



Norwegian University of  
Science and Technology

# Opinion Mining for Song Lyrics

Hanjie Shu

Master in Information Systems

Submission date: November 2010

Supervisor: Kjetil Nørvåg, IDI

Co-supervisor: Robert Neumayer, IDI



# Problem Description

The Master Thesis is supposed to build a basic opinion mining system, including the following work:

1. Use existing natural language processing tools to annotate text with parts-of-speech tags (e.g. verb, noun, . . . )
2. Extract and rank possible objects of interest (e.g. products, general discussions), using named entity recognition
3. Integration of synonym information, e.g. wordnet
4. Identify relevant opinions (e.g. amazing camera, opinions against war, certain groups of people, etc.)
5. Exploit time information for object or `hot topic' extraction
6. Apply the techniques to a collection of song lyrics
7. Adequately display the results with respect to both time and musical genre
8. Explore and analyse the results

Assignment given: 28. June 2010

Supervisor: Kjetil Nørvåg, IDI



## Abstract

The thesis presents an opinion mining system for song lyrics, which can fetch objects of interest and opinion words about them. Finally, opinion mining result is analyzed in terms of time information and musical genre.

In the process of constructing the system, many previous works are reviewed and some of them are applied to the thesis and different methods are compared for reaching a best solution (e.g. explore how to fetch objects of interest).

As well, the evaluation of the system has been done by running experiments with a collection of song lyrics containing hundreds of documents. The result from the system is compared with manual identification. The evaluation result shows that the system basically can present topics of one song lyrics and opinion words about them.

Finally, opinion mining result from a collection of song lyrics can be analyzed and some interesting things are presented, e.g. fetching most common topics, presenting the number of polarity words for each musical type or different year, opinion change on some common topics as time changes.

Besides, we have developed a program in Java for collecting song lyrics on Internet from one website. The program can help us collect thousands of song lyrics and search information of song publishing year or musical genre on Wikipedia.org.

The work in opinion mining for song lyrics is few at present. The thesis finishes an exploration in the subject and the exploration is valuable and useful for future work.

# Preface

This thesis is performed as Master Thesis of Master Program of Information System by Hanjie Shu in The Norwegian University of Technology and Science, NTNU.

The thesis subject is proposed by PhD Candidate Robert Neumayer in NTNU. The thesis intends to explore opinion mining for song lyrics. And the thesis Starts at 28<sup>th</sup> June, 2010 and finishes at 14<sup>th</sup> Nov. 2010.

I would like to thank my main advisor, Robert Neumayer and my supervisor at NTNU, Kjetil Nørkvåg, for their support and feedback.

# Contents

<b>Chapter 1 Introduction.....</b>	<b>1</b>
1.1 Motivation.....	1
1.2 Problem Definition.....	1
1.3 Thesis Outline.....	1
<b>Chapter 2 Background .....</b>	<b>3</b>
2.1 What is Opinion Mining and Sentiment Analysis? .....	3
2.2 Model of Opinion Mining .....	4
2.3 The Development of Opinion Mining and Sentiment Analysis .....	8
2.4 The Applications of Opinion Mining and Sentiment Analysis .....	9
<b>Chapter 3 Research Method .....</b>	<b>12</b>
3.1 Design Science .....	12
<b>Chapter 4 Problem Analysis.....</b>	<b>14</b>
4.1 Part-of-Speech Tagging.....	14
4.2 Fetch Objects of Interest .....	15
4.2.1 TF-IDF SCHEME .....	15
4.2.2 Existing Libraries for Keyphrase Extraction .....	16
4.2.3 Exploration of Methods .....	19
4.3 Fetch Opinion Words.....	25
4.3.1 WordNet.....	25
4.3.2 Word Sense Disambiguation (WSD).....	27
4.3.3 Lexical Resources for Opinion Words.....	30
4.4 Song Lyrics, Songs Time Information and Song Genres .....	31
<b>Chapter 5 Solution .....</b>	<b>32</b>
5.1 System Architecture .....	32
5.2 System components .....	33
5.2.1 Annotation Classes .....	33
5.2.2 UIMA.....	34
5.2.3 OpenNLP .....	35
5.2.4 Lucene .....	36
5.2.5 SentiWordNet .....	38
5.2.6 Subjective Clues .....	39
5.3 Processing Steps of System .....	41
<b>Chapter 6 Experiments and Evaluation .....</b>	<b>44</b>
6.1 Data Preparation.....	44
6.2 Experiment and Evaluation .....	46
<b>Chapter 7 Conclusion and Further Work.....</b>	<b>56</b>
7.1 Conclusion .....	56
7.2 Further Work .....	57
<b>References.....</b>	<b>58</b>
<b>Appendix A: Integration of OpenNLP and UIMA in Eclipse .....</b>	<b>61</b>

