Experiments in SenticNet

NTU-India Connect Research Internship Programme

Nigel Fernandez*

April 7, 2017

Contents

*nigelsteven.fernandez@iiitb.org

List of Tables

1 Introduction

`SenticNet` is a concept level sentiment analyser. I performed two major types of experiments in `SenticNet` (1) knowledge expansion and (2) consistency checking. In knowledge expansion I created new Sentic vectors to augment the existing data base of $50,000$ vectors. A total of 6175 new Sentic vectors were created with varying (subjective) accuracy. My expansion strategies are based on two related algorithms to find semantically similar words to a given word. In my consistency checking experiments, I performed both checks within `SenticNet` and checks between `SenticNet` and other models. The results of these checks were tabulated as confusion matrices.

2 Knowledge Expansion

The goal of this experiment was to augment the Sentic vector database of `SenticNet` by creating new Sentic vectors or concepts using `SentiWordNet` as a resource for new concepts. A concept is an 8 dimension vector of the form [concept name, mood tag 1, mood tag 2, semantic 1, semantic 2, semantic 3, semantic 4, semantic 5]. An example of a Sentic vector is ['hard-working', 'joy', 'surprise', 'assiduous', 'sedulous', 'endeavor', 'diligent', 'regular']. Mood tags are chosen based on polarity of the concept. For a positive polarity concept two relevant tags from [joyful, interesting, surprising, admirable] are chosen while for a negative polarity concept two relevant tags from [sad, scared, angry, disgusting] are chosen. The constraint for mood tags is that interesting and surprising or scared and angry should not be used together.

The 5 semantics in the Sentic vector are the top 5 semantically similar words to the concept from a corpus. The constraints to be followed for concept and semantic names are:

- Should be lemmatized rather that stemmed
- Should be in American English
- Should not contain negation
- Should not be prepositions or conjunctions
- Semantics should have same polarity as the input concept
- City or country names should not be used

2.1 Algorithm 1: Using Precomputed Lemmas from `WordNet` to Find Semantics

A synset in `WordNet` is a set of synonymous words. Each synset contains one or more lemmas, which represent a specific sense of a specific word. We can use these inbuilt relationships to form our Sentic vectors of semantically related semantics to a concept. This algorithm iterates over all synsets in `WordNet` . The polarity of a concept can be obtained using the `SentiWordNet` corpus. The NLTK library provides an interface to this corpus through which we can find the polarity of our concept. It discards synset words which are purely objective, i.e., have a zero positive and negative score or have an equal positive and negative score. It further discards proper nouns. After the filtering, it forms a set of lemmas for the current concept (synset defining word) which are representative of semantically similar words to the synset word and therefore can be used as possible semantics.

Concepts and semantics are lemmatized using the built in Morphy function of the NLTK library in Python. If we successfully remain with at least 5 semantic choices (lemmas) for a concept after the filtering process, we proceed to finding the appropriate mood tags for the concept. We use `SentiWordNet` to calculate the polarity of concepts. Since `SentiWordNet` does not define the polarity all lemmas in `WordNet` , we make the assumption of the set of lemmas of a concept having the same polarity as the concept itself. A concept is defined as positive polar if its positive score is greater that its negative score and negative polar if its negative score is greater than its positive score. If its positive score is equal to its negative score, we define it as purely objective and discard such concepts as stated before.

After obtaining the polarity of the concept, we find its appropriate mood tags from the relevant tag set (positive tags for positive polar words and vice versa). To do so we compute the Lin's similarity measure between the concept and each tag in the tag set. We choose the top two similar tags as appropriate mood tags for the concept. If the tag constraint is violated (interesting and surprising or scared and angry are top two in any order) then we choose the top and third best similar tag. Note that Lin's similarity measure was randomly chosen. Other similarity and relatedness measures can be experimented with in future. Lin's similarity measure returns

$$\frac{2 * \mathrm{IC}(x)}{\mathrm{IC}(word1) + \mathrm{IC}(word2)}$$

where IC(x) is the information content of x in the reference corpus. The Genesis corpus was used as reference.

