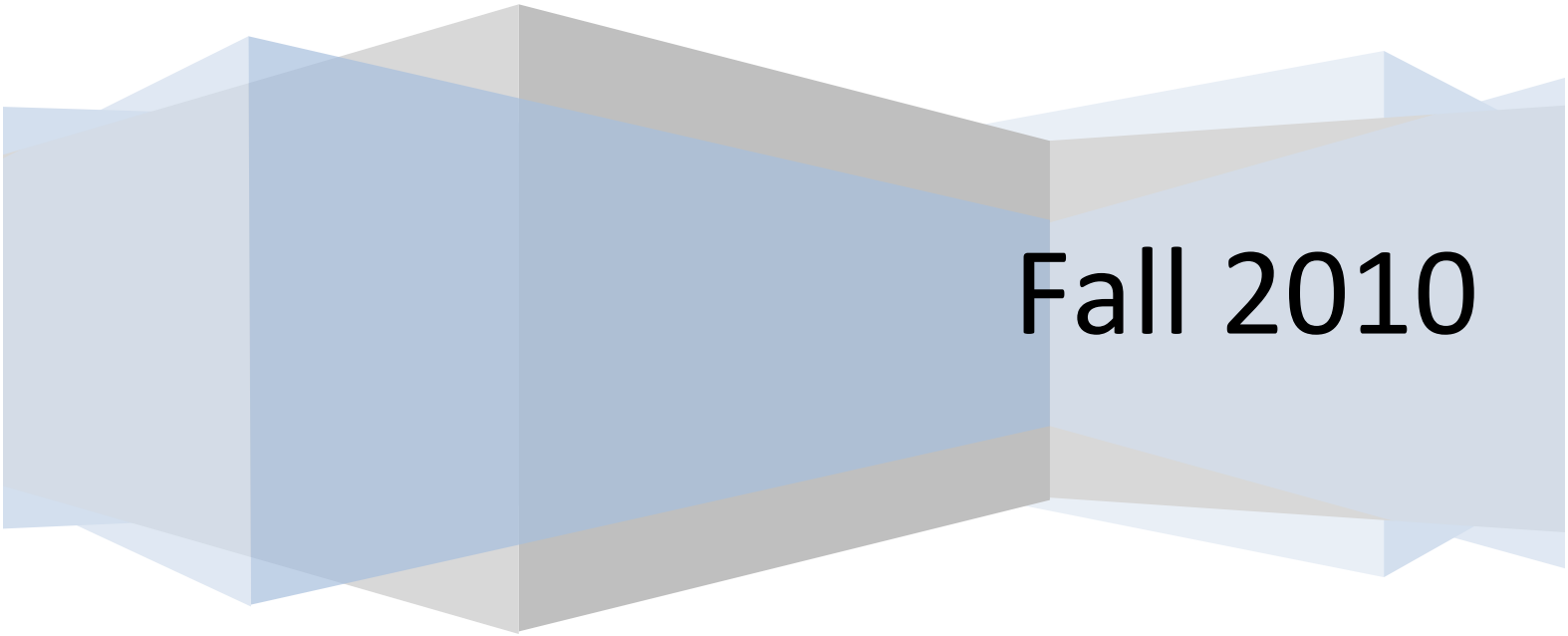
Computer Linguistic Course

Final Project on Opinion Mining /Story2Game Translator

Instructor: Professor Cercone

Razieh Niazi



Fall 2010

Table of Contents

Chapter 1: Opinion Mining

Opinion Mining- Sentiment Analysis

Introduction

As the Web grows wider, finding information on the Web is much more difficult. People rely on search engines to find relevant information. The question is what kind of information we can look for through web engines? The fact is that we are only able to find keyword-based information. Recently, with the growth of social sites, blogs and forum, with the pervasive access of users to the Internet and vast variety of products and services available on the web, a new paradigm has been introduced called Opinion Mining.

In opinion mining, we are looking for people opinions about any product, service or any other things. The goal in opinion mining is to mine opinions expressed in user's generated contents like review sites, forums, social networks, discussion groups, blogs and so on.

There are two types of evaluation: Direct opinion and comparisons. In Direct opinion, sentiment expressions are expressed on some objects like products, services, events and so on. This itself is categorized to several opinion search queries: 1) Find the opinion of an opinion holder like a person or organization on a topic or object, 2) find positive/negative opinions on a particular object.[1]

Finding opinions on an object is difficult. Indeed, search engines are not able to handle these kinds of queries. This is because of the following reasons:

- Ambiguity in natural language. People express their opinion in natural language. Natural language processing is highly ambiguous, and vague. It has ambiguity in all levels such as lexical, syntactic, semantic, referential, programmatic…
- Background knowledge: We understand other people talks because we have background knowledge about their world.
- Semi-structure or unstructured data format: the users' generated contents are typically represented as HTML, Text or XML.
- People express their opinions in various styles: There is not any rule to express our opinions. We can express our opinions in many different ways. See the following example. It represents different opinions from different reviewers about camera:

  Reviewer 1: I want to start off saying that this camera is small for a reason.

  Reviewer 2: I'm in high school, and this camera is perfect for what I use it for, carrying it around in my pocket so I can take pictures whenever I want to, of my friends and of funny things that happen. The only thing I don't like is the small size (8 MEG) memory card that comes with it. I have to move pictures off of it every day so I have room for more pictures the next, and I don't have enough money to buy the 256 MEG card that I've had my eye on for a while.

  Reviewer 3: got this as it flew off the shelves.

  Review 4: This is the only camera with all these capabilities.

So far, many researches have been conducted on direct opinion mining. The researches have been both at document (review) and sentence level. However, nothing has been done on Sentiment classification with regard to the feature's weight based on the user's input. The fact is that features of an object do not have the same weight for the reviewer. Look at the following example:

Reviewer: Camera is so small and light, I can carry it everywhere. I like it but I do not like its battery.

How can we evaluate this review? The reviewer likes the size of the camera but s/he does not like its battery. What would be the sentiment analysis result at sentence level? Positive or Negative?

One of the best strategies is to give weights to the features. By giving weight, we can highlight the importance of some features upon other features. Specifying weights for features is totally a user-dependent process.

In this project, I introduce a new paradigm to mine direct opinions and classify them as positive, negative or neutral. The sentiment analysis is done with respect to the feature's weight specified by the user. I also propose a new way to extract features and sentiment analysis. In these approaches, I employ four different indicators in four levels including Clause level, Phrase level, Word-level, and feature-level. I apply indicators in decision rules to mine interesting patterns. By applying these indicators, I am getting a high accurate result in both feature extraction and sentiment analysis.

The rest of this report is organized as follows: First, I will talk about related work and how my work is different from the existing work. Then, I present the proposed solution. I demonstrate the experimental result in Section. Section presents evaluation and finally in Section we have conclusion and future work.

Related Work

Several works have been done for sentiment classification. These works are done at either document levels (reviews) or sentence level.

In [1], the overall sentiment expressions of opinion holders (authors) are analyzed. In this works, data is reviews from epinions.com on automobiles, banks, movies and travel destinations. The approach has three steps:

- Part-of-Speech (POS) tagging , building a bi-gram model from reviews if it conforms specific patterns like JJ NN. (JJ Adjective, NN Noun)
- Use PMI to estimate semantic orientation (SO) of the extracted phrases
- Classify the review by getting the average of SO

The other work is done by [2]. In this work, they use PMI to obtain syntactic relations and other attributes with SVM.

[3] uses Naïve Bayesian classifier with a set of data features/attributes extracted from training sentences

A bootstrapping approach is used in [8]. In this work a precision classifier is first used to automatically identify some subjective and objective sentences. A set of patterns are then learned from these identified subjective and objective sentences. Syntactic templates are provided to restrict the kinds of patterns to be discovered,

e.g., <subj> passive-verb. The learned patterns are then used to extract more subject and objective sentences (the process can be repeated).

In [9], for subjective or opinion sentence identification, three methods are tried:

- Sentence similarity.
- Naïve Bayesian classification.
- Multiple naïve Bayesian (NB) classifiers.

For opinion orientation (positive, negative or neutral), it uses a similar method to [1], but with more seed words (rather than two) and based on log likelihood ratio (LLR). For classification of each word, it takes the average of LLR scores of words in the sentence and use cutoffs to decide positive, negative or neutral.

[4] uses sum up orientations of opinion words in a sentence. In this work, first sentences that have both topic phrase and holder candidates are selected. Then, the holder-based regions of opinion are delimited. Using POS tagger, adjectives, verbs and nouns are selected and their polarities are calculated. Finally the system combine them to obtain the holder's sentiment for the whole sentence. For evaluating sentiment words, authors set a small amount of seed words by hand in two categories: positive and negative. Then, they grow it by adding words by looking at the synonyms obtained from WordNet . To classify a new word, they use conditional probability:

$$\arg\max_{c} P(c \mid w)$$
$$\cong \arg\max P(c \mid syn_1, syn_2 .....syn_n)$$

Where w is unseen words, c is the class either positive or negative and Syn_i is the Wordnet synonym of w. To extract nouns, adjs and verbs they use POS and unigram models.

Finding clause or phrase polarities based on priori opinion words and classification are proposed in [5]. This approach has two steps. In the first step, it classifies each phrase as neutral or polar. In the second step, it takes all steps marked in step one as polar and disambiguities their contextual polarity. In this paper, word context is a bag of three words. Then, it moves the sentence to phrase level and make a dependency parse tree. Every node in the tree structure is a surface node. There are no abstract node line VP and NP. There are 5 forms of features which is evaluated in this paper: Word features, Modification Features, Sentence features, structure features and document features.

