same polarity and similar meaning fall near each other. In past works, this vector space model has been used to classify concepts as positive or negative by calculating the dot product between new concepts and prototype concepts.

In this case, rather than a distance, we need a discrete path between a key polar state and its opposite (e.g., `CLEAN` and `DIRTY`) throughout the vector space manifolds. While the shortest path (in a k-means sense) between two polar states in AffectiveSpace risks to include many irrelevant concepts, in fact, a path that follows the topological structure of the vector space from one state to its antithetic partner is more likely to contain concepts that are both semantically and affectively relevant. To calculate such a path, we use regularized k-means (RKM) [20], a novel algorithm that finds a morphism between a given point set and two reference points in a vector space $X \in R^d$ where $d \in N^+$ by exploiting the information provided by the available data.

Such morphism is described as a discrete path, composed by a set of prototypes selected based on the data manifolds. Consider a set of points $\boldsymbol{X} = \{\boldsymbol{x}_j \in R^d\}, j = 1, ..., N$ and two points $\boldsymbol{w}_0$ and $\boldsymbol{w}_{N_c} \in R^d$. The path connecting the two points $\boldsymbol{w}_0$ and $\boldsymbol{w}_{N_c+1}$ is described as an ordered set $\boldsymbol{W}$ of N_c prototypes $\boldsymbol{w} \in R^d$. Such path is found by minimizing standard k-means cost function with the addition of a regularization term that considers the distance between ordered centroids.

The cost function can be formalized as:

$$\min_{\boldsymbol{W}} \frac{\gamma}{2} \sum_{i=1}^{N} \sum_{j=1}^{N_c} \|\boldsymbol{x}_i - \boldsymbol{w}_j\|^2 \delta(u_i, j) + \frac{\lambda}{2} \sum_{i=0}^{N_c} \|\boldsymbol{w}_{i+1} - \boldsymbol{w}_i\|^2 \qquad (17)$$

where u_i is the datum cluster.

The novel cost function is composed of two terms weighted by the hyper-parameters γ and λ:

$$\Omega(\boldsymbol{W}, \boldsymbol{u}, \boldsymbol{X}, \gamma, \lambda) = \gamma \Omega_X(\boldsymbol{W}, \boldsymbol{u}, \boldsymbol{X}) + \lambda \Omega_W(\boldsymbol{W}). \qquad (18)$$

The first term coincides with the standard k-means cost function while the second one induces a path topology based on the centroids ordering and controls the level of smoothness of the path.

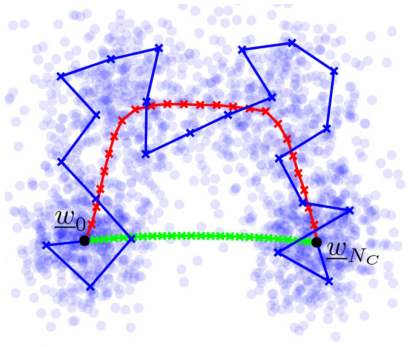

Figure 5: Hyper-parameters influence in the shape of the path.

Fig. 5 proposes a graphical example of the algorithm's behavior for different values of the regularization hyper-parameters: data are represented as blue dots and centroids as crosses; the blue line refers to a configuration in which the first cost function term is prominent; the green one to a configuration where the second term of the cost function is preponderant; finally, the red line refers to a configuration with a good trade-off between the two.

In our case, let C be the set of N concepts belonging to a specific primitive cluster and let $\{\boldsymbol{x}_1, .., \boldsymbol{x}_N\} \in R^d$ their projections induced by embedding F. Additionally, let $p_{start}, p_{end} \in C$ be the two key polar states corresponding to the two extremes of the path under analysis. Accordingly, RKM is used to identify the path that connects p_{start} with p_{end} in AffectiveSpace. Thus, the algorithm's output is the list of intermediate concepts that characterize the transition induced by the data distribution.

