Cristal Algorithm

Final Project Submission
Course Name: **NLP**
Course Number: **CS8761**
Date: **Dec 17, 2002**

Key Idea

Adjectives and adverbs in the definitions of a word match with those in the **context** that the word tends to occur in.

Generateing good and bad word lists

In our algorithm we read the list of good words from the seed file. Then using the information in LDOCE and BigMac, we extract all the words that share both, a semantic code (LDOCE) and a thesaurus entry (BigMac), with at least one of the words in the list. We call this **good list**. Similarly we extract **bad list** too. These lists are also submitted in files “goodList” and “badList” respectively.

Getting good and bad definitions

From LDOCE, we next extract the definitions of all the good words that are associated with the parts of speech “adj” and “adv” and put them is a pool. We call these **good definitions**. Similarly, we retrieve **bad definitions** too.

Generating good and bad context

To generate **good context**, we extract all the words from good definitions, whose at least 1 part of speech is “adj” or “adv”. LDOCE is used here. Similarly, from bad definitions we generate **bad context**.

Support for Key Idea

We derived the above mentioned key idea from a set of experiments, which we performed on a few arbitrary words that have some semantic orientation. For each of the words, we found all its definitions (from LDOCE) that are associated with the parts of speech “adj” and “adv”. We, then, extracted all the adjectives and adverbs from these definitions. We call the total number of adjectives and adverbs thus extracted, the size of the definitions of the word and we denote it by S. We then count the number of adjectives and adverbs in this set that match with those in our good context (explained above). We call this number “**good matches**” M_g. Similarly, we compute “**bad matches**” M_b.

As per our key idea, we should get, for a good word, $\mathbf{M_g > M_b}$ and for a bad word, $\mathbf{M_g > M_b}$**.** We scale these values with ‘S’, so as to avoid the impact of definition length. Therefore for good words we have $\mathbf{(M_g/ S) > (M_b / S)}$ and vice-versa for a bad word. This is what we observe in our experiments, whose results are tabulated below.

Word	**M_g/ S**	**M_b / S**
Awesome	**8/10**	**7/10**
Excellent	**6/6**	**4/6**
Good	**57/107**	**40/107**
Great*	**49/52**	**24/52**
Superb	**2/3**	**1/3**
Terrific	**9/10**	**7/10**
Awful	**7/12**	**8/12**
Bad*	**21/56**	**55/56**
Horrible	**3/8**	**6/8**
Painful	**6/13**	**8/13**
Poor*	**20/36**	**34/36**

Getting good and bad sentences from web

After getting good and bad contexts, we retrieve, good and bad sentences from web. For this we make use of good and bad lists. E.g. to generate list of good sentences, we take a word from the good list, retrieve sentences from the web that contain this word and put them in the pool of good sentences. Then we repeat the process for next word and so on. Similarly, we generate a list of bad sentences from the web.

Tagging Reviews

Once the above mentioned book keeping is done we are set to tag the reviews. We first extract the words from a given review that have some sentiment information content. To do this we count the number of occurrences of every word in the review in the list of good and bad sentences retrieved from web. We consider a word relevant if it occurs considerably more in one list than the other. E.g.

Say, the list of **good sentences** contains **140,000 words**, whereas, that of the **bad sentences** contains **180,000 words**.

Say, a word "**camera**" occurs **726 times** in the **good sentences** and **959 times** in **bad sentences**. Similarly, let these values respectively be **583** and **137** for the word "**perfect**".

We calculate the following values for every word and if the final score is greater than 1, we accept it else we reject it.

$$\text{Good Ratio } (R_g) = \frac{\text{Good Matches}}{\text{Number of words in good sentences}}$$

$$\text{Bad Ratio } (R_b) = \frac{\text{Bad Matches}}{\text{Number of words in bad sentences}}$$

$$\text{Score} = \frac{\text{abs } (R_g - R_b)}{\text{smaller } (R_b, R_g)}$$

For “camera”,

$$R_g(\text{camera}) = (726 / 140{,}000) = 0.0052$$
$$R_b(\text{camera}) = (959 / 180{,}000) = 0.0053$$
$$\text{Score (camera)} = \frac{\text{abs } (0.0053 - 0.0052)}{\text{smaller } (0.0053, 0.0052)} = 0.02$$

Similarly, for “perfect”,

$$R_g(\text{perfect}) = (583 / 140{,}000) = 0.0041$$
$$R_b(\text{perfect}) = (137 / 180{,}000) = 0.00076$$
$$\text{Score (perfect)} = \frac{\text{abs } (0.0041 - 0.00076)}{\text{smaller } (0.00076, 0.0041)} = 4.39$$

Thus, “camera” with score = 0.02 < 1 is rejected, while “perfect” with score 4.39 > 1 is accepted, which is what we want because the word “camera” does not contain any sentiment information, whereas the word “perfect” does.

Assigning scores to the relevant words

We next compute a score for every relevant word in the review. For this we make use of our good and bad context, previously generated.

For each word, we extract its definitions that correspond to the parts of speech “adj” and “adv”. From these definitions, we extract only the words that are adjectives and adverbs. We call this set of words AA-Set (Adjectives-Adverbs Set). Let its size be S. We count the number of elements from this set that match with the words in our good context. We call this count good match (Mg). Similarly, using bad context we compute bad match (Mb). Once the values of Mg, Mb and S are known, we compute the score for that particular word as

$$\text{Score (word)} = (Mg / S) - (Mb / S)$$

We normalize with S, so as to nullify the effect of greater number of matches in longer definitions. We this algorithm, we generate negative scores for the words with negative sentiment and positive for the ones with positive sentiment.

Scoring and tagging the review

Finally, the score of the whole review is the sum of scores of its individual relevant words. We tag a review as “positive” if its score > ***K***, “negative” if its score < ***-K*** and neutral otherwise. This ***K*** is an arbitrary constant. We experimented with ***K*** = 0.5, 0.3 and 0.

We got very good results for camera data, for smaller values of ***K***, which can be seen in our experiments file.