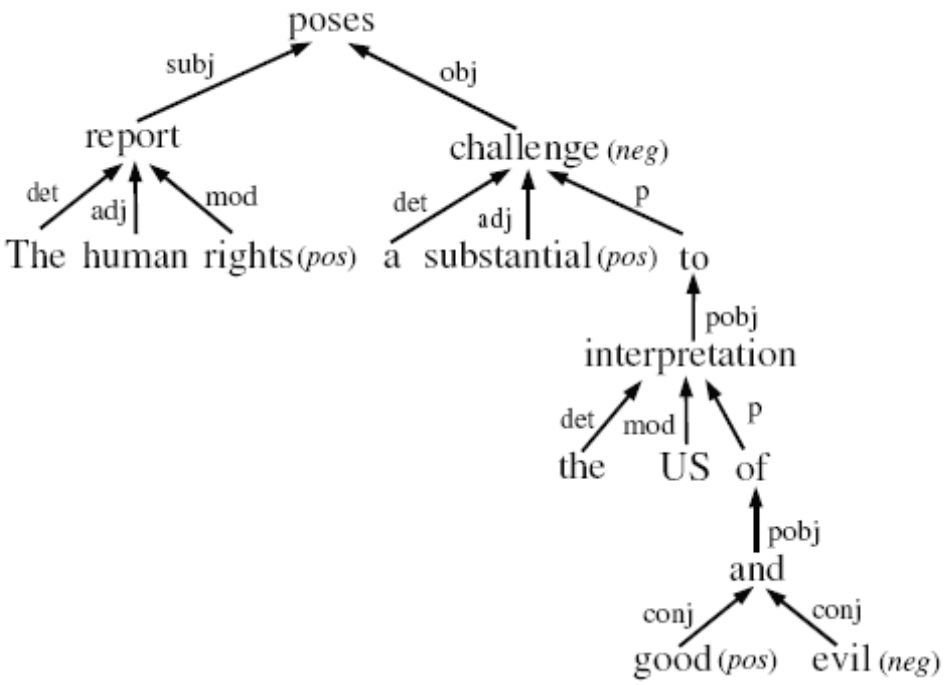


Figure 1: [5] http://www.cs.pitt.edu/~wiebe/pubs/papers/emnlp05polarity.pdf

The existing systems mostly use POS tagging, n-gram models and particular patterns. Some papers like[5] use a bag of 4 words. There are several problems with these approaches:

- The accuracy of model is increased with a higher degree of n. However, by increasing n in an n-gram model the number of parameters also increased. For instance if we have a vocabulary of 20,000 words, for building a bi-gram model we needs 20000*19999 = 400 million parameters.

Model	Parameters
1st order (bigram model):	$20,000 \times 19,999 = 400$ million
2nd order (trigram model):	$20,000^2 \times 19,999 = 8$ trillion
3th order (four-gram model):	$20,000^3 \times 19,999 = 1.6 \times 10^{17}$

Table 6.1 Growth in number of parameters for *n*-gram models.

Figure 2: n-gram parameters [10]

For this reason, approaches currently use bi-gram or tri-gram. It is hard to keep up with seven, or eight gram. The other issue is extracting interesting patterns from n-gram models. Finding interesting patterns is totally depends on the n in n-gram model. Consider for instance the following instance:

The only thing I don't like is the small size 8 MEG memory card that comes with it

In this sentence, we are interested in the following words (specified with Bold):

The only thing I **don't like** is the **small size** 8 MEG **memory card** that comes with it

Using n-gram model, what n do we need to extract all interesting words in one sequence?

My work is totally different from these works in several ways. I use Penn Treebank and parse the sentence that is pretty fast. I employ 4 indicators including Clause level, Phrase level, Word level and Feature level in decision rules. I apply feature's weight in classification. My algorithm works in any users' generated content (free text). I use back propagation technique to fix unrecognized patterns.

Proposed Solution's Objectives

The objective of this project is summarized as follows:

1) Classifying reviews is done in sentence level. The classification result estimated on each sentence is positive, negative or neutral.
2) The sentiment analysis is done on any arbitrary data from user's generated contents like forums, review sites, social networks sites, and blogs. The data can be represented in any way. It does not have to be in a specific format like Pros and Cons.
3) The sentiment analysis is sentence-level and feature based. Extracting interesting features play important roles in the result.
4) Sentiment analysis process uses Penn Treebank and employs 4 indicators in parallel to estimate the sentiment orientation of the sentences. These indicators are defined in Clause level, Phrase level, Word level and Feature level.
5) The training and testing algorithms is done on product reviews corpus.
6) For Dataset, I use data sets provided and used in [6]. These datasets were used for sentiment analysis. It is a feature- labeled corpus and has information on product review like canon, Diaper, iPod, Router and MP3.
7) For mining opinion words, I used Corpus-based approach. The corpus has negative and positive reviews in a very specific format for electronics and DVD. Datasets for each review (positive or negative) are represented in two different files. In order to use these datasets, I have to do several preprocessing tasks. Figure 4 shows the sample data for each review.
8) Employing weights for features in sentiment classification. Weights are entered by the user. Indeed, they are user's independent.

1 electronics_negative.review

gaps:1 well:1 it_together:1 a_stack:1 the_cd:1 bottom:1 are_gaps:1 cd:3 constips:1 a_cd:1 to_fit:1
save_your:2 steady_on:1 save:2 picture:1 your_money:2 over:1 a_rug:1 nice:1 advice:1 over_very:1
i:2 slightest_smudge:1 nice_for:1 errors:1 player_doesn't:1 the_sound:1 any_price:1 useless:1 if_there
but_i:1 two:1 i:2 even:1 without:1 one:1 inexpensive_cd:1 cd_cases:1 breaking_into:1 break_instantly
failure:1 people_should:1 my_software:1 and_save:1 implying:1 learn:1 the_firmware:1 they_mention:1
i:9 bang_for:1 because_out:1 well:1 because_i:1 <num>_bucks:1 discs_at:1 that_burned:1 burned_go
the_failure:1 failure:1 burns:1 using_the:1 wasted:1 cost:1 on_burning:1 get_one:1 more_the:1 succe
live:2 movies_just:1 it's_crap:1 i_rarely:1 found:1 if:1 horrid:1 rate_:1 0-50%:1 stutter:1 hope_it's:1 r
failure:1 problems_for:1 both:1 in_terms:1 with_*both*:1 others_it:1 the_reviews:1 reviews:1 proved_
knew:1 say_whoops:1 beware:1 this_was:1 whoops_a:1 cash:1 true...and:1 contacting:1 too:1 only_
only_it's:1 promised:1 this_purchase:1 asking_price:1 original:1 it's_definitely:1 i_was:1 paperwork_yet
pay:1 binding_is:1 rating_given:1 has_discouraged:1 tinstaafl:1 the_flap:1 and_sleeve:1 cost:2 so:1 o
i_would:1 cd/dvd_cases:1 collection:1 this_brand:2 over_the:1 so:1 many:1 cd/dvd:1 worst_quality:1
product_packages:1 confidence:1 always_so:1 amazon.one_mistake:1 say_i'm:1 once_i:1 use_it:1 so:
i:5 shipping:1 shipping_so:1 for_other:1 the_hand.the:1 shift:1 poorly:1 i_couldnt:1 nice_asthe:1 othe
i_can:1 about:1 about_<num>:1 product:2 say:1 spent:1 i've_only:1 i_would:1 never_recommend:1 i'v
before_i:2 must:1 ads:3 advertised:1 a_product:1 having_trouble:1 buy_another:1 with_tech:1 maybe
would_not:1 work:1 this_sd:1 gb:1 product_did:1 kingston:1 and_sandisk:1 product:1 sd:2 fuji_and:1
i:1 older:1 attempting_to:1 however_they:1 only_work:1 replacing_however:1 same_room:1 they_wou
extend:1 pretty_much:1 well:2 look:1 that_arrived:1 be_happier:1 four_of:1 <num>_bucks:1 internet_
but_when:1 any_longer.i:1 wore_off:1 i_was:1 longer.i:1 was:1 off:1 of_months:1 i_do:1 first:1 i_don'
jnr:1 reader:1 a_lemon:1 kodak_card:1 on_multiple:1 unit:1 from_jnr:1 computer_fails:1 detect:1 this_
you_put:1 books_on:1 stand_that:1 the_stand:1 unit_is:1 the_description:1 if_you:1 the_whole:1 des
flimsy_product:1 will_just:1 a_heavy:1 parts_snap:1 i'm:2 i'm_afraid:1 flimsy:2 to_put:1 plastic_and:1

1 electronics_positive.review

the_failure:1 area:1 my_cable:1 shut:1 equipment_due:1 spike:1 minutes_this:1 and_lcd:1 down_e
to_work:1 breif:1 usually_an:1 month_now:1 no:1 conditioning_usually:1 good_for:1 power_adapter
always:1 warning:2 to_tell:1 sounds:2 is_happening:1 the_unit:2 lot:1 like_a:1 what_is:1 single:1 s
protect_your:1 probably_satisfy:1 choice:1 on_your:1 company:1 breakage_these:1 cases_depend
one_can:1 always:1 recently_according:1 sd_card:2 amazon_for:1 product:1 very:2 problem_one:1
paid_i'd:1 wore_out:1 shell_and:1 was_a:1 nicely.so:1 i_paid:1 case:1 large_collection:1 cd's_and:
card_review:1 goes_for:1 value:1 great:1 kingston:2 kingston_1gb:1 value_for:1 great_value:1 this
wild:1 use_it:1 sd_card:1 trail:1 very:2 game:1 camera_for:1 buy_kingston:1 pictures_and:1 i_wou
muc:1 card:2 had_no:1 you_muc:1 delivery:1 you:1 problems:1 was_prompt...thank:1 no:1 i_had:1
cf:1 olympus_and:1 it_seems:1 expert:1 it_takes:1 my:2 though:2 card:3 going_through:1 though_
get-aways_and:1 respectable_i've:1 breeze_pretty:1 traveling_with:1 alot:1 use_it:1 not_bose:1 o
recomend_this:1 and_just:1 reader:2 attached_to:1 my_camera:2 cables:2 case:1 picture:1 i_coul
long_as:1 about:1 device_so:1 lit:1 a_bright:1 glow:1 it_other:1 say:1 about_it:1 as_long:1 a_simp
after_reading:1 about:1 thinking_about:1 the_reviews:1 case:1 lot:1 product:1 buying_a:1 <num>_
i:2 products:1 sandisk_products:1 highly_recommend:1 sandisk:2 very_pleased:1 fan:1 sandisk_fan
somewhere:1 replacement_parts:1 unrelated:1 linked_to:1 first_the:1 only_<num>:1 the_volume:1
i_can:1 shutter_on:1 rebel_xt:1 lag_or:1 my_camera:2 or_slow:1 cf_card:1 rate_of:1 no:1 frame:1
gb:1 pro:1 and_reliable..well:1 ultra:1 worth_the:1 worth:1 stick_pro:1 ii:1 fast:1 <num>_gb:1 gb_
and_while:1 ii_is:1 ultra_ii:1 was_fine:1 my_camera:1 quick:1 i_needed:1 the_ultimate:1 down_the:
i_returned:1 secure_digitial:1 was_not:1 me:1 mb:1 ultra:1 (sdsdh-512-901):1 to_me:1 ii:1 item:3
great:1 download:1 nicely_:1 the_job:1 nice_product:1 recommend_this:1 definitely:1 card:1 memc
ones:1 have_several:1 products:1 than_my:1 sandisk_products:1 i:1 than:1 old_ones:1 them_are:1
add_on:1 about:1 is_great:1 the_extra:1 without_worrying:1 sdsdh-1024-901:1 love_to:1 for_your

