Fine Grained Sentiment Analysis

Mohit Kaduskar, Nafisa Jassani
Department of Computer Science, University of Houston
mgkaduskar@uh.edu, njassani@uh.edu

Abstract. Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics. We are given a set of labeled aspect specific sentiment expressions across variety of review sentences. The aim of the project is to develop a system that can find new sentiment expression spans on unseen sentences of the same aspect. The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organizations across the world.

I. Introduction

Sentiment is a positive or negative attitude, held by a person or a group of people, that is directed at some person, product or organization.

The aim of this paper is to find negative sentiment phrases called issues from a dataset consisting of customer reviews. We are given a dataset that consists of labeled issue phrases in reviews given by customers with respect to various products on Amazon.com. We have data on variety of aspects like cord, jack, wire of various products like earphones, keyboard, etc. These are known as head aspects.

Given a sentence, ss = ($w1$, … wn), we need to discover the head aspect (HA/issue subject), w(HA)=i and a sub-sequence (wp, ...wq), $p \leq i \leq s$ that best describes the issue, i.e., an aspect specific opinion phrase on the head aspect and containing the head aspect. Positive opinion phrases are relatively easier to discover, issues are comparatively more challenging to detect as they can be both direct or indirect. 'The cord length was short' is an example of direct issue whereas 'the cord could not reach my pocket' is example of an indirect issue phrase. Our system learns from the data and tries to determine if the reviews consists of issues in the testing phase. It also aims to find the issue boundary and separate the issue phrases from the rest of the sentence.

Our system is divided into 2 phases:

1. Given a head aspect (HA), detect whether a sentence containing the HA mentions an issue.
2. Given a HA and a sentence mentioning an issue, extract the issue phrase boundary.

Besides the word, POS, phrase chunk tags and word polarity act as important features in determining if the sentence mentions an issue or no and also in separating the issue phrase boundary from the rest of the sentence.

As product issues are directly reflected in the language usage, word and POS (W+POS) n-gram features serve as natural baselines.

Pivot Features: We consider five feature families which take on a set of values:

i. POS Tags (TT): DT, IN, JJ, MD, NN, RB, VB, etc.
ii. Phrase Chunk Tags (CC): ADJP, ADVP, NP, PP, VP, etc.
iii. Prefixes (PP): anti, in, mis, non, pre, sub, un, etc.
iv. Suffixes (SS): able, est, ful, ic, ing, ive, ness, ous, etc.
v. Word Sentiment Polarity (WW): POS, NEG, NEU

The features can be in-depth explained as follows:

Part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e., its relationship with adjacent and related words in a phrase, sentence, or paragraph.

Example: ('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),

('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')

Text chunking divides a text into syntactically correlated parts of words called phrase chunk tags.

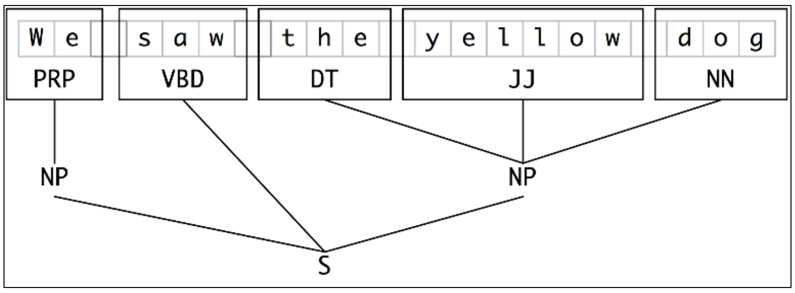


Diagram: Sample phrase chunk tag tree representation

Prefix and Suffix denote the additional characters that appear at the start or end of a root word.

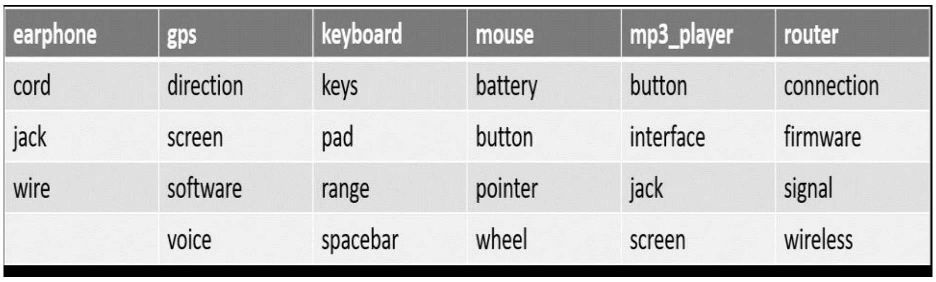
Example: (‘Unfortunately’, ‘ly’)

Word Sentiment polarity classifies each word as positive, negative or neutral. Examples of positive words are: nice, awesome, cool, superb whereas bad, uninspired, expensive, disappointed are examples of negative polarity words. The words not given positive or negative are assigned as neutral.

