Aspect Based Sentiment Analysis on Laptop Reviews

Carolina Caprile
Brandeis University
carocaprile@brandeis.edu

Abstract

The goal of this project is to identify opinions expressed within laptop reviews in terms of its attributes using a granular technique known as Aspect Based Sentiment Analysis. In this paper we describe the efforts in completing three information extraction subtasks: i) detecting all attributes alluded in each review, ii) discerning the aspect term that speaks to each attribute, iii) identifying the opinion related to that attribute and determine its polarity.

1 Introduction

The aim of Aspect Based Sentiment Analysis (ABSA) is to mine opinions from texts related to specific entities and attributes. Hence, this method is able to analyze large amounts of unstructured text and obtain granular information about the users' experiences and feedback.

This is very valuable information because it cannot be extracted from the users' ratings nor by applying regular Sentiment Analysis, given that both of these only provide a partial and generic perspective of the users' comment: its sentiment. To compensate this deficiency, ABSA aims to identify which aspect in particular is this sentiment referring to. Consider the following example:

"Love the graphics, awesome programs and really cool default background."

The positive sentiment of this review can be clearly identified by the vocabulary used: "love", "awesome" and "cool". Even though these assessments give us information about the sentiment of the review as a whole, the important question to answer is: to what aspect of the product does this sentiment relate to exactly? Does it regard its design, its operation system, its performance…? This is the question that ABSA intends to answer.

Another advantage of ABSA is that it can identify multiple polarities in one same comment. Take into account that in one same review a customer might express satisfaction regarding some aspects of the service but discontent concerning others. For instance:

"It absolutely is more expensive than most PC laptops, but the ease of use, security, and minimal problems that have arisen make it well worth the price tag."

When analyzing this review, ABSA is able to distinguish two different sentiments -positive and negative- related to two different aspects of the product: its usability and price, respectively. This information is far more precise than that provided by regular Sentiment Analysis, that would yield only one and misleading sentiment for the whole review.

In this paper, we aim to apply an Aspect Based Sentiment Analysis on laptop reviews. Toward this end, three information extraction subtasks will be completed: i) detecting all attributes alluded in each review, ii) discerning the aspect term that speaks to each attribute, and

iii) identifying the opinion related to that attribute and determine its polarity.

The rest of the paper is organized as follows. In Section 2, we make references to other works that have already used Aspect Based Sentiment Analysis. In Section 3 we define the problem at hand and explain our annotation schema with some examples. In sections 4 and 5 we describe our corpus and its preprocessing. In Sections 6 and 7 we discuss the efforts done in completing each subtask, methods used and their results. We conclude our paper and describe future work in Section 8.

2 Related Work

Traditional sentiment analysis is focused on sentiment polarity detection. Gradually, a fine-grained sentiment mining approach was needed and became popular. Researchers extracted the topic from given text (Gamon, 2005) and extracted the features and opinion phrases from product reviews (Hu & Liu, 2004; Popescu & Etzioni, 2005). This fine-grained approach, namely aspect-based sentiment analysis (ABSA) became an important level in sentiment analysis (Liu, 2012).

Most of the ABSA projects are based on reviews. The early ABSA studies tended to use the corpus of product reviews (Hu & Liu, 2004; Popescu & Etzioni, 2005; Titov & McDonald, 2008). In recent research, product reviews are still the mainstream datasets, however, movie reviews, hotel reviews etc. are also used (Thet, Na & Khoo, 2010).

One of the founding papers in the ABSA field was presented by Hu and Liu (2004), where they describe in detail the workflow for mining and summarizing customer reviews in order to help both potential customers and manufacturers to have access to relevant feedback in a structured object. In order to mine the product features from reviews, they apply a finding frequent noun and noun phrases method, based on the fact that, when people comment on different aspects of one entity, usually the vocabulary used converges.