Because positive and negative concepts are found in diametrically opposite zones of the space, we expect the paths calculated by means of RKM to traverse AffectiveSpace from one end to the other. This ensures the discovery of enough concepts that are both semantically and affectively related to both polar states. Towards the center of the space, however, there are many low-intensity (almost neutral) concepts. Hence, we only consider the first 20 nearest concepts to each polar state within the discovered morphism. If we set p_{start} = `CLEAN` and p_{end} = `DIRTY`, for example, we only assign the first 20 concepts of the path (e.g., `cleaned`, `spotless`, and `immaculate`) to p_{start} and the last 20 concepts of the path (e.g., `filthy`, `stained`, and `soiled`) to p_{end}.

We also use this morphism to assign emotion labels to key polar states, based on the average distance (dot product) between the concepts of the path (the first 20 and the last 20, respectively) and the key concepts in AffectiveSpace that represent emotion labels (positive and negative, respectively) of the Hourglass of Emotions [56], an emotion categorization model for sentiment analysis consisting of 24 basic emotions organized around four independent but concomitant affective dimensions (Fig. 6).

In the previous example, for instance, `CLEAN` would be assigned the label `pleasantness` because it is the nearest emotion concept to `cleaned`, `spotless`, `immaculate`, etc. on average. Likewise, `DIRTY` would be assigned the label `disgust` because it is the nearest emotion concept to `filthy`, `stained`, `soiled`, etc. on average.

This way, key polar states get mapped to emotion categories of the Hourglass model and, by the transitive property, all the concepts connected to such states inherit the same emotion and polarity classification (Fig. 7).

5 EXPERIMENTS

In this section, we evaluate the performance of both the subsymbolic and symbolic segments of SenticNet 6 (the former being the deep learning framework for primitive discovery, the latter being the logic framework for primitive specification) on 9 different datasets.

5.1 Subsymbolic Evaluation

In order to evaluate the performance of our context2vec framework for primitive discovery, we employed it to solve the problem of lexical substitution. We used ukWaC as the training corpus. We removed sentences with length greater than 80 (which resulted in a 7% reduction of the corpus), lower-cased text, and removed tokens with low occurrence. Finally, we were left with a corpus of 173,000 words. As for lexical substitution evaluation datasets, we used the LST-07 dataset from the lexical substitution task of the 2007 Semantic Evaluation (SemEval) challenge [34] and the 15,000 target word all-words LST-14 dataset from SemEval-2014 [30].

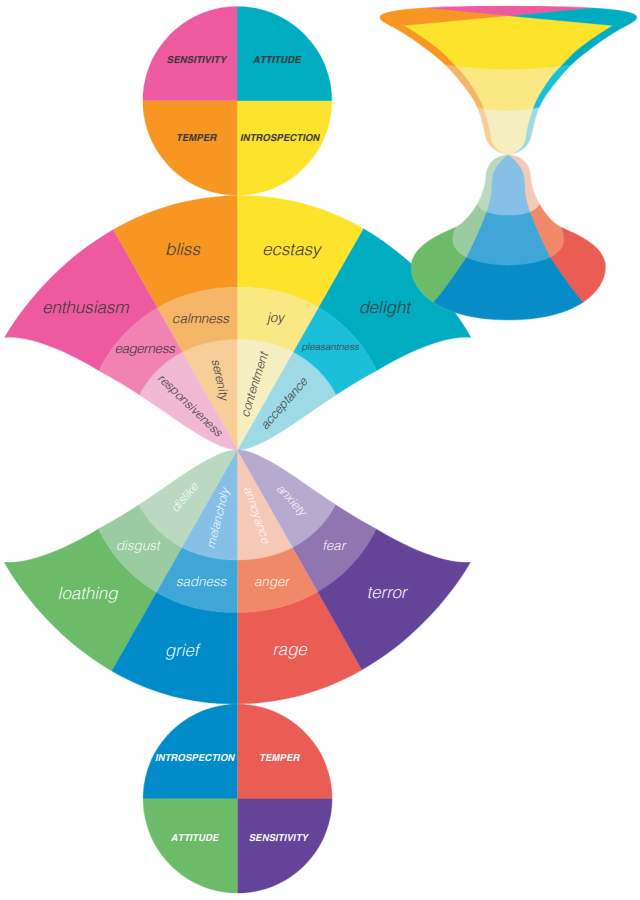

Figure 6: The Hourglass of Emotions.