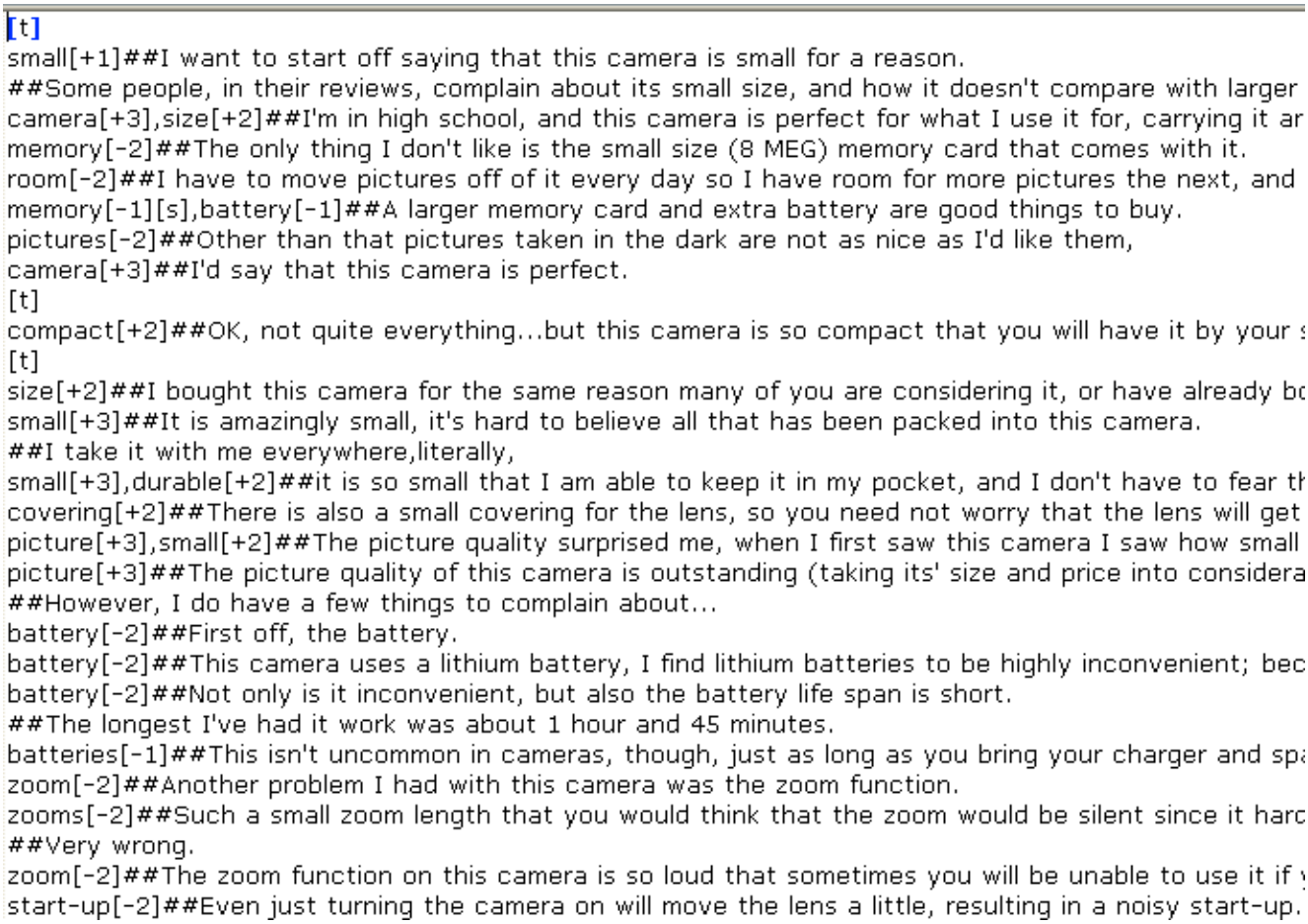
Figure 4: Corpus used for extracting opinion words

Corpus

For this project, I use two types of corpus:

For customer reviews, I use data sets provided and used in [6]. It is used for sentiment analysis. It is a feature-labeled corpus and has information on product review like canon, Diaper, iPod, Router and MP3. It contains two files:

- Unlabeled Corpus from Reviews on Canon S100
- Labeled Corpus from Reviews on Canon S100.



```
[t]
small[+1]##I want to start off saying that this camera is small for a reason.
##Some people, in their reviews, complain about its small size, and how it doesn't compare with larger
camera[+3],size[+2]##I'm in high school, and this camera is perfect for what I use it for, carrying it ar
memory[-2]##The only thing I don't like is the small size (8 MEG) memory card that comes with it.
room[-2]##I have to move pictures off of it every day so I have room for more pictures the next, and
memory[-1][s],battery[-1]##A larger memory card and extra battery are good things to buy.
pictures[-2]##Other than that pictures taken in the dark are not as nice as I'd like them,
camera[+3]##I'd say that this camera is perfect.
[t]
compact[+2]##OK, not quite everything...but this camera is so compact that you will have it by your s
[t]
size[+2]##I bought this camera for the same reason many of you are considering it, or have already bo
small[+3]##It is amazingly small, it's hard to believe all that has been packed into this camera.
##I take it with me everywhere,literally,
small[+3],durable[+2]##it is so small that I am able to keep it in my pocket, and I don't have to fear th
covering[+2]##There is also a small covering for the lens, so you need not worry that the lens will get
picture[+3],small[+2]##The picture quality surprised me, when I first saw this camera I saw how small
picture[+3]##The picture quality of this camera is outstanding (taking its' size and price into considera
##However, I do have a few things to complain about...
battery[-2]##First off, the battery.
battery[-2]##This camera uses a lithium battery, I find lithium batteries to be highly inconvenient; bec
battery[-2]##Not only is it inconvenient, but also the battery life span is short.
##The longest I've had it work was about 1 hour and 45 minutes.
batteries[-1]##This isn't uncommon in cameras, though, just as long as you bring your charger and spa
zoom[-2]##Another problem I had with this camera was the zoom function.
zooms[-2]##Such a small zoom length that you would think that the zoom would be silent since it harc
##Very wrong.
zoom[-2]##The zoom function on this camera is so loud that sometimes you will be unable to use it if y
start-up[-2]##Even just turning the camera on will move the lens a little, resulting in a noisy start-up.
```

Figure 4: the labeled corpus from reviews on Canon S100. Labels were done manually by authors

For building opinion words dictionary, I use another corpus demonstrated in Figure 4. It contains two files:

- Electronic Positive Reviews for building positive opinion words
- Electronic Negative Reviews for building a dictionary of negative opinion words

Datasets are noisy. I have to perform several preprocessing tasks.

The Proposed solution

In sentiment analysis, the main idea is to extract features and determine whether the opinion on the features is positive, negative or natural. Then, provide an opinion summary of multiple features in a sentence. Based on the tasks described above, the sentiment analysis process is done in the following steps:

a) Preprocessing Data

Customer reviews corpus which contains data from Reviews on Canon S100 is very noisy. I performed the following tasks:

- Remove punctuation marks
- Convert words to lowercase

For building opinion words dictionary, I use other corpus demonstrated in figure 4. The problem with this corpus is that the current data is useless. In order to get it work, I have done the following tasks:

- Tokenize
- Remove stop words
- Build a term-frequency based dictionary
    - Example:
        - perfect:32 (negative words dictionary)
        - Perfect: 272 (positive words dictionary)

- Serialize
  In order to load the dictionary faster, I serialize the dictionary objects.

b) Feature Extraction

The objective of feature extraction is to extract features from online product reviews. There are two common formats available on the Web: 1) Pros and Cons format and 2) Free format. The objective of the proposed task is to extract features from free text. An example of such review is shown in the following figure. The features have been specified in Bold:

I want to start off saying that this **camera** is small for a reason. Some people, in their reviews, complain about its small **size**, and how it doesn't compare with larger cameras.

I'm in high school, and this camera is perfect for what I use it for, carrying it around in my pocket so I can take **pictures** whenever I want to, of my friends and of funny things that happen.

The only thing I don't like is the small size (8 MEG) **memory card** that comes with it. I have to move pictures off of it every day so I have **room** for more pictures the next, and I don't have enough money to buy the 256 MEG **card** that I've had my eye on for a while.

A larger memory card and extra **battery** are good things to buy. Other than that **pictures** taken in the dark are not as nice as I'd like hem, I'd say this camera is perfect.

The **PhotoStitch software** is very cool if you want to do any 360 degree **panorama shots**.