The precision of this algorithm was improved in Popescu and Etzioni (2005), that applied pointwise mutual information (PMI) scores in order to remove those noun phrases that may not refer to aspects of a given entity.

With regards to opinion mining and polarity detection, Ding, X., Liu, B., and Yu, P. (2008) propose a holistic lexicon-based approach. In other words, they present model techniques to deal with opinion phrases and determining their semantic orientation by making use of the review and sentence context and turning to general natural language rules to identify and interpret their lexical features. The main aspects they deal with are i) the combination of multiple opinion words (which sometimes may be conflicting); ii) context or domain dependent opinion words; and iii) language constructs which can change the semantic orientations of opinion words.

The largest ABSA annotation task is the SemEval (Semantic Evaluation) 2016 project, task5 (Ponitiki et al., 2016). SemEval is an ongoing project of focuses on evaluations of computational semantic analysis systems. From 2014, SemEval started extending their sentiment analysis research to ABSA, and since then it has been using corpora containing reviews from diverse domains (e.g. restaurants, hotels, laptops, cameras, etc.) in multiple languages.

3 Problem Definition

Given that we are aiming for Aspect Based Sentiment Analysis, there are two components that we are interested in identifying:

1. **Aspect**: composed of the pair entity and attribute, that correspond to a topic and its specific feature.

2. **Sentiment**: each opinion is charged with a sentiment that can be either positive or negative.

Even though in the literature “aspect” is defined as a pair of entity and attribute, for our purposes “aspect” will be interpreted as one component, which refers to either a subcomponent or an attribute of the object. A review will always make reference to a laptop (object) and some of its aspects, which can be either a subcomponent (e.g. KEYBOARD) or an attribute (e.g. PRICE).

Table 1 shows the aspects that cover this domain. In total there are 30 categories, from which 21 are subcomponents and 9 are attributes.

As explained in the next section, these labels were restructured from the tag set of the SemEval ABSA tasks.

Aspects	
Subcomponents	**Attributes**
BATTERY	CONNECTIVITY
COMPANY	DESIGN_FEATURES
CPU	GENERAL
DISPLAY	MISCELLANEOUS
FANS_ COOLING	OPERATION_ PERFORMANCE
GRAPHICS	PORTABILITY
HARD_DISC	PRICE
HARDWARE	QUALITY
KEYBOARD	USABILITY
MEMORY	
MOTHERBOARD	
MOUSE	
MULTIMEDIA_ DEVICES	
OPTICAL_DRIVES	
OS	
PORTS	
POWER_SUPPLY	
SHIPPING	
SOFTWARE	
SUPPORT	
WARRANTY	

Table 1: List of aspects

There are some tags that are unspecific because they aim to target more general comments that still have a sentiment attached to it and are informative. For example, the attribute GENERAL is used for general appreciations towards the object as in "This laptop is amazing!" or "My Macbook Pro has been a huge disappointment". There is a clear sentiment that refers to the laptop but does not specify what attribute in particular is the one they are referring to. On its part, the MISCELLANEOUS label is used to target expressions that comment on aspects that are not covered by the current tag set. For instance, the observation "It's wonderful for computer gaming" refers to a specific use of the laptop.

Concerning the sentiment, each aspect will have an opinion associated with it that can be either positive, negative or neutral.

4 Corpus

For this project, we will use the laptop review corpus from the 2016 SemEval challenge[1], specifically the one from the ABSA task. This corpus is split in a training set, containing 450 reviews or 2500 sentences, and a test set of 80 texts and 808 phrases.

Each sentence can refer to multiple aspects simultaneously, which accounts for multi-labeled texts. In contrast, it is possible to find reviews that do not allude any of the given aspects, labeled as "OutOfScope", as in "If only Bill Gates would read some of what is said here MS would do a better job", where there's no reference to the laptop itself whatsoever.