Example: (‘good’, ‘Positive’), (‘low’, ‘negative’).

II. Data Set

The data set consists of reviews of customers given on Amazon.com. The data is divided in various folders each of a particular product. And each folder consists of files on a variety of aspects (referred to head aspects). Sample data structure of the given data set is as follows:



earphone	gps	keyboard	mouse	mp3_player	router
cord	direction	keys	battery	button	connection
jack	screen	pad	button	interface	firmware
wire	software	range	pointer	jack	signal
	voice	spacebar	wheel	screen	wireless

Here each column represents a folder, where column heading is the folder name. The various rows in each folder represent the various files.

For each domain, we report the head aspects and the counts as (xx/yy) where xx is the # of issue sentences and yy the total # of sentences in that domain: Router (1284/5063; connection, firmware, signal, wireless), GPS (632/2075; voice, software, screen direction), Keyboard (667/1446; spacebar, range, pad, keys), Mouse (494/2488; battery, button, pointer, wheel), MP3-Player (174/352; button, interface, jack, screen), Earphone (359/678; cord, jack wire). This dataset serves both of our tasks I and II.

III.Related Work

For task 1, we initially used high frequency words and their POS tags for classification into issue and non-issue sentences as explained in [1]. It yielded high accuracy results of about 70%, but words alone cannot determine the features expressing fine -grained sentiments.

Inorder to classify if the sentences contain issue or no, we make use of the following algorithms:

1. Naïve Bayes Classifier - Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

   $P(C / X) = P(C) * P(X / C) / P(X)$

   Where C is the expected class label, X is a vector containing features.

2. Linear SVC: It is another implementation of Support Vector Classification for the case of a linear kernel.

3. Multinomial Naïve Bayes: estimates the conditional probability of a particular word/ term/ token given a class as the relative frequency of term t in documents belonging to class c:

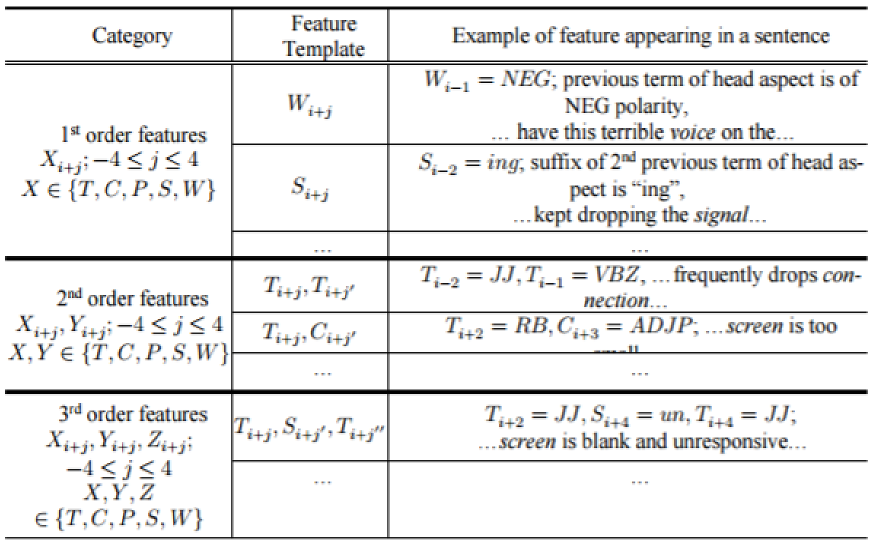
$$P(t \mid c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

4. Bernoulli Naïve Bayes: In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs.

5. Logistic regression: This algorithm measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

In [2], Professor Arjun Mukherjee makes use of word, POS, phrase, prefix, suffix and word polarity as features. But by observation we concluded that suffix and prefix do not improve the feature set for the given data, but in fact reduce the accuracy obtained for training the data using various classifiers.