<b>Appendix B: Song Lyrics .....</b>	<b>63</b>
<b>Appendix C: Code Examples .....</b>	<b>73</b>
<b>Appendix D: the List of Stop Words for Song's Title .....</b>	<b>85</b>



# Figures

Figure 2- 1: An example of a feature-based summary of opinions [38].....	6
Figure 2- 2: Visualization of feature-based opinion summary and comparison [38] .....	7
Figure 2- 3 : Summary of Reviews [1].....	10
Figure 2- 4: Opinion summarization system .....	10
Figure 4- 1: Operation of Maui [9] .....	17
Figure 4- 2: Workflow of Linguistic Preprocessing [40] .....	18
Figure 4- 3: Comparison of Keywords by Different Methods in UIMA.....	20
Figure 4- 4: Keywords by the Method of Local High Frequency Nouns.....	21
Figure 4- 5: WordNet Online .....	26
Figure 4- 6: Homographs of “Bank” [16] .....	27
Figure 5- 1: System Architecture .....	32
Figure 5- 2: Java Classes of Fetching Keywords .....	33
Figure 5- 3: Annotation in Text with UIMA .....	34
Figure 5- 4: Visualization Representation of Term Polarity of SentiWordNet [15] .....	38
Figure 5- 5: SentiWordNet Online .....	39
Figure 5- 6: Processing Steps of System.....	41
Figure 6- 1: Collection of Song Lyrics.....	44
Figure 6- 2: Information Box of Song on Wikipedia .....	45
Figure Appendix- 1: XML Descriptors of UIMA Wrappers for OpenNLP.....	61
Figure Appendix- 2: Parameter Settings in XML Descriptors .....	62

# Tables

Table 3- 1: Guideline for Design Science Research [8] .....	12
Table 4- 1: Comparison of Keywords by Different Methods-A.....	22
Table 4- 2: Comparison of Keywords by Different Methods-B.....	23
Table 4- 3: Comparison of Keywords by Different Methods-C.....	24
Table 4- 4: Accuracy Comparison of Adjective Sense Determination [6] .....	29
Table 6- 1: Evaluation of Fetching Song Topics-A.....	46
Table 6- 2: Evaluation of Fetching Song Topics-B .....	47
Table 6- 3: Evaluation of Fetching Opinion Words-A .....	48
Table 6- 4: Evaluation of Fetching Opinion Words-B.....	49
Table 6- 5: Opinion Change over Common Topic “American”-A.....	52
Table 6- 6: Opinion Change over Common Topic “American”-B .....	53
Table 6- 7: Hot Topics-A.....	54
Table 6- 8: Hot Topics-B .....	55

# Chapter 1 Introduction

## 1.1 Motivation

Today, people are trying to fetch opinion information and analyze it automatically with computers. As we can see, there are large amounts of information generated from users on the Internet, including product or movie reviews, forum entries, blog and so on. How to analyze the opinions expressed in these documents is attracting more and more attention from research domain and business domain. The new research domain is usually named Opinion Mining and Sentiment Analysis. So far, researchers or developers in the research domain have developed some techniques to the solution of the problem, e.g. the solution of extracting opinions from reviews in the papers [1] and [3].

Now, we hope that we can apply some techniques in the domain Opinion Mining or invent new methods to fetch opinions expressed in song lyrics. Song lyrics are an important part in one song. Through analyzing song lyrics, we can know that what meaning the songs deliver, and what topics presented in the songs, what opinions expressed about the topics. Especially, in political songs, there are strong sentiment and opinion expressions. Further, we can analyze opinion mining result with respect to songs' published time and musical types such as Rock, R&B, Jazz, for example, the change of topics in songs over time, the change of opinions expressed about the same topics over time, different topics presented in different musical types.

## 1.2 Problem Definition

The goal of the thesis is to build an opinion mining system for mining opinions expressed in song lyrics. Usually, the system firstly should fetch objects of interest presented in song lyrics, and then fetch opinions expressed about these objects. Finally, opinion mining result is analyzed in term of songs' published time and musical types.

## 1.3 Thesis Outline

**Chapter 2 Background** introduces some basic concepts in the domain of opinion mining, and the domain's development and application.

**Chapter 3 Research Method** presents the method Design Science adopted in the thesis, and how to apply the method to the thesis.

**Chapter 4 Problem Analysis** presents the concrete analysis of the problem, introduces some fields of interest related with the thesis like part-of-speech tagging, WordNet, Word Sense Disambiguation and so on, and presents exploration process for reaching a best solution.