As mentioned in the previous section, the annotation schema applied to this corpus differs from our current tag set. In the original 2016 SemEval challenge, the labels formed a pair of entity (subcomponent) and attribute, plus polarity. Given the complexity of this schema, the tag set was simplified to facilitate the subtasks of aspect extraction. From the original entity-attribute pairs, we kept the entities alone, except for the LAPTOP entity, from which we kept only its attributes.

After reorganizing the annotations and removing the unlabeled texts, our corpus ended up with 2039 sentences in total. It is important to note that reviews are split by sentences and that these are labeled individually.

Since our corpus is rather small considering the numerous aspects that are involved, the train and test set were merged and for the machine learning tasks, k-fold cross-validation was applied (with a k parameter of 5) to make our results more reliable.

5 Preprocessing

One of the greatest challenges of product reviews is that it is user generated content. In other words, it is unstructured and noisy. Hence,

[1] http://alt.qcri.org/semeval2016/task5/index.php?id=data-and-tools

Table 1: List of aspects

tokenization and normalization play a big role given that they will ultimately clean and structure the text before the proper IE subtasks.

Concerning normalization, the main actions taken were:

- Escaping of unescaped characters
- Replacing of special characters
- Replacing of non-ASCII characters
- Removal of stop words
- Dealing with contractions
- Expanding common abbreviations (e.g. "w/" to "with")
- Splitting bar delimited synonyms (e.g. "replacement/repair" to "replacement or repair")
- Splitting of hyphened words (e.g. "user-friendly" to "user friendly")
- Taking care of typos (".." instead of "…"; dashes at the end of a word)
- Reducing repeated characters ("sloooow" to "slow")

For misspellings, fuzzy string matching was applied using the *Fuzzywuzzy* Python library that implements Levenshtein Distance to calculate the differences between sequences.

6 Aspect Detection and Extraction

Our first task is aspect detection, also interpreted as a multi-class classification of aspects problem. The goal is to identify the aspects that are alluded in each review. Given the size of the tag set, we implemented a binary Logistic Regression classifier for each aspect that takes as input features the bag-of-words (BOW) of each sentence and its tags. The solver used was "newton-cg", which supports multinomial loss. Table 2 shows the average metrics after performing K-fold cross validation of k=5.

Aspect	**Accuracy**	**Precision**	**Recall**
BATTERY	0.973	0.891	0.697
COMPANY	0.954	0.721	0.603
CONNECTIVITY	0.974	0.572	0.520
CPU	0.990	0.495	0.500
DESIGN_ FEATURES	0.890	0.730	0.612
DISPLAY	0.961	0.857	0.714
FANS_COOLING	0.998	0.499	0.500
GENERAL	0.784	0.732	0.706
GRAPHICS	0.990	0.620	0.536
HARD_DISC	0.983	0.673	0.551
HARDWARE	0.996	0.498	0.500
KEYBOARD	0.972	0.841	0.640
MEMORY	0.992	0.746	0.565
MISCELL.	0.932	0.676	0.589
MOTHERBOARD	0.997	0.498	0.500
MOUSE	0.972	0.819	0.595
MULTIMEDIA_ DEVICES	0.986	0.940	0.649
OPERATION_ PERFORMANCE	0.907	0.818	0.685
OPTICAL_ DRIVES	0.997	0.499	0.500
OS	0.969	0.808	0.639
PORTABILITY	0.976	0.555	0.522
PORTS	0.997	0.499	0.500
POWER_SUPPLY	0.994	0.597	0.533
PRICE	0.957	0.832	0.711
QUALITY	0.894	0.625	0.550
SHIPPING	0.995	0.498	0.500
SOFTWARE	0.963	0.575	0.524
SUPPORT	0.936	0.706	0.601
USABILITY	0.933	0.706	0.615
WARRANTY	0.998	0.499	0.500

Table 2: Aspect detection metrics.