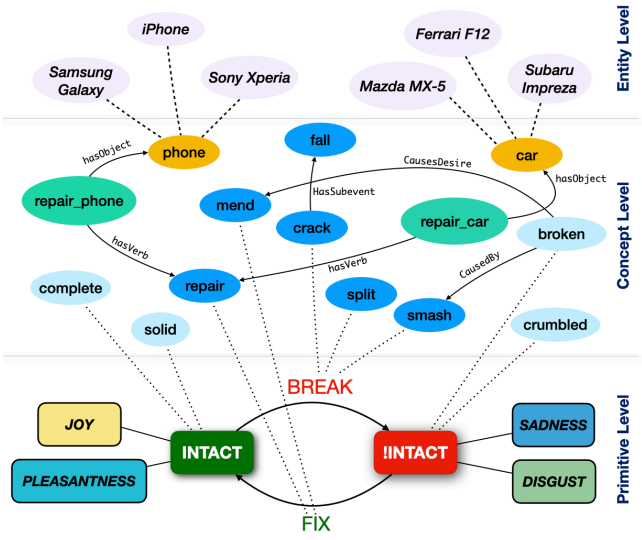

Figure 7: A sketch of SenticNet 6's semantic network.

The first one comes with a 300-sentence dev set and a 1710-sentence test set split; the second one comes with a 35% and 65% split, which we used as the dev set and test set, respectively. The performance is measured using generalized average precision in which we rank the lexical substitutes of a word based on the cosine similarity score calculated among a substitution and the context embedding. This ranking is compared to the gold standard lexical substitution ranking provided in the dataset.

Model	LST-07 [34]	LST-14 [30]
Baseline 1	52.35%	50.05%
Baseline 2	55.10%	53.60%
Context2vec	**59.48%**	**57.32%**

Table 2: Comparison between our approach and two baselines on two datasets for lexical substitution.

The performance of this approach is shown in Table 2, in which we compare it with two baselines. Baseline 1 has been implemented by training the skipgram model on the learning corpus and then simply taking the average of the words present in the context as context representation. The cosine similarity among this context representation and the target word embeddings is calculated to find a match for the lexical substitution. Baseline 2 is a model proposed by [35] to find lexical substitution of a target based on skipgram word embeddings and incorporating syntactic relations in the skipgram model.

5.2 Symbolic Evaluation

As mentioned earlier, the deconstruction of primitives into super-primitives is currently performed manually and, hence, it does not require evaluation. Therefore, we only evaluate the quality of key polar state specification using RKM (as shown in Table 3) in comparison with k-means and sentic medoids [8] on a LiveJournal corpus of 5,000 concepts (LJ-5k).

Model	LJ-5k
K-means	77.91%
Sentic medoids	82.76%
RKM	**91.54%**

Table 3: Comparison between RKM and two baselines on a dataset for concept polarity detection.

5.3 Ensemble Evaluation

We tested SenticNet 6 (available both as a standalone XML repository[2] and as an API[3]) against six commonly used benchmarks for sentence-level sentiment analysis, namely: STS [50], an evaluation dataset for Twitter sentiment analysis developed in 2013 consisting of 1,402 negative tweets and 632 positive ones; SST [53], a dataset built in 2013 consisting of 11,855 movie reviews and containing 4,871 positive sentences and 4,650 negative ones; SemEval-2013 [40], a dataset consisting of 2,186 negative and 5,349 positives tweets constructed for the Twitter sentiment analysis task (Task 2) in the 2013 SemEval challenge; SemEval-2015 [47], a dataset built for Task 10 of SemEval 2015 consisting 15,195 tweets and containing 5,809 positive sentences and 2,407 negative ones; SemEval-2016 [39], a dataset constructed in 2016 for Task 4 of the SemEval challenge consisting of 17,639 tweets about 100 topics and containing 13,942 positive sentences and 3,697 negative ones; finally, Sanders [2], a dataset consisting of 5,512 tweets on four different topics of which 654 are negative and 570 positive.