In [2], Professor also mentions 1st, 2nd and 3rd order features as shown in the table below:



Category	Feature Template	Example of feature appearing in a sentence
1st order features $X_{i+j}; -4 \leq j \leq 4$ $X \in \{T, C, P, S, W\}$	W_{i+j}	$W_{i-1} = NEG$; previous term of head aspect is of NEG polarity, ... have this terrible *voice* on the...
	S_{i+j}	$S_{i-2} = ing$; suffix of 2nd previous term of head aspect is "ing", ...kept dropping the *signal*...
	...	...
2nd order features $X_{i+j}, Y_{i+j}; -4 \leq j \leq 4$ $X, Y \in \{T, C, P, S, W\}$	$T_{i+j}, T_{i+j'}$	$T_{i-2} = JJ, T_{i-1} = VBZ$, ...frequently drops *connection*...
	$T_{i+j}, C_{i+j'}$	$T_{i+2} = RB, C_{i+3} = ADJP$; ...*screen* is too
	...	...
3rd order features $X_{i+j}, Y_{i+j}, Z_{i+j}$; $-4 \leq j \leq 4$ X, Y, Z $\in \{T, C, P, S, W\}$	$T_{i+j}, S_{i+j'}, T_{i+j''}$	$T_{i+2} = JJ, S_{i+4} = un, T_{i+4} = JJ$; ...*screen* is blank and unresponsive...
	...	...

Table 2: Pivot Feature Templates. The subscript *i* denotes the position of the issue subject (HA) which is italicized and the subscript *j* denotes the position relative to *i*.

We have however reduced the feature set since the entire featureset file consisting of all 3 order of features produced very huge model files and took a lot of processing time. We thus reduced the featureset template to only 1st and 2nd order.

In [2], the author also describes the various approaches used for separating the issue phrase from the rest of the sentence. He makes use of Unsupervised Heuristic Baseline (UHB), Sequence Modeling, Linear Chain Features and Phrase Structural Constraints. His results prove that CRF provides better accuracy than UHB.

To train our data we have also made use of CRF in which we train the model given the training file consisting of tuples in which each tuple consists of word, POS, prefix, suffix and word polarity. At the end of each tuple we also have a possible outcome – B, I, O ie. if the word forms the beginning, inside content or outside the issue phrase.

He also states making use of 5-fold cross validation to segregate the training and testing data. We have tried 10-fold cross validation where the entire issue dataset is broken down into 10 parts, where 1 part is used for testing and the remaining 9 parts are used for training the model.

IV. Method

Task I: Given a head aspect we need to determine if the sentence containing the head aspect mentions an issue or no.

To determine if the sentence contains an issue or not, we take a frame from -4 of head aspect to +4 of head aspect. We pass the word, POS and word polarity (positive, negative or neutral) of the words in the frame as features. These features are then used to train the model.

This task is divided into various sub-tasks:

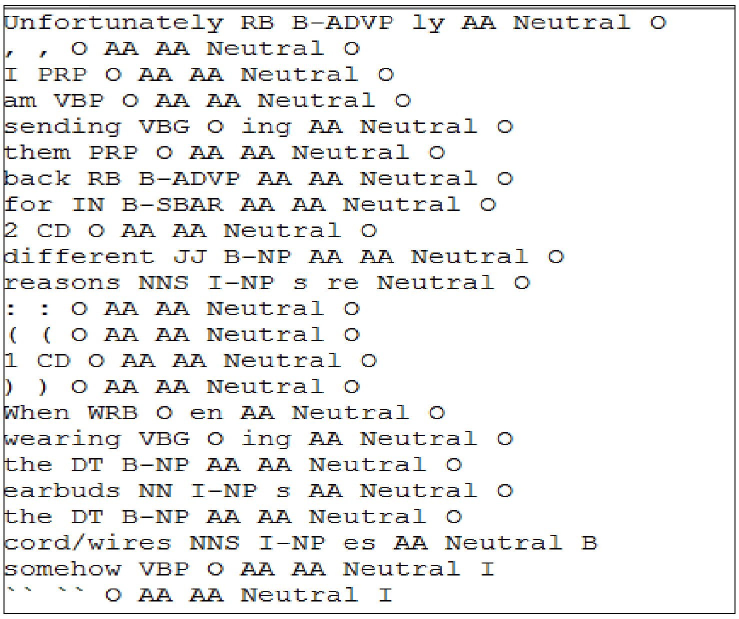
1. We combine files from a folder (ie. for a particular product) and form a single file. We store the head aspects in a list. These head aspects are later used when we need to form the frames of words from -4 to +4 of the head aspect.
2. Data pre-processing: Remove [VBZ, VBD tags] [ADJ, ADV tags] present at the end of the sentences in the dataset. Remove <i> tags (while creating training and test data) used to indicate the issue phrases and the punctuations present in the sentences.
3. We then define a frame from -4 to +4 of the head aspect and find features of the data.
    i. To find the POS Tags in a sentence we make use of POS tagger. A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word. To find the POS tag we make use of pos_tag functionality present in nltk. [3]
    ii. To find the phrase chunk tags we make define regular expressions for the various phrases such as noun phrase, verb phrase, etc. We then pass the expressions to RegexpParser (inbuilt in NLTK) to find the various chunk phrases.
    iii. To find the various prefix and suffixes we define the commonly used prefixes and suffixes, the list is obtained from [5]. We then check if a word ends or starts with the given suffix or prefix.
    iv. To define the word polarity, we make use of [6]. This serves as a dictionary inorder to identify the word polarity.
4. We use the 10-fold cross validation method to segregate the training and testing data since they are initially not separated.
5. The training data is then passed to the various classifiers and then test the accuracy of each of them.
6. We combine the various algorithms by creating a sort of voting system, where each algorithm gets one vote, and the classification that has the highest vote is the chosen one.