As evidenced, accuracy is very high for all categories, except for GENERAL. This is due to the fact that the corpus is highly imbalanced and that most categories are sparse. Precision ranges from the high-forties to the high-eighties. Aspects with high precision are usually very delimited and can be easily detected, such as BATTERY and PRICE. In contrast, aspects with low precision may overlap among each other, making it difficult for the classifier to classify correctly. For example, WARRANTY and SHIPPING are highly related to SUPPORT, resulting in precisions around 0.49. Finally, recall ranges between 0.5 and 0.711. The aspects with higher recall are, again, the most delimited categories or those in which vocabulary tends to

converge more, such as GENERAL and DISPLAY.

Our second task is to identify the tokens in the text that allude to these categories. These will eventually help us find the opinions associated to each aspect. For this, the frequent noun phrase approach proposed by Hu and Liu (2004) was implemented, enhanced by the application of Positive Pointwise Mutual Information (PPMI).

First a co-occurrence matrix was computed to calculate the PPMI between each word in the vocabulary and each one of the aspects. Then, with the help of the Stanford POS tagger, we identified all noun phrases present in each sentence. To determine which noun phrase referred explicitly to a given aspect, we picked the noun phrase with the highest PPMI value.

Unfortunately, for this task we don't have annotations of the aspect terms, not allowing to rely on any formal metric. To evaluate this task, it was necessary to analyze manually the results. Consider the following sample of associations made between the aspects and the nouns in the sentences.

"I will be considering Lenovo for all future laptop purchases."
COMPANY (Lenovo)

"One positive aspect of this laptop is the keyboard, which is larger than usual."
KEYBOARD (keyboard)

"But now I am having battery problems."
BATTERY (battery)

"I have a little bit of a learning curve with the keyboard shortcuts."
KEYBOARD (keyboard)

"Have not quite figured out how to make the font size larger overall."
DISPLAY (size)

"After doing much research for a laptop that had all the features I was looking for, this one hit all the criteria I had."
DESIGN_FEATURES (feature)

"The keyboard and sound are great, and the matte screen works well."
KEYBOARD (keyboard)
MULTIMEDIA_DEVICES (sound)
DISPLAY (screen)

"I put that free app on for word processing and it works like a charm."
SOFTWARE (app)

"Very good performer for the price."
OPERATION_PERFORMANCE (performer)
PRICE (price)

"I am still exploring the various features but overall I am satisfied with my purchase."
DESIGN_FEATURES (feature)
GENERAL (purchase)

"What a great laptop, I can run my games and work really fast."
GENERAL (laptop)
MISCELLANEOUS (game)
OPERATION_PERFORMANCE (work)

The easier associations to make are those with only one aspect tag and one noun phrase. But in many cases there are one or more aspect tags and multiple noun phrases, which may make it harder. Judging by this sample, PPMI seems to work good enough to make distinctions in presence of several attributes.
(For code, see Appendix A)

7 Sentiment Polarity Detection

This subtask actually consists of two steps: first, identifying the opinions regarding the features found; and second, determining the polarity of these opinions. For the former, we will identify opinion phrases with the help of a keyword opinion list provided by Hu and Liu[2].

To determine the semantic orientation of these opinions we will implement the lexicon-based approach proposed by Ding, Liu and Yu (2008). This consists of basically the following steps:

1) **Mark sentiment words and phrases**: For each sentence that contains one or more aspects, all sentiment word and phrases will be scored with a +1 (positive) or -1 (negative).

[2] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

2) **Apply sentiment shifters**: Identify words and phrases that change the sentiment orientation and score accordingly. For this step, we supplemented a list of negation words and also used a list of verbal sentiment shifters[3].
3) **Handle but-clauses**: The sentiment orientations before and after the contrary word (e.g. 'but') are opposite to each other. This helps determine ambiguous/unknown sentiments.
4) **Aggregate opinions**: Apply opinion aggregation function to determine the final orientation of the sentiment of each aspect according to resulting scores and distance between aspect and opinion

The score function is as follows:

$$score(f) = \sum_{w_i : w_i \in s \wedge w_i \in V} \frac{w_i SO}{dis(w_i, f)}$$

where w_i is an opinion word, V is the set of all opinion words, and s in the sentence that contains the feature (aspect term) *f*, and $dis(w_i,f)$ is the distance between feature f and opionion word w_i in the sentence *s*. w_iSO is the semantic orientation of the word w_i. The multiplicative inverse in the formula is used to give low weights to opinion words that are far away from feature *f*. Results are shown in Table 3.