We used these six datasets to compare SenticNet 6 with 15 popular sentiment lexica, namely: ANEW [6], a list of 1,030 words created in 1999; WordNet-Affect [55], an extension of WordNet made of 4,787 words developed in 2004; Opinion Lexicon [22], a lexicon of 6,789 words built in the same year by means of opinion word extraction from product reviews; Opinion Finder [63], a lexicon of 8,221 words created in 2005 using a polarity classifier; Micro WNOp [12], a lexicon of 5,636 words created in 2007; Sentiment140 [21], a lexicon of 62,466 words developed in 2009; SentiStrength [59] and SentiWordNet [3], two lexica created in 2010 consisting of 2,546 and 23,089 words, respectively; General Inquirer [57], a lexicon of 8,639 words with 1,916 of them containing polarity built in 2011; AFINN [41], a lexicon of 2,477 words constructed in the same year; EmoLex [38], a lexicon of 5,636 words built in 2013; NRC HS Lexicon [67] and VADER [23], two lexica developed in 2014 containing 54,128 and 7,503 words, respectively; MPQA [15], a lexicon of 8,222 words built in 2015; finally, SenticNet 5, the predecessor of SenticNet 6, a knowledge base of 100,000 commonsense concepts.

We set the experiment as a binary classification problem so the labels of both datasets and lexica were reduced to simply positive versus negative. To be fair to all lexica, two basic linguistic patterns [45] were used, namely: negation and adversative patterns. If we do not apply such patterns, in fact, sentences like "The car is very old but rather not expensive" would be wrongly classified by all lexica although most of them correctly list both 'old' and 'expensive' as negative (Fig. 8).

[2]http://sentic.net/downloads
[3]http://sentic.net/api

Model	Year	SST Dataset [53]	STS Dataset [50]	SemEval-2013 [40]	SemEval-2015 [47]	SemEval-2016 [39]	Sanders [2]
ANEW [6]	1999	31.21%	36.77%	42.72%	33.13%	42.20%	27.70%
WordNet-Affect [55]	2004	04.51%	11.98%	03.82%	03.27%	03.53%	05.64%
Opinion Lexicon [22]	2004	**54.21%**	60.72%	41.00%	43.15%	37.83%	54.33%
Opinion Finder [63]	2005	53.60%	55.71%	47.50%	43.97%	46.75%	46.98%
Micro WNOp [12]	2007	15.45%	18.94%	19.13%	16.97%	17.85%	15.36%
Sentiment140 [21]	2009	**55.75%**	**67.69%**	45.67%	**50.92%**	41.70%	**64.95%**
SentiStrength [59]	2010	36.76%	51.53%	37.28%	41.51%	33.97%	44.85%
SentiWordNet [3]	2010	50.19%	48.75%	50.15%	50.31%	**49.62%**	43.55%
General Inquirer [57]	2011	25.91%	11.14%	16.06%	12.47%	16.78%	10.29%
AFINN [41]	2011	44.81%	58.50%	43.82%	44.99%	40.13%	53.19%
EmoLex [38]	2013	46.94%	47.63%	45.12%	42.33%	42.38%	44.12%
NRC HS Lexicon [67]	2014	47.90%	49.86%	28.56%	42.54%	25.28%	54.33%
VADER [23]	2014	50.72%	**64.90%**	**50.36%**	49.08%	45.93%	**57.27%**
MPQA [15]	2015	53.71%	55.43%	46.75%	43.97%	45.42%	46.57%
SenticNet 5 [10]	2018	53.61%	55.71%	**68.17%**	**56.03%**	**70.80%**	48.37%
SenticNet 6	2020	**75.43%**	**83.82%**	**81.79%**	**80.19%**	**82.23%**	**77.62%**

Table 4: Comparison with 15 popular lexica on 6 benchmark datasets for sentiment analysis (top 3 results in bold).