Figure 5: Features extracted from users' generated contents

The following diagram shows the proposed solution to extract features:

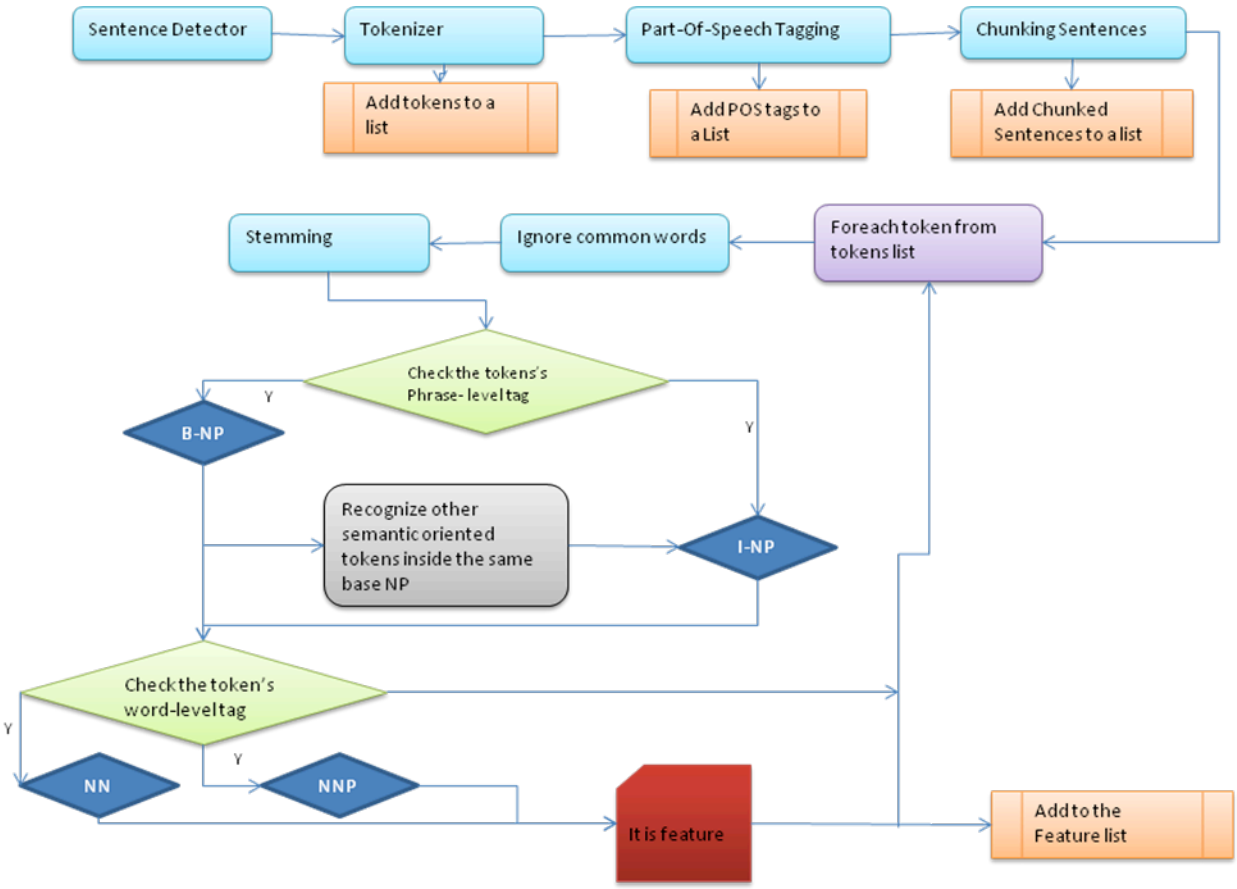


Diagram 1: Proposed algorithm for feature extraction

As demonstrated in Diagram 1, in order to extract features, given a free style text, sentences are detected and words are tokenized. The next step is Part-Of-Speech tagging. POS tagging is the process of making up the words related to a particular part of speech. The most common POS tags in English in the Word-level are: noun, verb, adjective, adverb, pronoun, preposition, conjunction and interjection.

In order to obtain semantic orientation of words and find out how words are inter-related together, I employ chunkers. Chunkers are the process of analyzing a sentence identifying constituents. The chunking process is done in three levels: Clause-level, Phrase level and Word level. For Feature extraction, I only use Phrase level and Word level. But for sentiment analysis, I use all three levels as well as feature level as four indicators to specify the position of each word on the word's context dimension.

For each token recognized in the sentence, the following task is done to extract features:

1. If token is a common words, ignore the token, go to the next token
2. Perform stemming. **stemming** is the process for reducing inflected (or sometimes derived) words to their stem, base or root form . A stemming algorithm reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish". This is necessary, because for example, the "Camera" and "cameras" or "Pictures" and "Pictures" introduce the same feature.

3. Use chunker list created in Chunking process, to extract phrases. Table 1 shows a list of phrase tables from Penn Treebank Tags:

A	B
ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
FRAG	Fragment.
INTJ	Interjection. Corresponds approximately to the part of speech tag UH
LST	List marker. Includes surrounding punctuation.
NAC	Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.
NP	Noun Phrase.
NX	Used within certain complex NPs to mark the head of the NP.
PP	Prepositional Phrase.
PRN	Parenthetical.
PRT	Particle. Category for words that should be tagged RP.
QP	Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Vereb Phrase.

Table 1 : Chunk tags in Phrase level

In the phrase level, I am only interested in Noun phrase (NP) for feature extraction. In phrase level in chunker list, each phrase contains one or more words. These words are semantically related together. In chunking NP, the first word in a Noun phrase (NP) is marked as "B-NP" and the other word inside the same phrase (base NP) is marked as "I-NP". The following shows an example:

Figure 7 shows a screen shot from the project output:

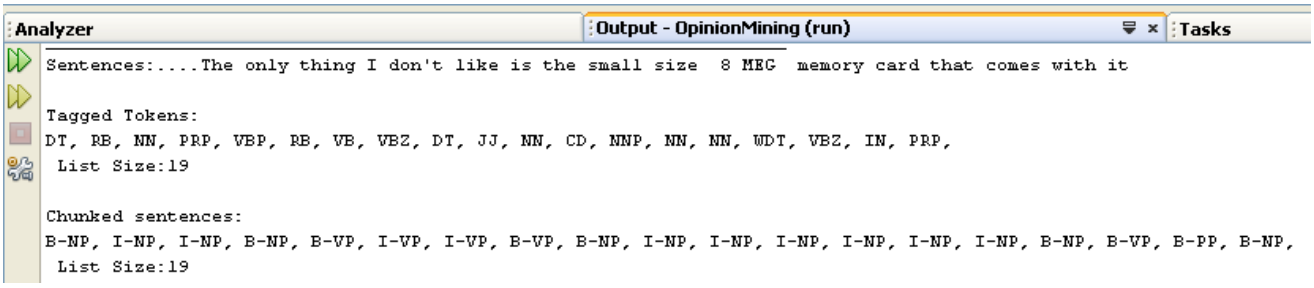


Figure 7: A screen shot of the program's output (for feature extraction)

4. For the tokens in the same base NP, I am only interested in extracting tokens whose POS tags are either NN or NNP. The extracted tokens are features.

c) Sentiment analysis algorithm

Sentiment analysis is done using four indicators:

- Chunking in Clause level
- Chunking in Phrase level
- Chunking in Sentence level
- Feature-level. Features extracted already as described in Section b.

The reason for selecting 4 indicators to perform sentiment analysis is because of the ambiguity levels in natural language. In spite of some existing algorithm which either use review text in a specific format like Pros and Cons or have some limitations in sentences, this algorithm works on every user s' generated content (Free format), and because of the difficulty layers I have to add extra layers to resolve issues.

I use 4 indicators as I already explained. The proposed algorithm is able to identify words which are semantically oriented in a sentence. I use clause level by chunking the sentences

S	simple declarative clause, i.e. c	word and	verb inversion.				
SBAR	Clause introduced by a (possibly	empty)	subordinating conjunction.				
SBARQ	Direct question introduced by a	word or a	phrase. Indirect questions and relative clauses shou				
SINV	Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal						
SQ	Inverted yes/no question, or m	question,	phrase in SBARQ.				