Task II: Given a HA and a sentence mentioning an issue, extract the issue phrase boundary.

For this phase, we create the training, testing files and the file containing a combination of feature patterns which are used to implement the CRF algorithm.

This task is divided into the following sub-tasks:

1. The first three sub-tasks of this task remain same as that of Task I.
2. We then generate a file containing all the words and their corresponding POS, phrase chunk tags, suffix, prefix and word polarity and the expected tag – B, I or O. Here, B denotes the beginning of an issue phrase, I denotes words inside the issue phrase and O denotes words which are not a part of issue phrase. Sample input file is as follows:



```
Unfortunately RB B-ADVP ly AA Neutral O
, , O AA AA Neutral O
I PRP O AA AA Neutral O
am VBP O AA AA Neutral O
sending VBG O ing AA Neutral O
them PRP O AA AA Neutral O
back RB B-ADVP AA AA Neutral O
for IN B-SBAR AA AA Neutral O
2 CD O AA AA Neutral O
different JJ B-NP AA AA Neutral O
reasons NNS I-NP s re Neutral O
: : O AA AA Neutral O
( ( O AA AA Neutral O
1 CD O AA AA Neutral O
) ) O AA AA Neutral O
When WRB O en AA Neutral O
wearing VBG O ing AA Neutral O
the DT B-NP AA AA Neutral O
earbuds NN I-NP s AA Neutral O
the DT B-NP AA AA Neutral O
cord/wires NNS I-NP es AA Neutral B
somehow VBP O AA AA Neutral I
`` `` O AA AA Neutral I
```

Sample input file containing the word, POS, phrase, suffix, prefix, polarity and the final tag indicating if the word is in the beginning, inside or outside the issue phrase.

3. We generate file with all possible sequences of the features and generate their combinations of one and two attributes. This file serves as a template when training the data using CRF.
4. We train the data using the following algorithms:
5. We then find the precision, recall and F1 score for each of the labels.

V. Experiment and Results

For task 1, we combined several features for all words in the frame from -4 to +4 positions of the head aspect. We tried various combinations of features using word, POS, suffix, prefix and polarity.

The approximate accuracy obtained when all features were used for Naïve Bayes and other algorithms was between 50-60%. We observed that word, POS and polarity serve as the best features for classification. We also implemented various classification algorithms like Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Logistic Regression, SGD Classifier, Linear SVC and NuSVC and observed that Naïve Bayes, Linear SVC classifier, Multinomial Naïve Bayes classifier, Bernoulli Naïve Bayes classifier and Logistic Regression classifier to be the best ones.

We have also implemented voters system to find the best tag for a particular classification set. We also implement 10-fold cross validation to find the average accuracy. One sample result obtained for the earphone domain is as follows:

Average Original Naive Bayes Algorithm accuracy percent: 63.2472191786708

Average MNB classifier accuracy percent: 63.085706855868146

Average Bernoulli NB classifier accuracy percent: 62.01592594233724

Average Logistic Regression Classifier accuracy percent: 62.64162752570011

Average Linear SVC Classifier accuracy percent: 62.748859675270964

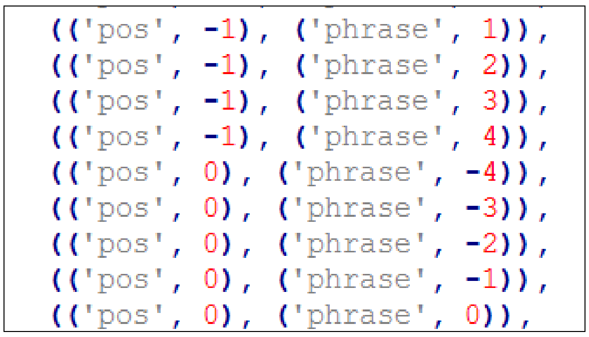
Average voted Classifier accuracy percent: 62.77546154562284