This completes the formation of our Sentic vector. If the concept is not present in the `SenticNet` database, we add the Sentic vector to our file of new Sentic vectors. This algorithm's pseudo code is present in algorithm 1. An observation to be made here is that we use `SentiWordNet` 's defined lemmas as a proxy for computing semantically similar words / semantics for an input concept. In my next algorithm, I manually compute a semantically similar set of words for an input concept using a reference similarity measure.

2.2 Algorithm 2: Using Similarity Measures to Find Semantics

This algorithm is different from algorithm 1 because it computes the similarity scores between word pairs to answer the fundamental question 'which are the top 5 semantically similar words to a concept'. This algorithm does not rely on lemmas as an implicit provider of semantically similar words but instead computes similarity scores directly.

We begin by iterating over all nouns (can also experiment with adjectives, verbs and adverbs in future) in `WordNet` to form two corpora, one in which all words are of positive polarity and another in which all words are of negative polarity. In the positive corpus, a dictionary of dictionaries in initialized with keys being word pairs mapping to Lin's similarity measures. Each word in an n word positive corpus will form the outer key mapping to a dictionary with $n-1$ inner keys, one key for each remaining word in the same corpus, thus forming all possible keys from word pair combinations. The same process (dictionary of dictionaries creation) is done for the negative corpus.

We iterate over all possible word pairs, compute their Lin's similarity measure and store it in the appropriate place in the dictionary. After both dictionaries (both positive and negative corpus) have been populated, we iterate over all words in a corpus and sort the remaining words based on the Lin's similarity measure present in the word's dictionary entries. This gives us a sorted list of semantically similar words to the input word / concept in decreasing order of similarity. Intuitively, this algorithm (in a brute force manner) answers the question, given a word find X similar words to it.

To compute the appropriate mood tags of the concept, the same procedure *find_best_mood_tags* in algorithm 1 is used. During the computation of the

Algorithm 1 Knowledge Expansion using WordNet Lemmas

```
procedure FIND_BEST_MOOD_TAGS(concept, polarity)
    if polarity == 'positive' then
        tags = [joyful, interesting, surprising, admirable]
    else
        tags = [sad, scared, angry, disgusting]
    tags = sort tags on decreasing Lin's similarity measure to concept
    if no tag conflict then
        return tag[0], tag[1]
    else
        return tag[0], tag[2]
procedure FIND_NEW_SENTIC_VECTOR_1(word_net, senti_word_net, sentic_net)
    for synset in WordNet.all_synsets do
        concept = synset.name
        if SentiWordNet(concept).pos_score == SentiWordNet(concept).neg_score then
            continue
        polarity = SentiWordNet(concept).pos_score > SentiWordNet(concept).neg_score
        if concept is a proper noun then
            continue
        for lemma in concept.lemmas() do
            if lemma is a proper noun then
                continue
                add lemma to semantic set
        lemmatize words in semantic set and remove duplicates
        if concept not in sentic_net_database then
            find_best_mood_tags(concept, polarity)
            add sentic vector to sentic_net
```

Sentic vector, all rules were followed (lemmatized forms, no proper nouns, same polarity, etc). Lin's similarity measure was chosen randomly. We can experiment with other similarity and relatedness measures. This algorithm's pseudo code is present in algorithm 2.

2.3 Results: New Sentic Vectors

The results of my two algorithms, 'Knowledge Expansion using WordNet Lemmas' (1) and 'Knowledge Expansion using Similarity Measure Computations' (2) were (subjectively) accurate. Since algorithm 1 relied on inbuilt similarity computations by `WordNet` in the form of inbuilt lemma realtions, the results were accurate. 37 positive and 36 negative for a total of 73 new Sentic vectors were created and stored in the 'Outputs/1' folder in both csv and txt file formats. There are files for only positive polarity Sentic vectors, negative polarity Sentic vectors and all Sentic vectors combined. Sample Sentic vectors from algorithm 1 are shown in table 1.

Algorithm 2 also performed well when the corpus size was increased as there was a higher probability of finding similar words to a concept in the larger corpus. A corpus size of 5500 positive and 5500 negative noun words were used. The corpus size was limited by the amount of memory in my computer $2*5500*5499$ keys. Future optimizations are possible. 2991 positive and 3111 negative for a total of 6102 new Sentic vectors were created and stored in the 'Outputs/2' folder in the same output format as 'Outputs/1'. Algorithm 2 resulted in a greater number of Sentic vectors as we formed our own lemma clusters (which could potentially lead to decrease in accuracy) and did not use inbuilt lemma clusters from `WordNet` which are very small (less than 5) for some words and discarded in algorithm 1. Sample Sentic vectors from algorithm 2 are shown in table 2.

Algorithm 2 Knowledge Expansion using Similarity Measure Computations

```
procedure FIND_BEST_MOOD_TAGS(concept, polarity)
    same as in algorithm 1
procedure FIND_NEW_SENTIC_VECTOR_2(word_net, senti_word_net, sentic_net)
    for synset in WordNet.all_synsets do
        concept = synset.name
        if SentiWordNet(concept).pos_score == SentiWordNet(concept).neg_score then
            continue
        polarity = SentiWordNet(concept).pos_score > SentiWordNet(concept).neg_score
        if concept is a proper noun then
            continue
        if concept.polarity == 'positive' then
            add lemmatized concept to positive corpus
        else
            add lemmatized concept to negative corpus
    initialize dictionary of dictionaries scores with keys as word pairs
    for words in positive corpus and negative corpus do
        compute Lin's similarity measure between every other word and store in scores
    for words in positive corpus and negative corpus do
        if concept not in sentic_net_database then
            sort all other words based on similarity values in dictionary scores to this current word and store in list semantics
            for word in semantics do
                if word has same polarity as concept then
                    store word in list final_semantics
            find_best_mood_tags(concept, polarity)
            add sentic vector to sentic_net
```

Concept	MoodTag1	MoodTag2	Semantic1	Semantic2	Semantic3	Semantic4	Semantic5
Positive Polarity							
craftily	admirable	surprising	cunningly	foxily	knavishly	slyly	trickily
spruce_up	admirable	surprising	spruce	titivate	tittivate	smarten_up	slick_up
forte	interesting	joyful	strong_suit	long_suit	metier	specialty	speciality
flightiness	joyful	interesting	arbitrariness	whimsicality	whimsy	whimsey	capriciousness
self-control	interesting	joyful	self-possession	possession	willpower	will_power	self-command
Negative Polarity							
drop_out	sad	angry	give_up	fall_by_the_wayside	drop_by_the_wayside	throw_in	throw_in_the_towel
barricade	angry	sad	block	blockade	stop	block_off	block_up
bedlam	sad	scared	booby_hatch	crazy_house	cuckoo's_nest	funny_farm	funny_house
neutralize	sad	disgusting	neutralise	liquidate	waste	knock_off	do_in
brawn	angry	sad	brawniness	muscle	muscularity	sinew	heftiness

Table 1: Sample Sentic Vectors from Algorithm 1

Concept	MoodTag1	MoodTag2	Semantic1	Semantic2	Semantic3	Semantic4	Semantic5
Positive Polarity							
world_record	joyful	interesting	record	success	hit	passing	advent
bear_hug	joyful	interesting	assay	best	embrace	reordering	embodiment
pride_of_place	surprising	admirable	lie	ararat	hiding_place	lookout	close_quarters
family_man	surprising	admirable	newlywed	lover	ma	methuselah	child
grandmaster	surprising	admirable	comer	chess_master	connors	most_valuable_player	football_hero
Negative Polarity							
forced_landing	sad	scared	ground-controlled_approach	crash_landing	intrusion	break	forgiveness
self-destruction	sad	scared	slaughter	murder	annihilation	vandalism	suicide_bombing
thuggery	sad	scared	robbery	wrath	envy	fall	crime
lynching	sad	scared	slaughter	murder	break	wilt	wound
health_care	sad	scared	medicaid	fine	accumulation	premium	largess

Table 2: Sample Sentic Vectors from Algorithm 2

3 Consistency Checks

The knowledge expansion experiments proved to be fun which led me to explore other word models and perform comparisons within and between `SenticNet` . The intra consistency check experiments checked for rule abidance by `SenticNet` (duplicate tags, all semantics should have same polarity, etc). The inter consistency checks aimed to compare `SenticNet` with other popular word models (ex: `SentiWordNet`) and obtain a confusion matrix to give an indication of agreement between the models.