Since most of the datasets we used are for Twitter sentiment analysis, initially we also wanted to apply microtext normalization to all sentences before processing them through the lexica. If we did that, however, we should have also applied many other NLP tasks required for proper polarity detection [9], e.g., anaphora resolution and sarcasm detection, so eventually we refrained from doing so. Classification results are shown in Table 4. SenticNet 6 was the best-performing lexicon mostly because of its bigger size (200,000 words and multiword expressions). Most of the classification errors made by other lexica, in fact, were due to a missing entry in the knowledge base. Most of the sentences misclassified by SenticNet 6, instead, were using sarcasm or contained microtext.

6 CONCLUSION

In the past, SenticNet has been employed for many different tasks other than polarity detection, e.g., recommendation systems [24], stock market prediction [31], political forecasting [46], irony detection [60], drug effectiveness measurement [42], depression detection [14], mental health triage [1], vaccination behavior detection [27], psychological studies [29], and more.

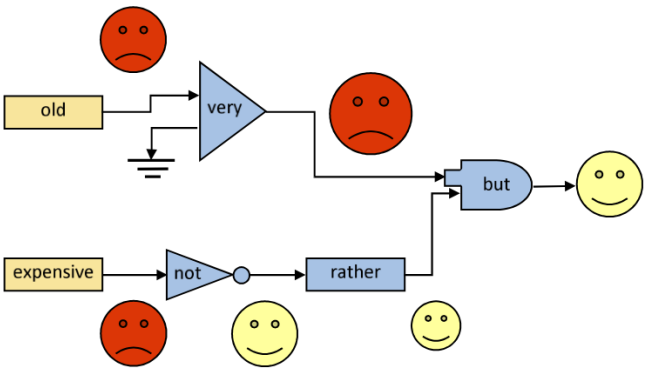

Figure 8: Sentiment data flow for the sentence "The car is very old but rather not expensive" using linguistic patterns.

To enhance the accuracy of all such tasks, we propose a new version of SenticNet built using an approach to knowledge representation that is both top-down and bottom-up: top-down for the fact that it leverages symbolic models (i.e., logic and semantic networks) to encode meaning; bottom-up because it uses subsymbolic methods (i.e., biLSTM and BERT) to implicitly learn syntactic patterns from data. We believe that coupling symbolic and subsymbolic AI is key for stepping forward in the path from NLP to natural language understanding. Machine learning is only useful to make a 'good guess' based on past experience because it simply encodes correlation and its decision-making process is merely probabilistic. As professed by Noam Chomsky, natural language understanding requires much more than that: "you do not get discoveries in the sciences by taking huge amounts of data, throwing them into a computer and doing statistical analysis of them: that's not the way you understand things, you have to have theoretical insights".

ACKNOWLEDGMENTS

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

REFERENCES

[1] Hayda Almeida, Marc Queudot, and Marie-Jean Meurs. 2016. Automatic triage of mental health online forum posts: CLPsych 2016 system description. In *Workshop on Computational Linguistics and Clinical Psychology*. 183–187.
[2] Sanders Analytics. 2015. Sanders Dataset. (2015). http://sananalytics.com/lab
[3] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.. In *LREC*. 2200–2204.
[4] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation* 43, 3 (2009), 209–226.
[5] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research* 13, 1 (2012), 281–305.
[6] Margaret Bradley and Peter Lang. 1999. *Affective Norms for English Words (ANEW): Stimuli, Instruction Manual and Affective Ratings.* Technical Report. The Center for Research in Psychophysiology, University of Florida.
[7] Erik Cambria, Jie Fu, Federica Bisio, and Soujanya Poria. 2015. AffectiveSpace 2: Enabling Affective Intuition for Concept-Level Sentiment Analysis. In *AAAI*. 508–514.

[8] Erik Cambria, Thomas Mazzocco, Amir Hussain, and Chris Eckl. 2011. Sentic Medoids: Organizing Affective Common Sense Knowledge in a Multi-Dimensional Vector Space. In *LNCS 6677*. 601–610.