Table 2: Chunking tags in Clause level

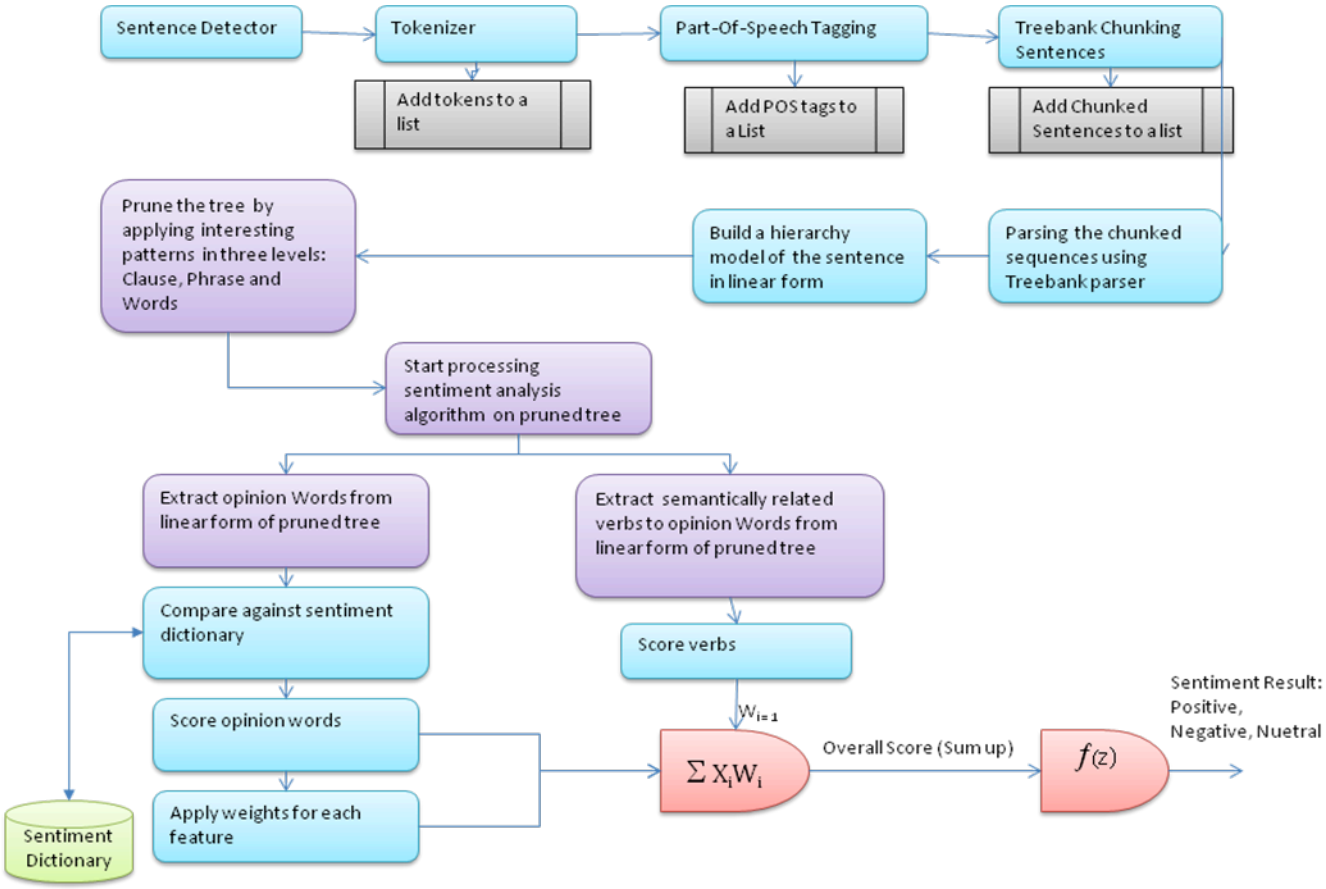


Diagram 2: The proposed sentiment analysis algorithm

The following demonstrates the processing tasks to perform sentiment analysis:

1. The process starts from detecting and tokenizing sentences. Figure 8 shows several sentences example detected by sentence detector module:

Simple Sentences:

- I want to start off saying that this camera is small for a reason

Complicated sentences:

- Some people in their reviews complain about its small size and how it doesn't compare with larger cameras
- I have to move pictures off of it every day so I have room for more pictures the next and I don't have enough money to buy the 256 MEG card that I've had my eye on for a while
- The picture quality surprised me, when I first saw this camera I saw how small it was an instantly assumed that the picture quality would not be good--but I was wrong!
- This camera uses a lithium battery; I find lithium batteries

Figure 8: Some sample sentences detected by sentence detector module

2. Every detected sentence is given to the tokenizer module as input and the output is an array of tokenized words in order.
3. Given the sentence and tokenized words, POS tagging is performed and a POS tag is assigned to each tokenized word. Figure 9 shot shows part-of-speech tags for each sentences:

Simple Sentences: I want to start off saying that this camera is small for a reason

```
Tagged Tokens:
PRP, VBP, TO, VB, IN, VBG, IN, DT, NN, VBZ, JJ, IN, DT, NN,
 List Size:14
```

Complicated sentences: This camera uses a lithium battery; I find lithium batteries to be highly inconvenient; because what if you are on vacation, where the nearest place to buy batteries is just a gas station--there is no way that you are going to find lithium batteries there.

```
Tagged Tokens:
DT, NN, VBZ, DT, NN, NN, PRP, VBP, NN, NNS, TO, VB, RB, JJ, IN, WP, IN, PRP, VBP, IN, NN, WRB, DT, JJS, NN, TO, VB, NNS, VBZ, RB, DT, NN, NN, RB, VBZ,
 List Size:46
```

Figure 9: POS tags for each sentence generated by POS tagging module

4. Given the POS tags and tokenized words, Penn Treebank chunking is performed on the sentences. Table 1 and 2 show the detailed description about tags. Figure 10 shows sample example generated from the chunker module:

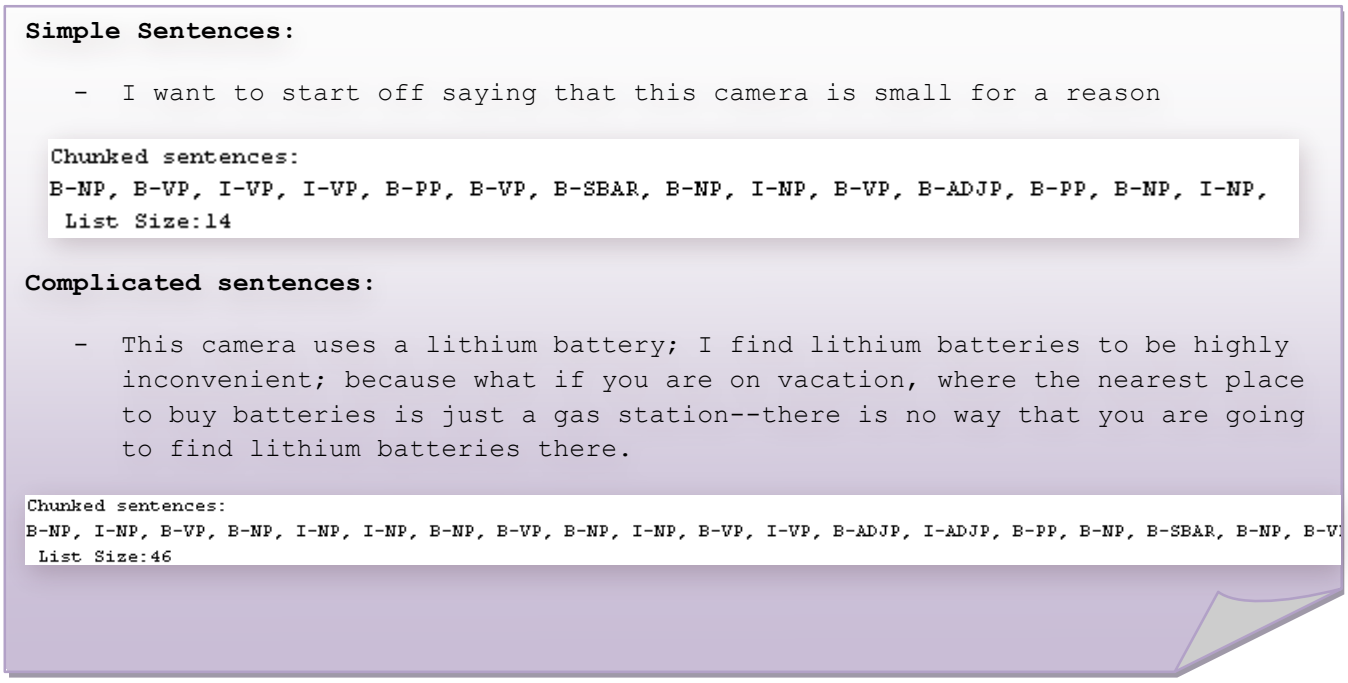


Figure 10: Chunks list generated by Chunker module

5. The next step is parsing chunked sentences. By pursing chunked sentences, we make a hierarchical
tree for each sentence. The tree is represented in a linear form.

Diagram 3 demonstrates how the parsed tree look likes [7].

```
(S (NP (NNP John))
   (VP (VPZ loves)
       (NP (NNP Mary)))
   (. .))
```

Diagram 3: Parsed tree

The following figures show the generated results from the parser module (Treebank parser):

Simple Sentences:

- I want to start off saying that this camera is small for a reason

```
TreeBank1....(TOP (S (NP (NP (DT Some) (NNS people)) (PP (IN in) (NP (PRP$ their) (NNS reviews)))) (VP (VBP complain) (PP (IN ab
TreeBank2....(TOP (S (NP (NP (DT Some) (NNS people)) (PP (IN in) (NP (PRP$ their) (NNS reviews)))) (VP (VBP complain) (UCP (PP (I
```

Complicated sentences:

- This camera uses a lithium battery; I find lithium batteries to be highly inconvenient; because what if you are on vacation, where the nearest place to buy batteries is just a gas station--there is no way that you are going to find lithium batteries there.

```
TreeBank1....(TOP (S (NP (DT This) (NN camera)) (VP (VBZ uses) (NP (DT a) (NN lithium) (NN battery)) (S (NP (PRP I)) (VP (VBP find
TreeBank2....(TOP (S (NP (DT This) (NN camera)) (VP (VBZ uses) (NP (DT a) (NN lithium) (NN battery)) (SBAR (S (NP (PRP I)) (VP (VB
```

Figure 11: Parsing sentences generated by Treebank parser

6. The next step is pruning the linear- format tree. This is done by applying interesting patterns on three indicators containing Clause level, Phrase level and Word level computed in the previous steps. Figure 12 shows how irrelevant words are pruned from the parse tree:

Sentence: The quick brown fox jumps over the lazy dog.

```
Sentences:....The quick brown fox jumps over the lazy dog

Tagged Tokens:
DT, JJ, JJ, NN, VBZ, IN, DT, JJ, NN,
 List Size:9

Chunked sentences:
B-NP, I-NP, I-NP, I-NP, B-VP, B-PP, B-NP, I-NP, I-NP,
 List Size:9
TreeBank1....(TOP (NP (NP (DT The) (JJ quick) (JJ brown) (NN fox) (NNS jumps)) (PP (IN over) (NP (DT the) (JJ lazy) (NN dog)))))
TreeBank2....(TOP (S (NP (DT The) (JJ quick) (JJ brown) (NN fox)) (VP (VBZ jumps) (PP (IN over) (NP (DT the) (JJ lazy) (NN dog)))))

Stack:
(, (, NP, (, NP, (, ), (, JJ, quick, ), (, JJ, brown, ), (, NN, fox, ), (, NNS, jumps, ), ), (, (, ), (, NP, (, ), (, JJ, lazy, )

 , (, NN, dog, ), ), )

   List Size:43
```