Detailed results can be obtained by observing the separate file of Results included in this folder. Following is the table showing the accuracy results for various classification algorithms across various domains:

Algorithm\Product	Earphone	GPS	Keyboard	Mouse	MP3 Player	Router
Naive Bayes	63.24	75.097	69.034	76.095	68.79	77.32
MNB classifier	63.085	77.101	68.69	82.87	67.84	79.76
Bernoulli NB	62.015	75.46	67.99	82.31	67.82	79.18
Logistic Regression	62.64	81.60	72.01	84.69	65.45	80.16
Linear SVC	62.74	79.93	70.99	81.99	63.66	79.17
Voted classifier	62.77	78.93	69.03	83.67	67.55	80.081

For task 2, we trained the CRF Suite using multiple features; we also used combination of 1, 2 and 3 features to form the templates. We observed that combination of word, POS, phrase and polarity when taken as features provide the best results. We also use only a combination of 1 and 2 features as templates.

Sample feature set is as shown below:



```
(('pos', -1), ('phrase', 1)),
(('pos', -1), ('phrase', 2)),
(('pos', -1), ('phrase', 3)),
(('pos', -1), ('phrase', 4)),
(('pos', 0), ('phrase', -4)),
(('pos', 0), ('phrase', -3)),
(('pos', 0), ('phrase', -2)),
(('pos', 0), ('phrase', -1)),
(('pos', 0), ('phrase', 0)),
```

Sample features present in chunking.py file

We also tried shuffling the training and testing data used for implementation of CRF Suite using 10-fold cross validation. The sample results obtained for keyboard domain are as follows:

Holdout group: 10

***** Epoch #57 *****

Loss: 245.279993

Improvement ratio: -0.008014

Feature L2-norm: 7.503838

Learning rate (eta): 0.000717

Total number of feature updates: 34029

Seconds required for this iteration: 1.837

Performance by label (#match, #model, #ref) (precision, recall, F1):

O: (1194, 1322, 1246) (0.9032, 0.9583, 0.9299)

B: (38, 56, 66) (0.6786, 0.5758, 0.6230)

```
I: (158, 200, 266) (0.7900, 0.5940, 0.6781)
Macro-average precision, recall, F1: (0.790583, 0.709336, 0.743656)
Item accuracy: 1390 / 1578 (0.8809)
Instance accuracy: 31 / 66 (0.4697)
```

The above output is obtained for 10th holdout when processing the file using 10-fold cross validation; we obtain similar results for each of the 10 holdouts.

By default, the CRFSuite makes use of ‘lbfgs’ algorithm as mentioned in [4], we also tested the model by tuning its various parameters. By tuning the -a parameter we change the algorithm used while training the data using CRFSuite. The different algorithms we tried are lbfgs and l2sgd. The results obtained for earphone and gps domain are as follows:

(Precision, Recall, F1)

Algorithm\Product	Earphone	GPS
L-BFGS with L1/L2 regularization	(0.787, 0.519, 0.589)	(0.87, 0.696, 0.76)
SGD with L2-regularization	(0.82, 0.65, 0.714)	(0.86, 0.714, 0.77)

VI. Conclusion:

Through this project, we classified if the sentence contained an issue or no and we also tried to segregate the issue from the rest of the sentence. We used 10-fold cross validation to segregate the train and test data. We used algorithms like Naïve Bayes, Linear SVC classifier, Multinomial Naïve Bayes classifier and Bernoulli Naïve Bayes to classify if the sentence contained an issue or no.

For task 2, we identify the issue phrase from the rest of the sentence. This issue phrase is given by the words tagged with labels B and I which mark the beginning and inside words of the issue phrase. To identify the issue phrases, we made use of CRF Suite.

VII. References.

[1] - https://pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/

[2] – Extracting Aspect Specific Sentiment Expressions implying Negative Opinions by Professor Arjun Mukherjee

[3] - http://www.nltk.org/book/

[4] - http://www.chokkan.org/software/crfsuite/

[5] - http://teacher.scholastic.com/reading/bestpractices/vocabulary/pdf/prefixes_suffixes.pdf

[6] - http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar