CSE 498 Project2 Report

Aspect Based Sentiment Analysis

Team members:
Zhaohan Xi, Zexu Wang, Chenghao Ma

I. Background and Goals of Project

Sentiment analysis is increasingly viewed as a vital task both from an academic and a commercial standpoint. The majority of current approaches, however, attempt to detect the overall polarity of a sentence, paragraph, or text span, regardless of the entities mentioned (e.g., laptops, restaurants) and their aspects (e.g., battery, screen; food, service).
By contrast, this task is concerned with aspect based sentiment analysis (ABSA), where the goal is to identify the aspects of given target entities and the sentiment expressed towards each aspect. Datasets consisting of customer reviews with human-authored annotations identifying the mentioned aspects of the target entities and the sentiment polarity of each aspect will be provided.

II. Description of Subtasks

<II-1>Subtask 1: Aspect term extraction
Given a set of sentences with pre-identified entities (e.g., restaurants), identify the aspect terms present in the sentence and return a list containing all the distinct aspect terms. An aspect term names a particular aspect of the target entity.
For example, "I liked the *service* and the *staff*, but not the *food*", "The *food* was nothing much, but I loved the *staff*". Multi-word aspect terms (e.g., "hard disk") should be treated as single terms (e.g., in "The *hard disk* is very noisy" the only aspect term is "hard disk").

<II-2>Subtask 2: Aspect term polarity
For a given set of aspect terms within a sentence, determine whether the polarity of each aspect term is positive, negative, neutral or conflict (i.e., both positive and negative).
For example:
"I loved their **fajitas**" → {fajitas: *positive*}
"I hated their **fajitas**, but their **salads** were great" → {fajitas: *negative*, salads: *positive*}
"The **fajitas** are their first plate" → {fajitas: *neutral*}
"The **fajitas** were great to taste, but not to see" → {fajitas: *conflict*}

<II-3>Subtask 3: Aspect category detection
Given a predefined set of aspect categories (e.g., price, food), identify the aspect categories discussed in a given sentence. Aspect categories are typically coarser than

the aspect terms of Subtask 1, and they do not necessarily occur as terms in the given sentence.
For example, given the set of aspect categories **{food, service, price, ambience, anecdotes/miscellaneous}**:
“The restaurant was too expensive” → **{price}**
“The restaurant was expensive, but the menu was great” → **{price, food}**

<II-4>Subtask 4: Aspect category polarity
Given a set of pre-identified aspect categories (e.g., **{food, price}**), determine the polarity (*positive*, *negative*, *neutral* or *conflict*) of each aspect category.
For example:
“The restaurant was too expensive” → **{price: negative}**
“The restaurant was expensive, but the menu was great” → **{price: negative, food: positive}**

III. Description of Dataset

***<III-1>Restaurant reviews*:**
This dataset consists of over 3K English sentences from the restaurant reviews of Ganu et al. (2009). The original dataset of Ganu et al. included annotations for coarse aspect categories (Subtask 3) and overall sentence polarities; we modified the dataset to include annotations for aspect terms occurring in the sentences (Subtask 1), aspect term polarities (Subtask 2), and aspect category polarities (Subtask 4). Additional restaurant reviews, not in the original dataset of Ganu et al. (2009), are being annotated in the same manner, and they will be used as test data.

***<III-2>Laptop reviews*:**
This dataset consists of over 3K English sentences extracted from customer reviews of laptops. Experienced human annotators tagged the aspect terms of the sentences (Subtask 1) and their polarities (Subtask 2). This dataset will be used only for Subtasks 1 and 2. Part of this dataset will be reserved as test data.

<III-3>Dataset format:
The sentences in the datasets are annotated using XML tags.
The following example illustrates the format of the annotated sentences of dataset.

```
<sentence id="813">
        <text>All the appetizers and salads were fabulous, the steak was mouth
watering and the pasta was delicious!!!</text>
        <aspectTerms>
                <aspectTerm term="appetizers" polarity="positive" from="8" to="18"/>
                <aspectTerm term="salads" polarity="positive" from="23" to="29"/>
                <aspectTerm term="steak" polarity="positive" from="49" to="54"/>
                <aspectTerm term="pasta" polarity="positive" from="82" to="87"/>
        </aspectTerms>
        <aspectCategories>
                <aspectCategory category="food" polarity="positive"/>
        </aspectCategories>
</sentence>
```

The possible values of the polarity field are: “positive”, “negative”, “conflict”, “neutral”. The possible values of the category field are: “food”, “service”, “price”, “ambience”, “anecdotes/miscellaneous”.
The format of the annotated sentences of the laptops dataset are same as the restaurant datasets, with the only exception that there are no annotations for aspect categories.

<III-4>Useful Link:
On the following link we can find all dataset in this project:
http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools

IV. How We Use the Dataset

<IV-1> Based on train, test and trial
<a> For the training data:
We use them to “train” some useful tools designed by ourselves. Both restaurant and laptop reviews are useful.
<b> For the testing data:
We use them to get outcomes of this project. Both restaurant and laptop reviews are useful.
<c> For the trial data:

We use to estimate our methods and evaluate its accuracy. But only restaurant reviews are useful because the laptop reviews are incomplete.

<IV-2> Based on type of dataset
<a>Restaurant data
Using for all subtasks, to extract aspect terms, extract aspect categories, then determine their corresponding polarities,
<b>Laptop data
Using for only subtask1 and 2, since this type of dataset does not have aspect categories, then subtask3 and 4 is meaningless in this kind of dataset.

<IV-3> Based on given original files
<a> In training dataset, their are totally 2 version of data, we use the second version of training data since the second version have modified some little error of the first version.
<b> In testing dataset, there are two phases of data for each type of dataset, we use both of them since phase A are raw texts, which we can use in subtask1 and 3, and phase B are text with given aspect terms and categories, which we can use in subtask2 and 4 directly and do not need to use outcomes of subtask1 and 3 to analyze polarity.

V. Methods Instruction

In this project, even we could used some sophisticated model or tools to analyze the sentence, which we could get the final outcomes directly and quickly. That sophisticated models and tools are also time-saving that we could finish this whole project in a single day with high accuracy outcomes. But this is not interesting and we could nearly learn nothing from this project. Also, this is not what professor want us to do.
Thus our team give up such sophisticated model or tools, we only use some basic methods such as POS tagger and Dependency Parser which could help us to do some useful analysis of a sentence. All methods and algorithms in our project are built by ourselves. To do this, we read some useful paper such as Hu and Liu (2004) and extract our own way from these paper.
Here I introduce our methods in details. Subtask 1 and 3 are put together and subtask 2 and 4 are put together since their methods are uniformed.

<V-1> Subtask1 and 3:

<1-1> Initial Method of subtask1:

Our team aimed to extracted the aspected term from a sentence. Normally, the aspect term is noun or phrase centered as a noun (with some adjectives to describe the noun). Thus the first step is extracted all noun. Then we should judge which nouns or noun-relevant phrases can be seen as aspect term. This idea guides us use training data in a suitable way.

Normally, the training data are used to train a model, then we use the testing data to test our outcomes. But in this project we do not have a specific model, but we could also use the training data to do some “train” work.

Aspect Term Dictionary:

Since all the aspect terms are give in the training data, what we do is build a big dictionary based on theses aspect terms, in this report we call it as ***Aspect Term Dictionary (AT Dict).*** What we think is use this *AT Dict* to detect whether a noun or noun phrase in testing sentence in it, if yes, then it is high probability that this noun or phrase is an aspected term since all training data an testing data are produce from same dataset, then sentence and words use in both training data and testing data have high similarity.

But this method not get good outcomes: we have 1200 words and phrase in our *AT Dict* but only 200 words are used in testing sentence, there are so many sentences in testing data which should have aspect terms but cannot be extracted even one. Thus we modified our methods by enlarge our *AT Dict* in a suitable way.

<1-2> Enhance method of subtask1:

We remain our *AT Dict* but find a way to enlarge it, which could highly enhance the accuracy to detect the aspect terms from testing data. We use the frequency based method. For each sentence in the testing data, we extract all nouns and noun phrases, then we sort them by its frequency.

Our team manually set a threshold (in our project is 2, since the sentences in testing data are such limited and a word has low probability to be used twice), if the frequency of a noun or phrase is not less than threshold, then we judge it as an aspected term and put it in *AT Dict* .

We notice that many noun phrases are aspect term but only be used once in all testing sentences, since these phrases and have many ways to combine more than one words together, thus it is in low probability a same phrase occur more than twice among sentences. But if such a phrase contain nouns in our *AT Dict*, we believe there is high probability that it is an aspect term and put it into our *AT Dict*. In this way we could

enlarge our dictionary effectively. Then for each sentence in testing data, we could judge whether this sentence contains nouns or phrases in our *AT Dict* and detect its aspect terms.
There is also a further enhance combine with the method of subtask3, we will touch it later.

<1-3> Method of subtask3:
In this subtask, what we want is find a correspondence between aspect term and aspect category. Because training data and testing data are uniformed, thus we believe such kind of correspondence are also uniformed.

Aspect Term and Category Correspondence Dictionary:
In this subtask, we build another dictionary which contains such kind of correspondence between each aspect term and aspect category, we call it ***Aspect Term and Category Correspondence Dictionary (ATCC Dict)***. In subtask1 we have an Aspect Term Dictionary, we could easily get all aspect categories from training data and get the correspondence: which is, for each aspect term in our dictionary, we could find exactly one aspect category based on training sentence. There are only 5 aspect categories: “food”, “service”, “price”, “ambience” and “anecdote/miscellaneous”, each aspect term must belong to one of them.
After we get *ATCC Dict*, we could determine aspect category after extracting aspect term of each sentence, this method has high accuracy with sentence has aspect term. But for sentences do not have one aspect term, we could not use this method.

<1-4> Complement method of subtask3 with no aspect term situation:
This part of method is not an enhance but a complement of method above, using to deal with the situation that a testing sentence dose not have aspect term, in this time we could not use *ATCC Dict* but only based on semantics.