Sentence2: I want to start off saying that this camera is small for a reason.

```
Sentences:....I want to start off saying that this camera is small for a reason

Tagged Tokens:
PRP, VBP, TO, VB, IN, VBG, IN, DT, NN, VBZ, JJ, IN, DT, NN,
 List Size:14

Chunked sentences:
B-NP, B-VP, I-VP, I-VP, B-PP, B-VP, B-SBAR, B-NP, I-NP, B-VP, B-ADJP, B-PP, B-NP, I-NP,
 List Size:14
TreeBank1....(TOP (S (NP (PRP I)) (VP (VBP want) (S (VP (TO to) (VP (VB start) (PP (IN off) (S (VP (VBG saying)
TreeBank2....(TOP (S (NP (PRP I)) (VP (VBP want) (S (VP (TO to) (VP (VB start) (PRT (RP off)) (S (VP (VBG sayin(
            (SBAR (IN that) (S (NP (DT this) (NN ca
           ;) (SBAR (IN that) (S (NP (DT this) (NN c

Stack:
(, (, S, (, NP, (, ), ), (, VP, (, VBP, want, ), (, S, (, VP, (, ), (, VP, (, VB, start, ), (, (, ), (, S, (, VP,

(, VBG, saying, ), (, (, ), (, S, (, NP, (, ), (, NN, camera, ), ), (, VP, (, VBZ, is, ), (, ADJP,

(, JJ, small, ), ), (, (, ), (, NP, (, ), (, NN, reason, ), ), ), ), ), ), ), ), ),

 List Size:89
```

Figure 12: Pruned tree (all irrelevant words are pruned)

7. Now that we have the pruned tree, sentiment analysis is just started. It starts from the inner sentence (tag S), looking for interesting pattern in phrase level like NP and VP and in the word level like "(,NN, word,) " and "(,NNP, word,)". Then, it compares the word against the features list. If it is a feature, it looks for verb phrase and adjective phrase. If it is a negative verb, it gives a score of -1, otherwise 1 to the verb. The same is done for adjectives. It compares adjective against the opinion words dictionary. If the word is in a negative dictionary and has a higher term frequency than positive dictionary, a score of -1 is given to the adjective; otherwise 1. The total opinion score is the sum of verb and adjective scores.

Sentence: I want to start off saying that this camera is small for a reason.

```
Interesting patterns:
camera,
 List Size:1


is,
 List Size:1


small,
 List Size:1
Sum of Adj:    1
Sum of opinion:   2
```

Figure 13: extracting interesting patterns from the pruned tree

8. Classification is done based on the following function:

$$f(z) = \begin{cases} \text{Positive} & Z>0 \\ \text{Neutral} & Z=0 \\ \text{Negative} & Z<0 \end{cases}$$

Z in this function is sun of opinions for one sentence.

d) Produce summary of the opinions

Features weights easily can be applied in sentiment analysis process. Given a weight W_i to the feature i, the formula is changed as follows:

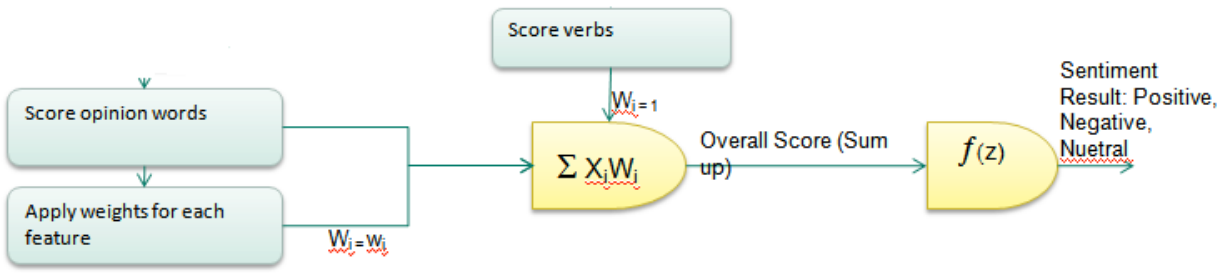


Figure 14: Applying weights in formula

In order to apply weights, a score of -1 is given to negative verbs and a score of 1 is given to positive verbs. A weight of 1 is also given to the verbs.

Regarding the opinion words, a score of -1 is given to the bad opinion words, and a score of 1 is given to the good opinion words. A weight of W_i is also given to each feature.

Consider, for instance, the following sentence:

> "the **size** is small but the **battery** is not good"

Given a weight of 5 to the battery and a weight of 2 to the size, the total score would be:

> 2*(1*1)+5*(-1*1)= 2-5= -3 → Result: Negative

Experiment Result

Experiment result is summarized in two categories:

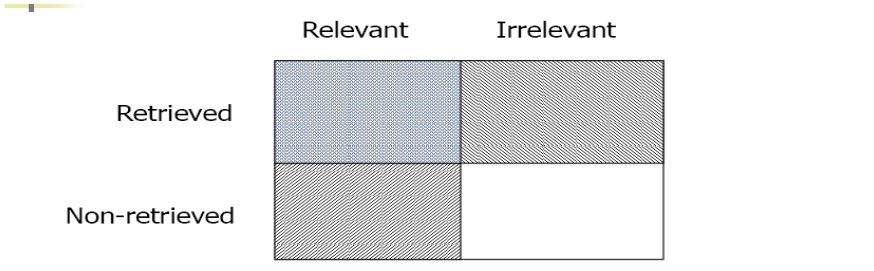
- Feature extraction
- Sentiment analysis

As explained already in Corpus section, there are two corpuses available: 1) Unlabeled Corpus from Reviews on Canon S100 and 2) Labeled Corpus from Reviews on Canon S100.

I performed my feature extraction algorithms on unlabeled corpus. Then, I wrote a program to extract features from Labeled corpus. Features on labeled corpus were done by authors manually. In some cases verbs and adjectives are considered as features. Also, features in labeled corpus do not stemmed. For instance, "picture" and "pictures" are considered as two separate features where as in my feature extraction algorithm, I perform stemming. I also do not consider verbs and adjectives as features. Figure 15 shows a screen shot of the features extracted by two different strategies. The red color shows the errors in feature extraction. The white color represents extra features extracted by the proposed algorithm comparing with the labeled features. A complete list of features is demonstrated in Appendix.

Labeled Features		My Features
battery		advantage
best built		amazon
buttons		August
camera		auto mode
cap		auto mode point
casing		baby
color balance		balance
color rendering		Battery
controls		battery charger
delay		battery life
durable		battery photography
expandability		battery pack
exposure		boating
features	NNs	button
fits	verb not as a feature	calculator watch
flash		camera
focus		cannon
housing		Canon
image quality		capture moment
images		censorship
light		Christmas
looks		clip max
mode		color rendering
memory		concern
memory expansion		consideration
metal case		control
optical zoom		course
performance		crap
performs		default resolution
picture		dinner
picture detail		download

Figure 15: Features extracted from unlabeled corpus (right side) and labeled corpus (left side)



$$\text{Precision} = \frac{\text{Relevant\&Retrieved}}{\text{Retrieved}} \qquad \text{Recall} = \frac{\text{Relevant\&Retrieved}}{\text{Relevant}}$$

Due to two different strategies to extract features described above, I am not able to calculate Precision and Recall and compare two algorithms. However, the following shows the result from the proposed feature extraction algorithms:

The proposed feature extraction algorithm:

	Proposed Feature Extraction	**Features extracted from labeled corpus**
Total number of features:	122	111
Error	15	Labeled manually
Relevant	80%	
Retrieved		
Extra features extracted (comparing with labeled features)	60	

Table 3: Features extracted by two different strategies

Retrieved= 68 out of 112 retrieved = 68

Regarding the sentiment analysis and finding interesting patterns, I used decision rule learning, general to specific search. To do this, I performed my algorithm on unlabeled corpus by running the program. As it classified the opinion, I compared them with the labeled corpus, If they matched, I removed that particular reviews from unlabeled corpus. Figure 16 shows some of these results :