	Precision	Recall	Accuracy
Positive	0.812	0.621	0.440
Negative	0.791	0.476	
Neutral	0.477	.477	

Table 3: Polarity classification results.

The sentiment polarity detection task reached an accuracy of 44%, a low score probably due to the difficulty of identifying neutral sentiment. However, precision for both positive and negative polarity rendered decent results. This was to be expected given that it is a single label classification problem (one sentiment per aspect) and there are only three categories in play (versus 30 in aspect detection).

[3] https://raw.githubusercontent.com/uds-lsv/lexicon-of-english-verbal-polarity-shifters/master/shifter_lemma_lexicon.csv

(For code, see Appendix B)

8 Conclusions

In this paper we presented an aspect-based sentiment analysis implementation on laptop reviews from the 2016 SemEval. Three subtasks were undertaken with diverse results. Low scores in the aspect detection task can be attributed to three items: i) the number of classes as already mentioned, ii) the size of the data set, which is too small and sparse to train a classifier, and iii) the presence of implicit features, which was not included in the scope of this project. All of these are difficult challenges that will need to be addressed in future work.

With regards to the aspect extraction task, even though results seem promising, metrics are not reliable enough to evaluate its performance. Hence, it will be necessary to work with an annotated dataset to assess it appropriately.

Finally, the sentiment detection task needs special treatment of neutral opinions. It could also be enhanced with the help of a dependency parser and/or WordNet in order to identify opinions that were missed. All in all, it rendered promising results for positive and negative polarity, which shows that a lexicon-based approach still can compete against alternative classifiers for an ABSA task.

9 References

Bhargava, P., Spasojevic, N. & Hu, G. 2017. Lithium NLP: A System for Rich Information Extraction from Noisy User Generated Text on Social Media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text,* pp. 131-139. Retrieved from http://aclweb.org/anthology/W17-4417.

Ding, X., Liu, B., and Yu, P. (2008). A holistic lexicon-based approach to opinion mining. In Proceedings of the Conference on Web Search and Web Data Mining (WSDM-2008).

Gamon, M., Aue, A., Corston-Oliver, S., and Ringger, E. 2005. Pulse: Mining Customer Opinions from Free Text. In: Famili A.F., Kok J.N., Peña J.M., Siebes A., Feelders A. (eds) *Advances in Intelligent Data Analysis VI*. IDA 2005. Lecture Notes in

Computer Science, vol 3646. Springer, Berlin, Heidelberg.

Hu, M. and Liu, B. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, Pages 168-177. Seattle, WA, USA.

Liu, B. 2012. Sentiment Analysis and Opinion Mining, Pages 10-12 and 58-78. Morgan & Claypool Publishers.

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I., and Manandhar, S. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27-35. Dublin, Ireland.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., and Androutsopoulos, I. 2015. SemEval-2015 Task 12: Aspect Based Sentiment Analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 486-495. Denver, Colorado, USA.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., AL-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X, Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S.M. and Eryiğit, G. 2016. SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *In Proceedings of SemEval-2016*, pages 19-30. San Diego, California, USA.

Popescu, A.M. and Etzioni, O. 2005. Extracting Product Features and Opinions from Reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 339-346. Vancouver, British Columbia, Canada.

Thura Thet, T., Na, J., and Khoo, C. 2010. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 36 (6), pages 823-848.