[9] Erik Cambria, Soujanya Poria, Alexander Gelbukh, and Mike Thelwall. 2017. Sentiment Analysis is a Big Suitcase. *IEEE Intelligent Systems* 32, 6 (2017), 74–80.

[10] Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *AAAI*. 1795–1802.

[11] Erik Cambria, Yangqiu Song, Haixun Wang, and Newton Howard. 2014. Semantic Multi-Dimensional Scaling for Open-Domain Sentiment Analysis. *IEEE Intelligent Systems* 29, 2 (2014), 44–51.

[12] Sabrina Cerini, Valentina Compagnoni, Alice Demontis, Maicol Formentelli, and Caterina Gandini. 2007. Micro-WNOp: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining. *Language resources and linguistic theory: Typology, Second Language Acquisition, English linguistics* (2007), 200–210.

[13] Zhuang Chen and Tieyun Qian. 2019. Transfer Capsule Network for Aspect Level Sentiment Classification. In *ACL*. 547–556.

[14] Ting Dang, Brian Stasak, Zhaocheng Huang, Sadari Jayawardena, Mia Atcheson, Munawar Hayat, Phu Le, Vidhyasaharan Sethu, Roland Goecke, and Julien Epps. 2017. Investigating word affect features and fusion of probabilistic predictions incorporating uncertainty in AVEC 2017. In *Workshop on Audio/Visual Emotion Challenge*. 27–35.

[15] Lingjia Deng and Janyce Wiebe. 2015. MPQA 3.0: An entity/event-level sentiment corpus. In *NAACL*. 1323–1328.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[17] Cıcero Nogueira dos Santos and Maıra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*. 69–78.

[18] Umberto Eco. 1984. *Semiotics and Philosophy of Language*. Indiana University Press.

[19] Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61 (2018), 1–64.

[20] Marco Ferrarotti, Sergio Decherchi, and Walter Rocchia. 2019. Finding Principal Paths in Data Space. *IEEE Transactions on Neural Networks and Learning Systems* 30, 8 (2019), 2449–2462.

[21] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford* 1, 12 (2009).

[22] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*. 168–177.

[23] Clayton J Hutto and Eric GIlbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*. 216–225.

[24] Muhammad Ibrahim, Imran Sarwar Bajwa, Riaz Ul-Amin, and Bakhtiar Kasi. 2019. A neural network-inspired approach for improved and true movie recommendations. *Computational intelligence and neuroscience* (2019), 4589060.

[25] Ray Jackendoff. 1976. Toward an explanatory semantic representation. *Linguistic Inquiry* 7, 1 (1976), 89–150.

[26] Ray Jackendoff. 1983. *Semantics and cognition*. MIT Press.

[27] Aditya Joshi, Xiang Dai, Sarvnaz Karimi, Ross Sparks, Cecile Paris, and C Raina MacIntyre. 2018. Shot or not: Comparison of NLP approaches for vaccination behaviour detection. In *SMM4H@EMNLP*. 43–47.

[28] Jerrold Katz and Jerry Fodor. 1963. The structure of a Semantic Theory. *Language* 39 (1963), 170–210.

[29] Megan O Kelly and Evan F Risko. 2019. The Isolation Effect When Offloading Memory. *Journal of Applied Research in Memory and Cognition* 8, 4 (2019), 471–480.

[30] Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What Substitutes Tell Us - Analysis of an "All-Words" Lexical Substitution Corpus. In *EACL*. 540–549.

[31] Xiaodong Li, Haoran Xie, Raymond YK Lau, Tak-Lam Wong, and Fu-Lee Wang. 2018. Stock prediction via sentimental transfer learning. *IEEE Access* 6 (2018), 73110–73118.

[32] Qiao Liu, Haibin Zhang, Yifu Zeng, Ziqi Huang, and Zufeng Wu. 2018. Content Attention Model for Aspect Based Sentiment Analysis. In *WWW*. 1023–1032.

[33] Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In *AAAI*. 5876–5883.

[34] Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 task 10: English lexical substitution task. In *SemEval*. 48–53.

[35] Oren Melamud, Omer Levy, Ido Dagan, and Israel Ramat-Gan. 2015. A Simple Word Embedding Model for Lexical Substitution. In *VS@HLT-NAACL*. 1–7.

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[37] Marvin Minsky. 1975. A framework for representing knowledge. In *The psychology of computer vision*, Patrick Winston (Ed.). McGraw-Hill, New York.

[38] Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence* 29, 3 (2013), 436–465.

[39] Preslav Nakov, Alan Ritter, Sara Rosentha, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *SemEval*.

[40] Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. In *SemEval*. 312–320.

[41] Finn Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *CoRR* abs/1103.2903 (2011).

[42] Samira Noferesti and Mehrnoush Shamsfard. 2015. Using Linked Data for polarity classification of patients' experiences. *Journal of biomedical informatics* 57 (2015), 6–19.

[43] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *EMNLP*. 79–86.

[44] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect Extraction for Opinion Mining with a Deep Convolutional Neural Network. *Knowledge-Based Systems* 108 (2016), 42–49.

[45] Soujanya Poria, Erik Cambria, Alexander Gelbukh, Federica Bisio, and Amir Hussain. 2015. Sentiment Data Flow Analysis by Means of Dynamic Linguistic Patterns. *IEEE Computational Intelligence Magazine* 10, 4 (2015), 26–36.

[46] Lei Qi, Chuanhai Zhang, Adisak Sukul, Wallapak Tavanapong, and David Peterson. 2016. Automated coding of political video ads for political science research. In *IEEE International Symposium on Multimedia*. 7–13.

[47] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *SemEval*. 451–463.

[48] David Rumelhart and Andrew Ortony. 1977. The representation of knowledge in memory. In *Schooling and the acquisition of knowledge*. Erlbaum, Hillsdale, NJ.

[49] Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword Expressions: A Pain in the Neck for NLP. In *CICLing*. 1–15.

[50] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2013. Evaluation datasets for Twitter sentiment analysis: a survey and a new dataset, the STS-Gold. In *AI*IA*.

[51] Roger Schank. 1972. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology* 3 (1972), 552–631.

[52] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.

[53] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*. 1631–1642.

[54] Robert Speer and Catherine Havasi. 2012. ConceptNet 5: A Large Semantic Network for Relational Knowledge. In *Theory and Applications of Natural Language Processing*. Chapter 6.

[55] Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: An Affective Extension of WordNet. In *LREC*. 1083–1086.

[56] Yosephine Susanto, Andrew Livingstone, Bee Chin Ng, and Erik Cambria. 2020. The Hourglass Model Revisited. *IEEE Intelligent Systems* 35, 5 (2020).

[57] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics* 37, 2 (2011), 267–307.

[58] Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for Twitter sentiment classification. In *SemEval*. 208–212.

[59] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American society for information science and technology* 61, 12 (2010), 2544–2558.

[60] Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. We usually don't like going to the dentist: Using common sense to detect irony on Twitter. *Computational Linguistics* 44, 4 (2018), 793–832.

[61] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. 2019. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *ICML*. 6545–6554.

[62] Anna Wierzbicka. 1996. *Semantics: Primes and Universals*. Oxford University Press.

[63] Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. OpinionFinder: A system for subjectivity analysis. In *HLT/EMNLP*. 34–35.

[64] Lei Xu. 1997. Bayesian Ying–Yang machine, clustering and number of clusters. *Pattern Recognition Letters* 18, 11 (1997), 1167–1178.

[65] Fan Yang, Zhilin Yang, and William Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*. 2319–2328.

[66] Wei Zhao, Haiyun Peng, Steffen Eger, Erik Cambria, and Min Yang. 2019. Towards scalable and reliable capsule networks for challenging NLP applications. In *ACL*. 1549–1559.

[67] Xiaodan Zhu, Svetlana Kiritchenko, and Saif Mohammad. 2014. NRC-canada-2014: Recent improvements in the sentiment analysis of tweets. In *SemEval*. 443–447.