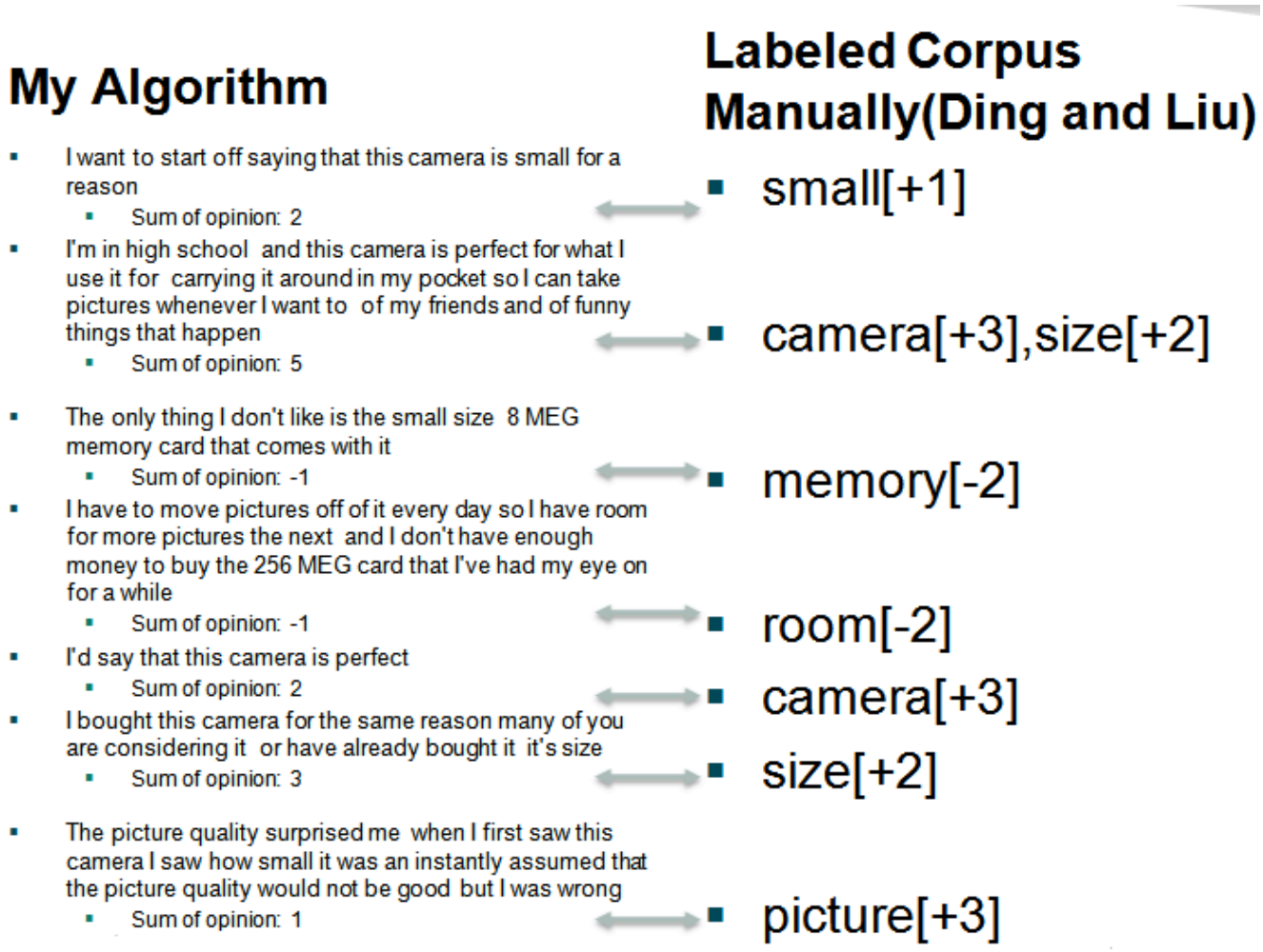


Figure 16: The proposed sentiment analysis algorithm (left side), Labeled corpus manually (right side)

Method Summaries generated by Java Doc:

Method Summary	
static void	**chunkingSentences**() Chunking sentences
static void	**convertListToMap**() convert a list to Hashmap structure
static java.util.Map	**deserializableFromFile**(java.lang.String filename) Perform deserialization of opinion words
static void	**featureExtraction**() Feature extraction method
static void	**findNamesInSentences**() Find names in a text
static boolean	**isBadVerb**(java.lang.String verb) Check if the word is negative
static void	**loadCommonWords**() Load Stop words
static void	**loadDoctoHashMap**(java.lang.String filename, java.util.Map map) Read a file and load it to Hashmap
static void	**loadFeatures**() Load Features from the file
static void	**main**(java.lang.String[] args)
static boolean	**matchFeature**(java.lang.String mFeature) Match features
static void	**posTags**() Part of Speech tagging
void	**preprocessText**(java.lang.String sentence) Preprocessing Task
static void	**printArray**(java.util.List list)

static void	**printArrayList**(java.util.List list, java.lang.String message) Print ArrayList
static void	**printHashMap**(java.util.Map m, int index) Print Hashmap
static void	**prunTree**(java.lang.String parsedSen) Pruns the tree
static java.lang.String	**removePunctuations**(java.lang.String sentence) Remove punctuation marks from documents
java.lang.String	**removeStopWords**(java.lang.String sentence) Remove stop words
static java.lang.String	**sentimentAnalysis**() Sentiment Analysis
static void	**sequenceAnalyzer**(java.lang.String word) Find synsets from WordNet
static void	**serializableToFile**(java.util.Map o, java.lang.String filename) Perform serialization of opinion words
static java.lang.String[]	**spansToStrings**(opennlp.tools.util.Span[] spans, java.lang.String s) Convert Span Object to a String
static java.lang.String	**stemming**(java.lang.String word) Perform stemming
static void	**writeToFile**(java.lang.String filename, java.util.List<java.lang.String> list) Write a list to a file
static void	**writeToFileHashMap**(java.lang.String filename, java.util.Map m) Write a hashmap structure to a file

Appendix:

Feature Extraction:

Labeled Features		My Features
battery		advantage
best built		amazon
buttons		August
camera		auto mode
cap		auto mode point
casing		baby
color balance		balance
color rendering		Battery
controls		battery charger
delay		battery life
durable		battery photography
expandability		battery pack
exposure		boating
features	NNs	button
fits	verb not as a feature	calculator watch
flash		camera
focus		cannon
housing		Canon
image quality		capture moment
images		censorship
light		Christmas
looks		clip max
mode		color rendering
memory		concern
memory expansion		consideration
metal case		control
optical zoom		course
performance		crap
performs		default resolution
picture		dinner
picture detail		download

picture quality		ease
pictures		Ebay
price		edge
prints		Elph S100
read		email
resolution		example
rugged		expandability
s100		exposure
sharp	Adj	Exposure compensation
shots		file
small		film
smaller		film equivalent
smallest		flash
software		Fool proof
tiny		front zipper
usability		function
use		grip finger
weighs		hate
zoom range		kayaking
auto color balance		king
auto focus		LCD
auto mode		LCD color
automatic modes		light
batteries		light compact
body		lithium
bundled software		lo behold
cameras		loh behold
canon s100		luck
capacity		macro
case		macro/infinity mode
color and light		Maine
compact		manual

covering		memory card
design		metal case -NOT PLASTIC.
download		mind camera
drivers		money
dynamic range		month
ease of use		mylife
elph s100		NYC
fits		oil
fun		panorama
functionality		person
hot		photo quality paper
jewel		PhotoStitch software
lcd		pics
lens		picture
macro		practice
macro mode		price
manual mode		pricy
manual modes		print
mode		print computer
movie mode		problem camera zoom function
on camera controls		quality
panorama mode		razor
panorama modes		rendering
panorama setting		resolution
photos		room thing battery
pics		sale
pictures of people		Santa
price		screen
processing		Second
product		size
purchase		snow background
resolution setting		story

room		sunlight
s100		Tamrac
screen		time
size		trouble
smal		Turkey
snapshots		usability
software		use
start up		vacation
tft		Viking
tft screen		Walgreen
use		Windows
viewfinder		work
wait		XP
work		Zoom
zoom		ZoomBrowser
zooms		auto mode
		Nova Scotia
		Photoshop
		room
		TFT.
		XP usb software
		best-built
		ease
		metal case
		picture detail
		software
		panorama mode

References:

[1] Turney, P.D. (2002), Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, Pennsylvania, 417-424

[2] Mullen T. and Cllier N. (2004), Sentiment Analysis using Support Vector Machines with Diverse Information Sources, *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP' 04), 412-418

[3] Wiebe, R. Bruce, and T. O'Hara. (1999). Development and use of a gold standard data set for subjectivity classifications. *Proceedings of 37th Annual Meeting of the Assoc. for Computational Linguistics* (ACL-99), 246–253

[4] Kim, S.-M. and E.H. Hovy. (2004). Determining the Sentiment of Opinions. *Proceedings of the COLING conference*. Geneva, Switzerland.

[5] Wilson, T. and Wiebe, J. and Hoffmann, P. (2005),Recognizing contextual polarity in phrase-level sentiment analysis,Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing,347-354

[6] Opinion Mining, Sentiment Analysis, and Opinion Spam Detection Review Datasets (9 products), http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html (Accessed: Dec 20, 2010)

[7] Treebank, http://en.wikipedia.org/wiki/Treebank (Accessed: Dec 18, 2010)

[8] Rilloff and Wiebe. (2003). Learning Extraction Patterns for Subjective Expressions, *Proceedings of the* Empirical Methods in Natural Language Processing (EMNLP 03).

[9] Yu and Hazivassiloglou . (2003). Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentence (EMNLP-03).

[10]Manning, C.D., and Schutze, H. (1999). Foundations of Statistical Natural Language Processing, MIT Press, Cambridge, MA.

Chapter 2: Story to game Project

Story 2 Game Translator

The objective of this project is to translate a story written in natural language to a mobile game.

Here is an example of the story:

```
Once upon a time there was a king and queen who lived in a golden castle
with their beautiful daughter. One night an ugly ogre captured the
beautiful princess and locked the princess in his tall, dark tower. The
king and queen were very sad. They promised to give a bag of gold  to the
knight that rescued the princess. All the knights in the land wanted to
rescue the princess. They rode to the ogre's tower.

The ogre was so scary. They rode away as fast as they could. The next day
a friendly dragon was flying over the ogre's tower when he heard the
princess cry for help. The dragon blew the ogre into the ocean. The
dragon put the princess on his back and flew into the sky.

In the end, they flow over the tower and the castle, over the mountains
and caves. The princess was so happy to be free she kissed the dragon.
All at once he turned into a handsome prince and they lived happily ever
after.
```

Figure 1: A smaple story

The same algorithm proposed for opinion mining described in Chapter 1 can be used in Story to game project.

Here we have three abstract layers:

- Natural Language
- Resources like images, characters pictures
- Game API (using J2ME)

J2ME standing for Java micro edition is a Java language for mobile devices. The java-enabled devices like Nokia, Blackberry, Sony Ericsson, and Motorola support J2ME.

In natural language layer, we have text. Each word in the text is considered as an attribute. In contrast, in Game API using J2ME we have only a restricted number of objects with a variety of methods and arguments. The primary task in this project is to map words to J2ME objects (See Figure 2). However, due to the limited amount of time, I only implement the first layer.