Titov, I. and McDonald, R.. 2008. A Joint Model of Text and Aspect Ratings for Sentiment Summarization. In *Proceedings of ACL*, pages 308-316. Columbus, Ohio, USA.

Zhuang, L., Jing, F. and Zhu, X. (2006). Movie Review Mining and Summarization. CIKM '06 Proceedings of the 15th ACM international conference on Information and Knowledge Management, pp. 43-50. Retrieved from http://delivery.acm.org/10.1145/1190000/1183625/p43-zhuang.pdf?ip=129.64.122.171&id=1183625&acc=ACTIVE%20SERVICE&key=7777116298C9657D%2E41475A04D4A330A4%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1552430137_36a9ded9569283015cd2073ce615391c

10 Appendix A

```
def get_X(train, id_map, features, categories,
cat_dict):
'''Computes X matrix for training classifier'''

    matrix = np.zeros((len(train),
    len(features)))
    for i in id_map:
        id = id_map[i]
        #print(type(id))
        for j in range(len(features)):
            text = train[id]['text']
            for token in text:
                if token in features[j]:
                    matrix[i][j] = 1
    return matrix


def get_Y(train, id_map, category):
'''Computes Y matrix for training classifier'''

    # label matrix
    matrix = np.zeros(len(train),)

    for i in id_map:
        id = id_map[i]
        opinions = train[id]['opinions']
        for opinion in opinions:
            cat = opinion['category']
            if category == cat:
                matrix[i] = 1

    return matrix


def train_aspect_classifiers(train, id_map,
categories, feat_dict, solver):
'''Trains aspect classifier'''

    # training classifier
    avg_scores = defaultdict(dict)
    classifiers = defaultdict(dict)
    for cat in categories:
        features = feat_dict[cat]
        x = get_X(train, id_map, features,
                categories, feat_dict)
        y = get_Y(train, id_map, cat)
        kf =
        model_selection.KFold(n_splits=5,
        shuffle=False)
        classifier =
        LogisticRegression(random_state=0,
        solver=solver, max_iter=100).fit(x, y)
        classifiers[cat]['model'] = classifier
         score =
        model_selection.cross_val_score(classif
        ier, x, y, cv=kf)
        classifiers[cat]['score'] = score
         scoring = ['accuracy',
        'precision_macro', 'recall_macro']
         metrics = ['test_accuracy',
        'test_precision_macro',
        'test_recall_macro']
         scores =
        model_selection.cross_validate(classifi
        er, x, y, scoring=scoring, cv=5,
        return_train_score = False)

        for met in metrics:
                    avg_scores[cat][met] =
        scores[met].mean()

    return classifiers
```

```
def feature_liaison(reviews, noun_dict,
adj_dict, ppmi_matrix, word_keys, cat_keys):
'''Associates a feature (noun phrase) to each
identified aspect'''

    for id, review in noun_dict.items():
   opinions = reviews[id]['opinions']
        features = review['text']
        absa_tuples = []
        cat_feat_ppmi = defaultdict(dict)
        for feature in list(features):
            feature = lem.lemmatize(feature)
            cat_list = []
            for op in opinions:
                cat = list(op.values())[0]
                cat_list.append(cat)
                cat_ind = cat_keys[cat]
                if feature in word_keys.keys():
                    feat_ind=word_keys[feature]
                    cat_feat
                =ppmi_matrix[feat_ind][cat_ind]

                cat_feat_ppmi[cat][feature]=
                cat_feat

            for cat in cat_list:
                        max_word=
        max(cat_feat_ppmi[cat], key=lambda k:
        cat_feat_ppmi[cat][k])

                for op in opinions:
                    if op['category'] == cat:
                                op['feature'] =
                                max_word

    return reviews
```

11 Appendix B

```
# Sentiment Shifters
negation = ['no', 'not',
            'without', 'never',
            'none', 'nobody',
            'nothing', 'neither',
            'nowhere', 'hardly',
            'barely', 'rarely','scarcely']

contrast = ['but', 'however',
            'unfortunately',
            'though', 'although',
            'despite', 'yet',
            'nonetheless', 'nevertheless',
            'still', 'except']


def score_function(id, review, lemma_dict, feature, adj_dict, positive_lexicon, negative_lexicon, neg_words, verbal_shifters):
'''Implements the score function to all reviews'''

    feat_score = 0
    for adj in adj_dict[id]['text']:
        adj = lem.lemmatize(adj)
        text = review['text']
        w_sent_score = sentiment_score(text, adj, lemma_dict, positive_lexicon, negative_lexicon, neg_words, verbal_shifters)
        dist = opinion_feat_dist(review['text'], lemma_dict, feature, adj)
        if dist != 0:
            feat_score += (w_sent_score/dist)

    return feat_score


def sentiment_score(text, adj, lemma_dict, positive_lexicon, negative_lexicon, neg_words, verbal_shifters):
'''Computes sentiment score for each opinion word, applying sentiment shifters'''


    adj_lem = adj
    lemmas_adj = lemma_dict[adj]
    for lem_adj in lemmas_adj:
        if lem_adj in text:
            adj = lem_adj

    adj_ind = text.index(adj)
    prev_window = text[(adj_ind-2):adj_ind]

    # checking for shifters
    shift = False
    for token in prev_window:
        if token in neg_words or token in verbal_shifters:
            shift = True

    if len(prev_window) > 0:
        if prev_window[0] in neg_words or verbal_shifters:
            if prev_window[1] in neg_words or verbal_shifters:
                shift = False

    sent_score = 0
    if shift is False:
        if adj_lem in positive_lexicon:
            sent_score = 1
        elif adj_lem in negative_lexicon:
            sent_score = -1
    else:
        if adj_lem in positive_lexicon:
            sent_score = -1
        elif adj_lem in negative_lexicon:
            sent_score = 1

    # BUT-HANDLING: when no sentiment was marked
    if sent_score == 0:
        but_ind = -1
        for word in range(len(text)):
            if text[word] in contrast:
                but_ind = word

        # if adj is after but clause
        clause_score = 0
        if adj_ind < but_ind and but_ind > 0:
            text = text[but_ind+1:]
            for word in range(len(text)):
                clause_score += sentiment_score(text, text[word], lemma_dict, positive_lexicon, negative_lexicon, neg_words, verbal_shifters)
            sent_score = clause_score*(-1)

        elif adj_ind > but_ind and but_ind > 0:
            text = text[:but_ind]
            for word in range(len(text)):
                clause_score += sentiment_score(text, text[word], lemma_dict, positive_lexicon, negative_lexicon, neg_words, verbal_shifters)
            sent_score = clause_score*(-1)

    return sent_score


def opinion_feat_dist(text, lemma_dict, feature, adj):
    '''Distance between an opinion and a feature in a given text'''

    lemmas = lemma_dict[feature]
    for lem in lemmas:
        if lem in text:
            feature = lem

    lemmas_adj = lemma_dict[adj]
    for lem_adj in lemmas_adj:
        if lem_adj in text:
            adj = lem_adj
    dist = text.index(feature) - text.index(adj)

    return abs(dist)


def sent_extraction(reviews, lemma_dict, adj_dict, pos_lex, neg_lex, neg_words, verbal_shifters):
'''Determines sentiment for each review'''

    for id, rev in reviews.items():
        for op in rev['opinions']:
            if 'feature' in op.keys():
                feature = op['feature']
                score = score_function(id, reviews[id], lemma_dict, feature, adj_dict, pos_lex, neg_lex, neg_words, verbal_shifters)
                op['score'] = score
                if score > 0:
                    op['sentiment'] = 'positive'
                elif score < 0:
                    op['sentiment'] = 'negative'
                else:
                    op['sentiment'] = 'neutral'
    return reviews
```