3.1 Intra: Within `SenticNet`

Within `SenticNet` I performed the following checks:

- Check if the set of 5 semantics of a concept are defined concepts themselves in `SenticNet`
- Check if concepts have duplicate tags
- Check for negative intensity for positive polarity concepts and positive intensity for negative polarity concepts
- Check if 5 semantics of a concept have the same polarity as the concept
- Check for interest-surprise or fear-anger used together in tags of a concept
- Check if concepts and their related semantics are lemmatized according to the Morphy function of `WordNet` called by NLTK

The results of these intra consistency checks are presented in table 3.

3.2 Inter: Between `SenticNet` and other Models

After completing simple intra consistency checks the next step was to compare the agreement / correlation levels of `SenticNet` with other popular word models (on overlapping words). These comparisons were done on the polarity labels. The notation followed in the confusion matrix is that the external model's (ex: `SentiWordNet`) polarity labels were taken as the true labels and the labels of `SenticNet` were termed as predicted labels. I imported the following models (either their databases or APIs):

Check	Output
No of undefined semantics	199
No of concepts with duplicate tags	3013
No of concepts with mismatched polarity - intensity values	0
No of concept - semantic pairs with different polarity	38093
No of interest-surprise or fear-anger instances	0
No of non lemmatized concepts or semantics	1201

Table 3: Consistency Checks within `SenticNet`

- Bing Liu's Opinion Lexicon [1]
- MPQA Subjectivity Lexicon [2]
- SentiWordNet [3]
- Harvard General Inquirer [4]

The results of these comparison checks can be visualized in the confusion matrices 4, 5, 6 and 7.

	Predicted Positive	Predicted Negative
Positive	1169	96
Negative	313	2681

Table 4: Confusion Matrix Taking Bing Liu's Opinion Lexicon as True Labels

	Predicted Positive	Predicted Negative
Positive	1869	176
Negative	675	3229

Table 5: Confusion Matrix Taking MPQA Subjectivity Lexicon as True Labels

A summary comparison between `SenticNet` and other models is given in table 8.

	Predicted Positive	Predicted Negative
Positive	4671	1774
Negative	1973	5313

Table 6: Confusion Matrix Taking SentiWordNet as True Labels

	Predicted Positive	Predicted Negative
Positive	1044	105
Negative	167	1328

Table 7: Confusion Matrix Taking Harvard General Inquirer as True Labels

Compared To Model	Accuracy	Precision	Recall	F1 Score
Bing Liu's Opinion Lexicon	0.904	0.789	0.924	0.851
MPQA Subjectivity Lexicon	0.857	0. 735	0.914	0.815
SentiWordNet	0.727	0.703	0.725	0.714
Harvard General Inquirer	0.897	0.862	0.909	0.885

Table 8: Comparison between `SenticNet` and other Models

4 Future Steps

Knowledge expansion algorithms from the learning domain can be explored (ex: using a Naive Bayes classifier after training on a labeled corpus). Other similarity and relatedness measures can be experimented with other than the current Lin's similarity measure.

Bibliography

[1] Hu and Liu. *Bing Liu's Opinion Lexicon*. URL: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html.

[2] Janyce Wiebe Theresa Wilson and Paul Hoffmann. "Polarity in Phrase-Level Sentiment Analysis". In: *Proceedings of HLT/EMNLP* (2005).

[3] Andrea Esuli Stefano Baccianella and Fabrizio Sebastiani. "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining". In: *Proceedings of the International Conference on Language Resources and Evaluation* (2010).

[4] *Harvard General Inquirer*. URL: http://www.wjh.harvard.edu/~inquirer/.