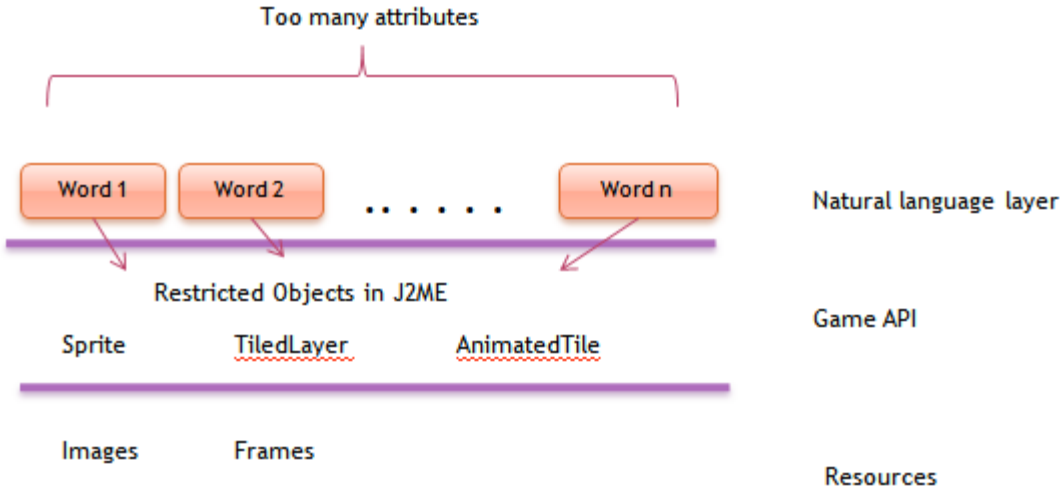


Figure 2: Abstract layers in Story2Game project.

Natural Language Layer

Given two many words in this layer the primary task is to obtain key features. What are key features? Key features are words that cam map to the Game API. They are represented as a tuple of <subject, verb, object>. To achieve this approach, insignificant words must be pruned from the sentences. In order to prune insignificant words , we can use the same algorithm proposed in Chapter 1. Diagram 1 shows the steps implemented to prune insignificant words:

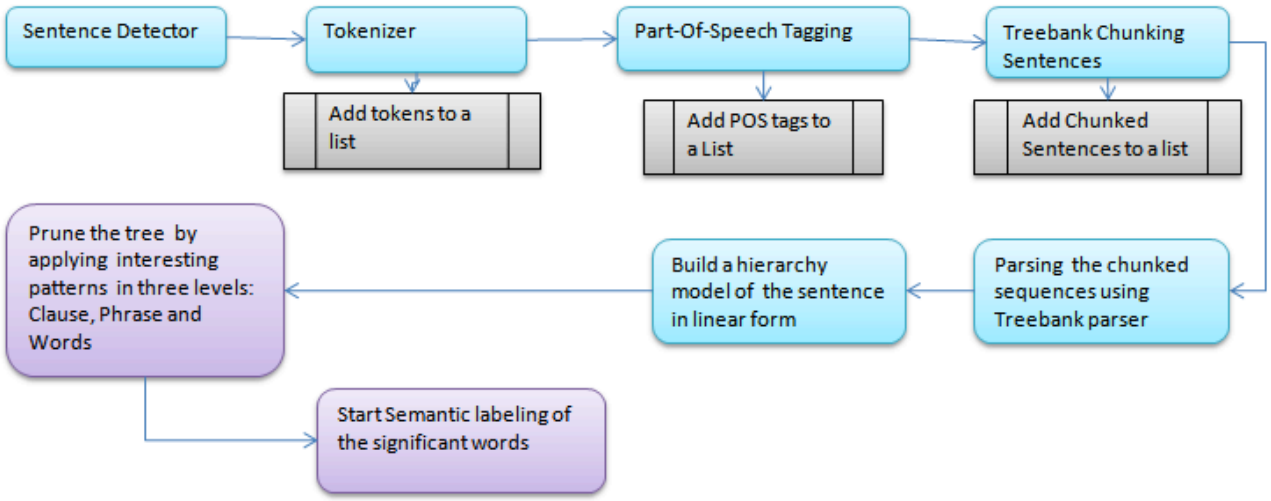


Diagram 1: Steps implemented to prune the insignificant words

The description about each module has been presented in Chapter 1.

The only part that is new in Story2Game project and not described in Chapter 1 is “semantic labeling of the significant words”. By performing this module, indeed, we are able to map the significant words to the game API objects. I will discuss about Semantic Labeling Process later in the next section

Semantic Labeling Process

Since we have the pruned tree, we can perform semantic labeling process. We have a limited number of labels here. Indeed, labels are defined in the domain of a game. Table 1 shows some of these labels.

Semantic Labels	Words mapped to the semantic labels
Action	Move, walk, fast, capture,lock
Instrument	Toy, gun,
Agent	Nouns
Collision	hit
Emotion	Sad, happy

Table 1: Words mapped to the semantic labels (initial seed)

Words in Table 1 are considered as initial seed. We can obtain more words for each label by getting the synonyms from WordNet dictionary.

After mapping words in pruned tree to semantic labels, we can build a conceptual graph. Semantic labels specify conceptual relationship in conceptual graph. Given the conceptual relations, we are able to perform a Java script code.

Here is a rough idea about semantic labeling. The process needs to be trained for a specific story.

Experiment

Given the following story, we are only interested in mining the following words described as Bold in Figure 3.

Once upon a time there was a **king** and **queen** who **lived** in a **golden castle** with their beautiful **daughter**. One **night** an ugly **ogre captured** the beautiful **princess** and **locked** the princess in his tall, dark **tower**. The **king** and **queen** were very **sad**.

Figure 3: Significant Words

By performing the proposed algorithm on sentences, we can prune the parse tree. Figure 4 demonstrates the pruned tree generated by the program.

```
_____________________________________________________
Sentences:....Once upon a time there was a king and queen who

Tagged Tokens:
RB, IN, DT, NN, EX, VBD, DT, NN, CC, NN, WP,
 List Size:11

Chunked sentences:
B-SBAR, I-SBAR, B-NP, I-NP, B-NP, B-VP, B-NP, I-NP, I-NP, I-NP, O,
 List Size:11

Features...
gold knight, help, ,
 List Size:3
TreeBank1....(TOP (S (ADVP (RB Once)) (PP (IN upon) (NP (DT a) (NN time))) (NP (EX there)) (VP (VBD was) (NP (DT a) (NN king) (CC and) (NN queen) (WP
TreeBank2....(TOP (S (ADVP (RB Once)) (PP (IN upon) (NP (DT a) (NN time))) (NP (RB there)) (VP (VBD was) (NP (DT a) (NN king) (CC and) (NN queen) (WP

Stack:
(, (, S, (, ADVP, (, ), ), (, (, ), (, NP, (, ), (, NN, time, ), ), ), (, NP, (, ), ), (, VP, (, VBD, was, ), (, NP, (, ), (, NN, king, ), (, ), (, NN
 List Size:52

_____________________________________________________
Sentences:....lived in a golden castle with their beautiful daughter

Tagged Tokens:
VBN, IN, DT, JJ, NN, IN, PRP$, JJ, NN,
 List Size:9

Chunked sentences:
B-VP, B-PP, B-NP, I-NP, I-NP, B-PP, B-NP, I-NP, I-NP,
 List Size:9

Features...
gold knight, help, ,
 List Size:3
TreeBank1....(TOP (VP (VBN lived) (PP (IN in) (NP (DT a) (JJ golden) (NN castle))) (PP (IN with) (NP (PRP$ their) (JJ beautiful) (NN daughter)))))
TreeBank2....(TOP (VP (VBN lived) (PP (IN in) (NP (NP (DT a) (JJ golden) (NN castle)) (PP (IN with) (NP (PRP$ their) (JJ beautiful) (NN daughter))))))

Stack:
(, (, VP, (, VBN, lived, ), (, (, ), (, NP, (, ), (, JJ, golden, ), (, NN, castle, ), ), ), (, (, ), (, NP, (, ), (, JJ, beautiful, ), (, NN, daughter
 List Size:43

_____________________________________________________
Sentences:....ugly ogre captured the beautiful princess and locked the princess in his tall  dark tower

Tagged Tokens:
JJ, NN, VBD, DT, JJ, NN, CC, VBD, DT, NN, IN, PRP$, JJ, JJ, NN,
 List Size:15

Chunked sentences:
B-NP, I-NP, B-VP, B-NP, I-NP, I-NP, O, B-VP, B-NP, I-NP, B-PP, B-NP, I-NP, I-NP, I-NP,
 List Size:15

Features...
gold knight, help, ,
 List Size:3
TreeBank1....(TOP (S (NP (JJ ugly) (NN ogre)) (VP (VP (VBD captured) (NP (DT the) (JJ beautiful) (NN princess))) (CC and) (VP (VBD locked) (NP (DT the
TreeBank2....(TOP (S (NP (JJ ugly) (NN ogre)) (VP (VP (VBN captured) (NP (DT the) (JJ beautiful) (NN princess))) (CC and) (VP (VBD locked) (NP (DT the

Stack:
(, (, S, (, NP, (, JJ, ugly, ), (, NN, ogre, ), ), (, VP, (, VP, (, VBD, captured, ), (, NP, (, ), (, JJ, beautiful, ), (, NN, princess, ), ), ), (, )
 List Size:78
```

```
Sentences:....The king

Tagged Tokens:
DT, NN,
 List Size:2

Chunked sentences:
B-NP, I-NP,
 List Size:2

Features...
gold knight, help, ,
 List Size:3
TreeBank1....(TOP (NP (DT The) (NN king)))
TreeBank2....(TOP (S (NP (DT The)) (CONJP (NN king))))

Stack:
(, (, NP, (, ), (, NN, king, ), ), ),
 List Size:11
```

```
Sentences:....and queen were very sad

Tagged Tokens:
CC, NN, VBD, RB, JJ,
 List Size:5

Chunked sentences:
O, B-NP, B-VP, B-ADJP, I-ADJP,
 List Size:5

Features...
gold knight, help, ,
 List Size:3
TreeBank1....(TOP (S (CC and) (NP (NN queen)) (VP (VBD were) (ADJP (RB very) (JJ sad)))))
TreeBank2....(TOP (S (NP (DT and) (NN queen)) (VP (VBD were) (ADJP (RB very) (JJ sad)))))

Stack:
(, (, S, (, ), (, NP, (, NN, queen, ), ), (, VP, (, VBD, were, ), (, ADJP, (, ), (, JJ, sad, ), ), ), ), ),
 List Size:30
```

Figure 4: Pruned tree generated from story sentences