Adjective correspondence based method:
We notice that no matter in training data or in testing data, each sentence exactly has at least one aspect category but not all sentences have aspect term. Thus we build some correspondence between aspect category and sentence element itself.
We also use the training data to do some “train” issues. For each training sentence, we use POS tagger to get all adjectives, since this sentence has definite aspect category, we could believe that adjectives in this sentence are used to describe its category. Thus, after analyze all training sentences, we could get 5 long lists, each of them contains adjectives for specific category.

Then we tag each testing sentence and get its adjectives, we find all category lists containing these adjectives, we could believe these category are also aspect category for this sentence. For sentence without any adjective in these category lists, we judge it as the “anecdote/miscellaneous” category.
To enhance the accuracy of this method, we use WordNet to find synonyms of each adjective in each list, because some reviews may not use the original adjective but uses its synonyms.

<1-5> Enhance again of method in subtask1:
In method above we use the correspondence of adjective and category, we could based this idea to get further enhance of our frequency based method in subtask1.

Adjective correspondence based method:
In subtask1, if a noun or noun phrase has frequency lower than our threshold and it dose not contain a word in *AT Dict*, then we judge it as not the aspect term. In order to decrease the probability of misjudge situation, we use POS tagger to find adjectives in a testing sentence and detect whether they are in one of the category lists, if yes, we believe this adjective are use to describe an aspect term, which also corresponds to an aspect category. Then we use Dependency Parser to find such a noun and believe it as aspect term of this sentence.

<V-2>   Subtask2 and 4:

<2-1> Method of subtask2:
Subtask2 and 4 only need to analyze polarity of aspect terms or categories, in the relevant dataset the aspects are given, thus we could use them to do our task and do not need to based on outcomes of subtask1 and 3.

Aspect adjective method:
In this project, the aspect terms are all noun or noun phrase, which is the topic of a sentence. Here we come out a new concept in our project, which uses adjectives in a sentence as key word to distinguish different polarity, this idea make sense because, in a sentence, there is always adjective to show the sentimental polarity of speaker, thus adjective should be key in sentiment analysis.

Notice the dataset has an interesting property: **No matter in train, test or trial data, as long as they are in same type, restaurant or laptop, the semantics will always same.** Which means, for example, we could use adjectives in training data to distinguish the adjectives in testing data in which kind of polarity.

Here we also use Stanford Dependency Parser to get all noun~adjective pairs in training data, in which the noun is the given aspect term. Based on polarity of each aspect term, we could put their correspond adjectives into different lists. There are total four lists, for four polarities: “positive”, “negative”, “neutral” and “conflict” respectively.
After get four lists containing aspect adjectives, we use WordNet to enlarge our lists since many adjectives used in testing data may not be used in training data. This step could also enhance the accuracy of our project.
Then in testing data, since the aspect terms are given, for each text we also use Dependency Parser to extract all noun and adjective pairs, in which the nouns are given aspect terms. Then we could simply detect adjective in which polarity list to determine the polarity of correspond aspect terms.

<2-2> Methods of subtask4:
Just like subtask3, which only for restaurant dataset since only restaurant texts have aspect category.

Ensemble method:
Here we use an ensemble method to solve this subtask, notice we called it ensemble because it combine nearly all methods we have used in the former subtasks, considering the situation of subtask4 is quite complex. Here we come out another assumption in text, by which we could deal with this subtask in a reasonable way.

Polarity uniform assumption:
After research the dataset, our team get another important assumption in this project : **In each review text, the sentiment is always uniform in whole sentence or in part of it.** We find most sentences in data has uniform polarity, no matter how many aspect terms or how many aspect categories they have. For sentences do not have uniform polarity, there always exist some adversatives such “although”, “but”, “however” in sentences, polarity on each side of these adversatives is also uniform. Under this assumption, some analysis could be quite explicit.

There are three situation in this subtask: (a) sentence with aspect term and have uniform polarity; (b) sentence with aspect term but does not have uniform polarity; (c) sentence without aspect term.

(a) For the first situation:
We believe the polarity of whole text is uniform, which means, no matter how many aspect terms or how many aspect categories it has, their polarities are same. Thus we could determine each aspect category polarity by finding the uniform polarity of aspect terms, which is outcome of subtask2.
(b) For the second situation:
We believe the polarity is part uniform in a text, thus we should firstly find the correspond aspect category of each aspect term, using the *ATCC Dict* in subtask3, we could quickly find relevant aspect category, then its polarity is the polarity of correspond aspect term. In this situation, we use the *Correspondence Method* in subtask3, this method make sense because there is a tight connect of an aspect term and category, thus their polarity must be same.
(c) For the last situation:
We could not use in *Correspondence Method* subtask3, thus a more advanced method should be use, which we could put a sentence into TextBlob and analyze its polarity. Before doing this, it would be better that we use the four *Aspect Adjective Lists* in subtask2 and find whether there are some adjectives in this text and determine whether the polarity of this text is uniform, if not, we should break the text by its adversatives and put each part into TextBlob.

VI. Pseudocode and Flow Chart

In this part, we put complete algorithms in our project by pseudocode and flow chart, in which we put subtask1 and 3 together, 2 and 4 together, since they are done in a complete section in our program.
Note we do not distinguish which is dealing the restaurant data and laptop data, since laptop data does not have subtask3 and 4, we just simply put our whole algorithms here, and laptop data only use part of this algorithm.

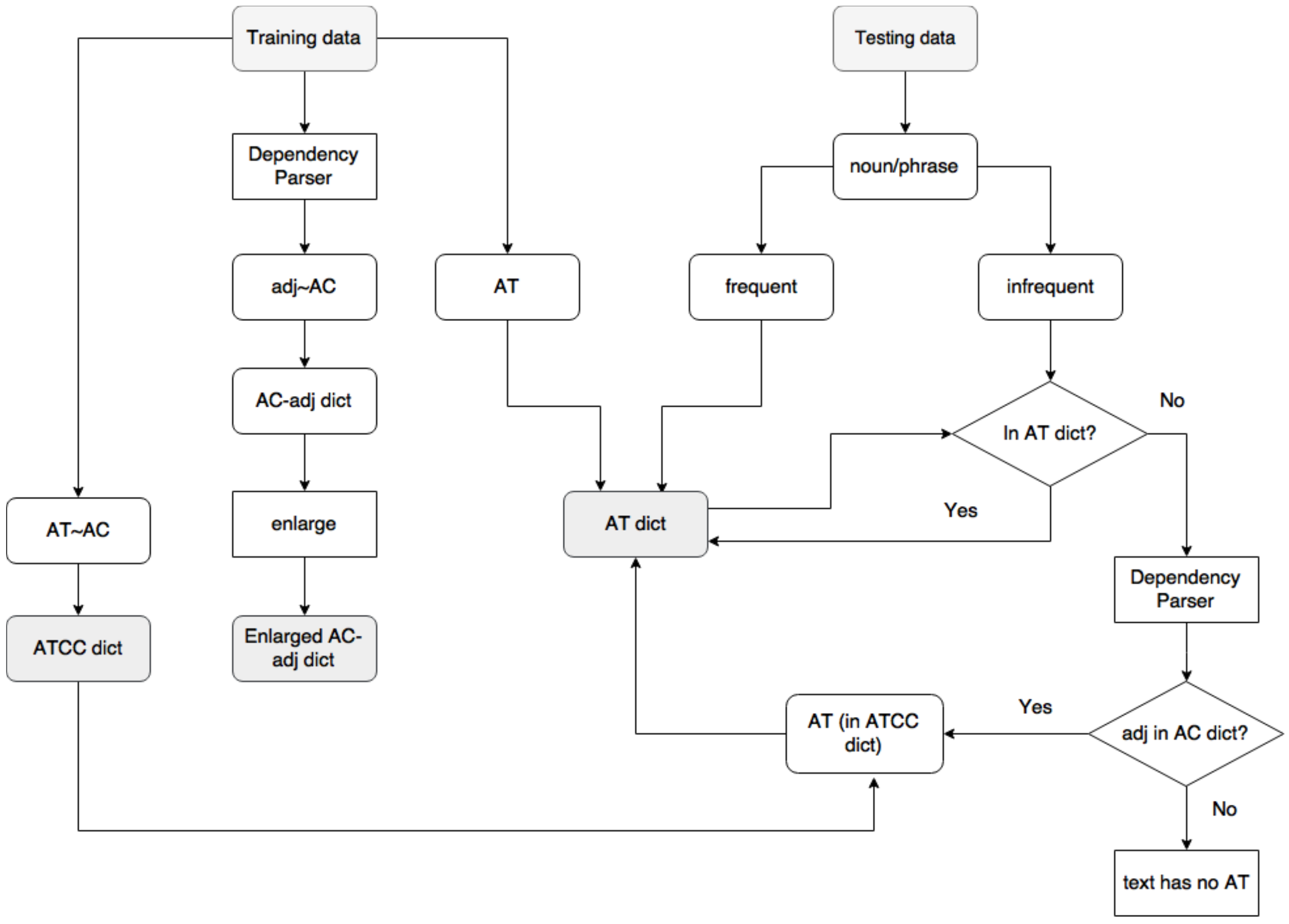
<VI-1: Pseudocode of subtask 1 and 3>:

```
1 Build AT dictionary(training data, testing data):
2         for each text in training data:
3                 get their AT and put into AT dictionary
4         for each text in testing data:
5                 extract nouns and noun phrases
6                 set threshold and count frequent nouns or phrases
7                 put frequent candidates into AT dictionary
```

```
        for each infrequent candidates:
            judge whether containing words in AT dictionary
            if yes, put it into AT dictionary
    return a long list with AT dictionary and infrequent candidates

Build AC-adj dictionary(training data):
    for each text in training data:
        extract adj by Stanford Dependency Parser
        put adj into corresponding AC dictionary
    enlarge AC-adj dictionaries by WordNet
    return AC-adj dictionaries

Build ATCC dictionary(training data):
    for each text in training data
        find correspondence by AT and AC
    return ATTC dictionary

Enhance AT dictionary(AT dict, infrequent candidate list, AC-adj dict, ATCC dict):
    for each word in infrequent candidate list:
        find whether in AC-adj dict
        if yes, find correspond AT in ATCC dict
        put this AT into AT dict
    return AT dict

Task1(testing data, AT dict):
    for each text in testing data:
        extract all nouns and noun phrases
        if some are in AT dict, judge them as AT of this text
        if no one in, believe this text does not has AT
    return outcome of task1

Task3(outcome of task1, AC-adj dict, ATCC dict):
    for each text in outcome of task1:
        if it has AT:
            for each AT:
                find AC in ATCC dict
            if there are AC find, judge it as AC of this text
            if no AC find:
                extract adj by Stanford Dependency Parser
                find adj in which AC-adj dict, judge as AC of this text
        if text does not have AT:
            extract adj by Stanford Dependency Parser
            find adj in which AC-adj dict, judge as AC of this text
    return outcome as final outcome of subtask 1 and 3
```

<VI-2: Flow chart of subtask 1 and 3>:

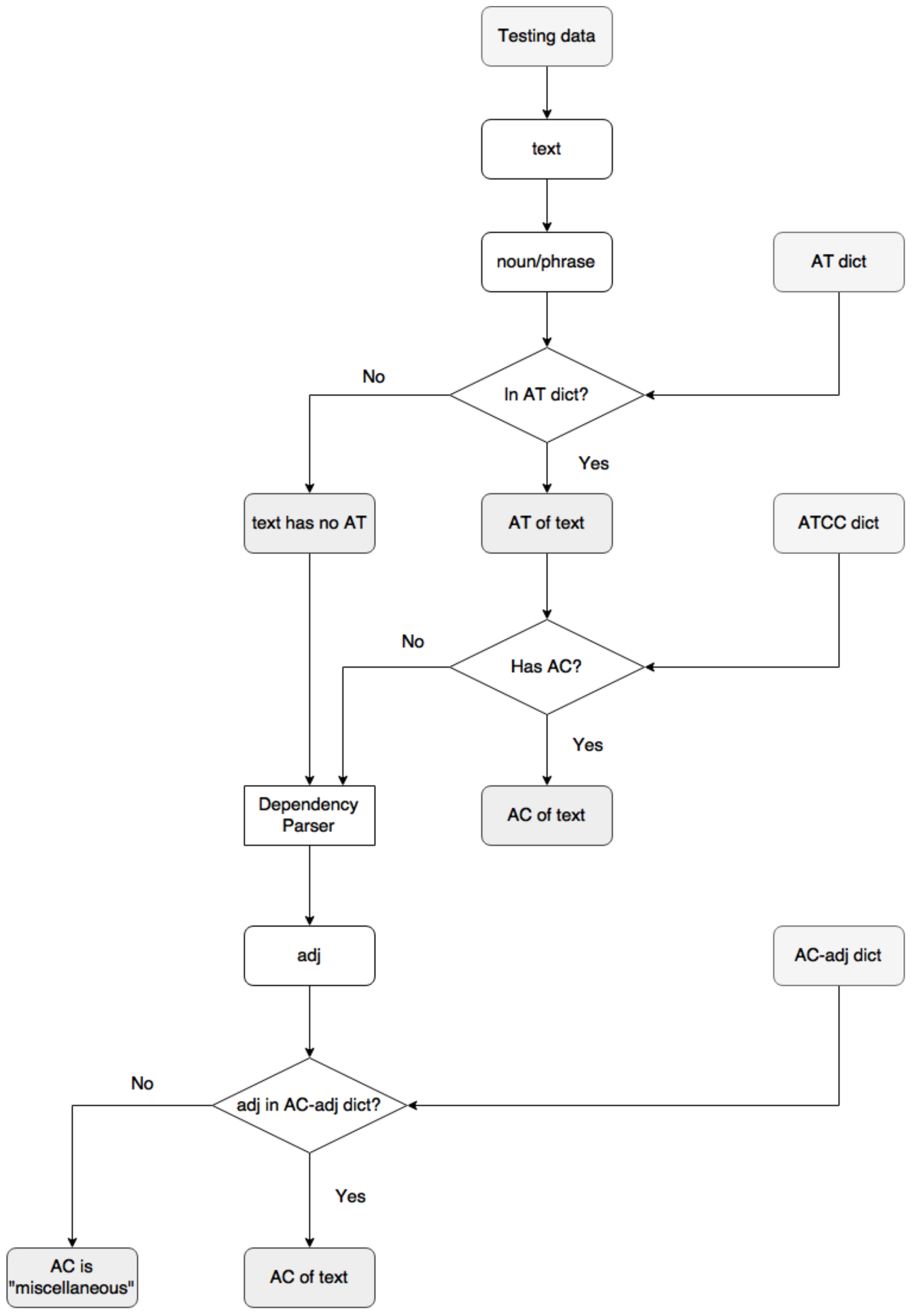
<get our tools>



In this step, we use:
— training data
— testing data

After this step, we get:
— AT dict
— ATCC dict
— Enlarged AC-adj dict

<processing testing data>

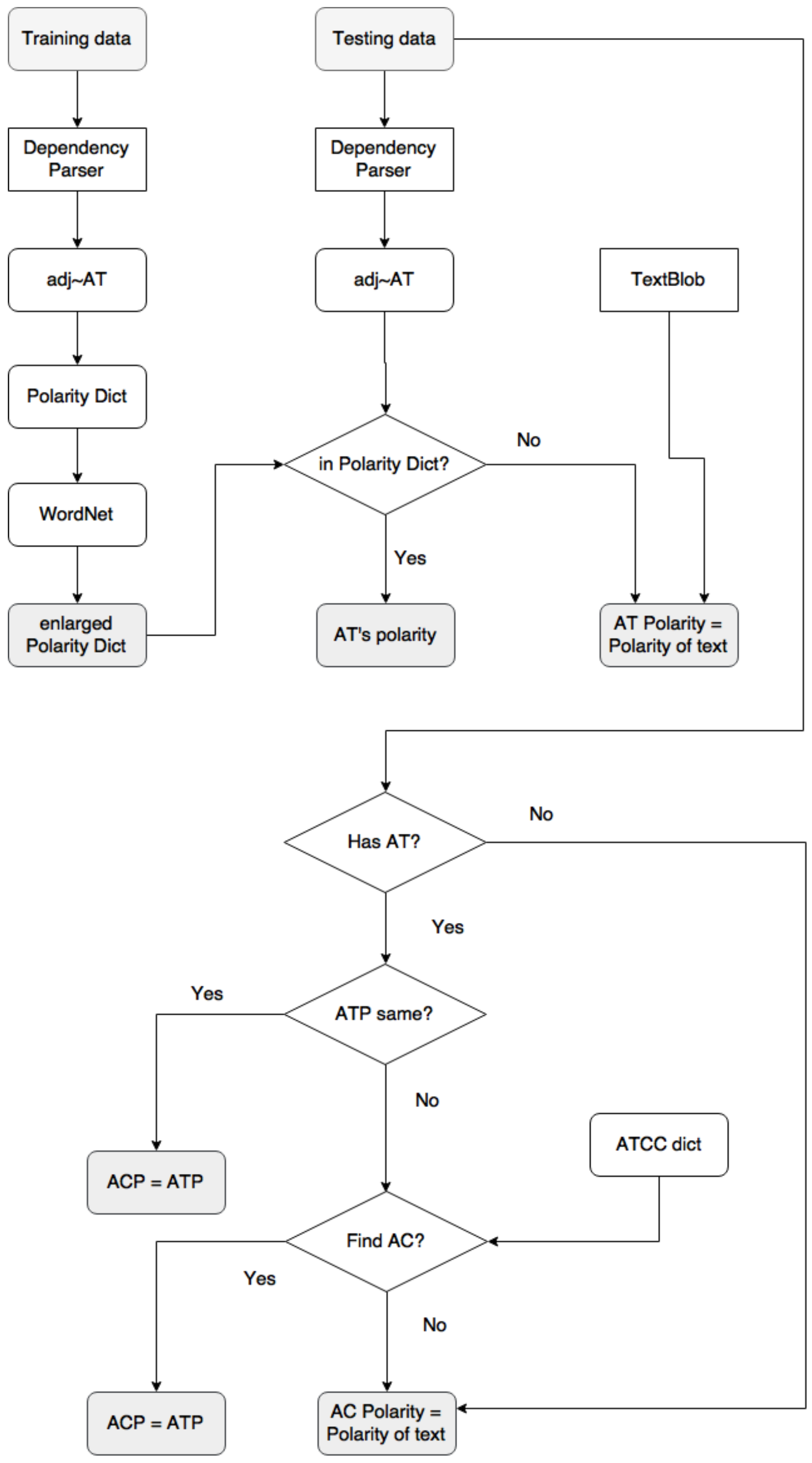


Testing data
text
noun/phrase
AT dict
In AT dict?
No
Yes
text has no AT
AT of text
ATCC dict
Has AC?
No
Yes
Dependency Parser
AC of text
adj
AC-adj dict
adj in AC-adj dict?
No
Yes
AC is "miscellaneous"
AC of text

<VI-3: Pseudocode of subtask 2 and 4>:

```
Build Polarity Dictionary(training data):
    for each text in training data:
        use Stanford Dependency Parser find adj for each AT
        put these adj into Polarity Dict based on polarity of AT
    for each Polarity Dicts:
        use WordNet to enlarge Polarity Dict
    return Polarity Dicts

Task2(testing data, Polarity Dicts):
    for each text in testing data:
        find adj for each AT by Stanford Dependency Parser
        determine adj in which Polarity Dicts
        if in some Dicts:
            determine the polarity as AT polarity
        if adj not in any Dicts:
            analyze the text by TextBlob and get an polarity of text
            determine each AT polarity as polarity of text
    return outcomes of task2

Task4(testing data, Polarity Dicts, ATCC Dict, outcomes of task2):
    for each text in testing data:
        if text has AT:
            if all AT polarities are same:
                AC polarity = AT polarity
            if all AT polarities are not same:
                for each AC:
                    find correspond AT in ATCC Dict
                    if find:
                        AC polarity = AT polarity
                    if not find:
                        use TextBlob analyze text and get polarity
                        AC polarity = text polarity
        if text not have AT:
            use TextBlob analyze text and get polarity
            for each AC in text:
                AC polarity = text polarity

    return outcomes of task4
    //for restaurant data, this is the final outcome of task2 and 4
    //for laptop data, outcome of task2 is the final outcome
```

<VI-4: Flow chart of subtask 2 and 4>:
<get our tools> & <processing testing data>

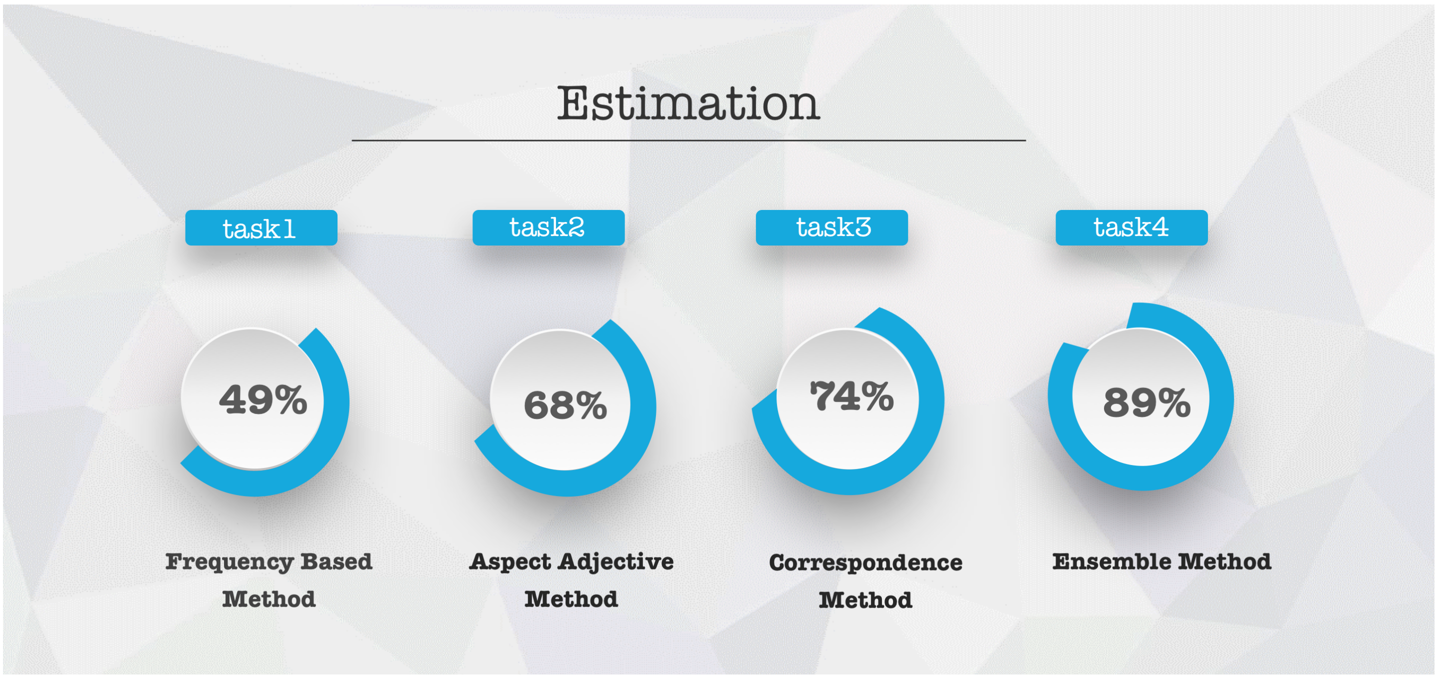

Training data
Dependency Parser
adj~AT
Polarity Dict
WordNet
enlarged Polarity Dict
Testing data
Dependency Parser
adj~AT
in Polarity Dict?
No
Yes
AT's polarity
TextBlob
AT Polarity = Polarity of text
Has AT?
No
Yes
ATP same?
Yes
No
ACP = ATP
ATCC dict
Find AC?
Yes
No
ACP = ATP
AC Polarity = Polarity of text

VII. Estimation

In this part, we only use restaurant trial data since only this type of trial data is complete, that is, some texts of laptop trial data are not put any aspect term and its polarity in it, thus we could not use this texts to estimate our data.

<VII-1: Estimation outcome>

Here is outcome of estimation (screen from our presentation PPT):



Estimation

task1	task2	task3	task4
49%	68%	74%	89%
Frequency Based Method	Aspect Adjective Method	Correspondence Method	Ensemble Method

Accuracy is shown in each subtask.

<VII-2: Estimation outcome analysis>

No matter which part of estimation get an satisfying outcome, because our team has give correct assumption and suitable methods in this project.

The accuracy of subtask1 is relevant low, this may because the scale of trial data is limited, there is only 100 texts, thus using frequency based method cannot always get right aspect term, which means hot topic in this method, we mainly depend aspect terms in training data. But in real testing data, things would be better since the scale of texts is larger enough, thus frequency based method works better.

Another methods mainly rely on our several assumption about dataset and semantics of texts, thus outcomes could match real situation logically. Each method is not only rely on a single tools or idea, as you can see, there are many ideas combined into a complete algorithms, which includes all possible situation of this project.

VIII. Submitted Files Instruction

Instead of put this part of information into read me file, I will introduce all files we submitted and what they are used to in this report, in this part.
Here is an overlook of all folders:

codes data temporary files outcomes

<VIII-1> Folder "codes"

This folder contain all source codes in this program, all codes are in ".ipynb" form, which need use Jupyter Notebook to open and run, here we introduce all program files by its work sequence in this project.
—"XML to CSV.ipynb": This program contains many sub-program which use to read the original XML files and turn it into CSV files, outcomes are put into folder "temporary files".
—"task1 & 3 RST.ipynb" and "task1 LPT.ipynb": These two programs contain codes dealing with task1 and 3 for restaurant (RST) and laptop (LPT) respectively.
—"task 2 & 4 RST.ipynb" and "task 2 LPT.ipynb" : These programs contain codes dealing with task2 and 4 for restaurant (RST) and laptop (LPT) respectively.
—"Get final CSV outcomes.ipynb": This program contains codes to use temporary outcomes in former programs to get final submitted outcomes.
—"estimate.ipynb": This program uses trial data to do estimation.

<VIII-2> Folder "data"

This folder contains all original XML dataset, which are downloaded from web.

<VIII-3> Folder "temporary files"

This folder are useless to users, even though, we submitted since we have done so many works and produce so many temporary files, including the tools and dictionaries we trained from data, all files are useful to get next step of outcomes, this folder are used to put all of these temporary files.

<VIII-4> Folder "outcomes"

This folder contains all final outcomes in CSV form.

In files of task1 and 3, the first column are aspect terms, in which "," is used to lay between them, the following columns are aspect categories (if have) and original review texts.
In files of task2 and 4, there are a "()" use to distinguish different aspect terms and categories, in which their is a pair of aspect term/category and its polarity, separated by a ",".

IX. Teamworks

Works	Done by
1, Choose topic of this project	Zhaohan, Zexu, Chenghao
2, Proposal and milestone report	Zhaohan, Zexu, Chenghao
3, Consult to professor for suggestion	Zhaohan
4, Read paper	Zhaohan, Zexu
5, Search useful tools	Zexu, Chenghao
6, Design algorithms for subtask1 and 3	Zexu, Chenghao
7, Design algorithms for subtask2 and 4	Zhaohan, Chenghao
8, Write codes for subtask1 and 3	Zexu
9, Write codes for subtask2 and 4	Zhaohan
10, Write codes for estimation	Chenghao
11, Review codes and modify functions	Zhaohan, Zexu, Chenghao
12, Write report	Zhaohan, Zexu, Chenghao
13, Make PPT and modified	Zexu, Chenghao
14, Presentation	Zhaohan

Instead of do each work separately, our team really works like a real team that finish everything with suggestion of each other. Thus as this table shows, nearly all works we finished as a team, not only single three people put together and without communication to each other, that's boring.

X. Mutual and Self Evaluation

Name	Mutual Evaluation	Self Evaluation
Zhaohan Xi	Could always have new idea especially when some methods are not useful, also could always find a way to modify incomplete methods. He is also good at communicating thus he may give a good presentation.	I am willing to done all additional works in order to get good outcomes of our project, also, I pursue thus there may be many external modification in my work.
Zexu Wang	Has advanced knowledge of programming, could finish complex program tasks efficiently, with nearly no bugs and satisfying outcomes. His codes is easily to read and also in good style.	I could finish programming teamwork quickly, especially in Python. Any works allocated to me will be efficiently solved, which, I believe, can help to push the progress of our project.
Chenghao Ma	Always could estimate some ideas completely, without any negligence of any possible situation. Also always have some new ideas which could always instruct our team find a new path to achieve a goal.	Even though I do not have such advanced programming knowledge, since I am an ISE student, but every little negligence will be caught by my eyes, and I could come up new idea to modify them.

XI. Conference

[1] M. Hu and B. Liu, *Mining and summarizing customer reviews*. Proceedings of the 10th KDD, pp. 168–177, Seattle, WA, 2004.
[2] *B. Liu, Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. Morgan & Claypool, 2012.*
[3] S.-M. Kim and E. Hovy, *Extracting opinions, opinion holders, and topics expressed in online news media text*. Proceedings of the Workshop on Sentiment and Subjectivity in Text, pp. 1– 8, Sydney, Australia, 2006.