**Chapter 5 Solution** presents how to build the opinion mining system, the architecture and main components of the system, and the introduction of system processing steps.

**Chapter 6 Experiments and Evaluation** presents experiments that are done on the system built and analyzes experiment results and gives some evaluation with the system built if it is good.

**Chapter 7 Conclusion and Further Work** presents what have been done in the thesis, and suggests the further work.

## Chapter 2 Background

This chapter will present some background knowledge relevant with the thesis. We will introduce what are opinion mining and sentiment analysis, and model of opinion mining, and their development process, and their applications.

### 2.1 What is Opinion Mining and Sentiment Analysis?

Textual information includes two kinds: facts information and opinion information. Facts information is objective statement about objects, and opinion information is subjective statement that expresses persons' opinion about objects. Most of researches on text information processing focus on mining and retrieval of facts information. But more and more researchers and business man begin to become interested on mining of opinion information.

The rise of World Wide Web brings us many user generated information (e.g. forum post, blog, review), which contains a large number of opinion information. When one wants to see how good one product he or she wants to buy is, it is not necessary to ask other friends if we can fetch opinion information about the product on Web. Before political election, the computational survey about what voters think also can be done like this. Similarly, manufacturers can do market investigation through mining opinion information on Internet in order to know what products current customers really like. All these reasons push the development of research on opinion mining and sentiment analysis.

#### Opinion Mining

The term *opinion mining* appears in the paper [25] "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews" by Dave et al. They define the ideal opinion mining tool:

*"Process a set of search results for a given item, generating a list of product attributes (quality, features, etc.) and aggregating opinions about each of them (poor, mixed, good)."*

However, the term has recently been interpreted more broadly, containing many different aspects of analysis in evaluative text.

#### Sentiment Analysis

The history of the term *sentiment analysis* parallels that of the term *opinion mining* in certain respects. The paper in 2001 by Das and Chen [34] and Tong [35] appears the term "sentiment", which is used in reference to the automatic analysis of evaluative text and tracking of the predictive judgments. In many papers, the term "sentiment

analysis” focuses on the specific application of classifying reviews (positive or negative). So some people suggest the term should refer specifically to this narrow task. However, many still explain the term more broadly to mean computational treatment of opinion, sentiment, and subjectivity in text.

Therefore, when broad interpretation is applied, opinion mining and sentiment analysis denote the same study field. Then, we will present more concrete definitions of some elements contained in the study like opinion holder, feature, and semantic orientation of opinion and so on. These contents will be included in the model of opinion mining given below. The model is referred to the article of Bing Liu [38].

## 2.2 Model of Opinion Mining

Opinion can be expressed on anything like product, movie, topic, individual, organization, or event. The term *object* is used to denote the entity on which opinion is given. An object can be decomposed with the part-of relationship. It has a set of components (parts) and a set of attributes.

*Definition of **object**: An object is an entity which can be topic, product, event, individual or organization. It is associated with the pair  $O: (T, A)$ , where  $T$  is a hierarchy of components and sub-components of object  $O$ .  $A$  is a set of attributes of object  $O$ . Each component has its own sub-components and a set of attributes.*

However, simply, we often use the term *feature* to represent components and their attributes. One object itself is also a feature.

We define one document  $d$ , which can be a movie review, a blog, a forum post that evaluates on some objects. One document  $d$  consists of some sentences, so  $d = \{s1, s2, s3, s4 \dots\}$ .

*Definition of **opinion passage on one feature**: the opinion passage about a feature of one object is a group of consecutive sentences in one document  $d$ . It expresses a positive or negative opinion on the feature. Several sentences can together express opinions on the same feature of one object, and it is also possible that a single sentence express opinions on more than one feature.*

*Definition of **opinion holder**: one opinion holder means a person or an organization who publishes the opinion on an object. For example, author of forum post, blog, news article.*

*Definition of **semantic orientation of an opinion**: the semantic orientation of an opinion on one feature means positive, negative, or neutral.*

## Model of Feature-based Opinion Mining

So, we put things above together. One object  $O$  consists of a set of features  $F$ ,  $F = \{f_1, f_2, f_3, f_4, \dots, f_n\}$ , which include object itself. In an evaluative document  $d$ , opinion holder expresses opinions (positive, negative or neutral) on one feature or several features of one object. Opinion mining task is to fetch all these information.

In a given evaluative document  $d$ , the output of opinion mining result consists of a set of quadruples. Each quadruple can be denoted by  $(H, O, f, SO)$ , where  $H$  is opinion holder,  $O$  is the object,  $f$  is a feature of the object  $O$ , and  $SO$  is the semantic orientation of opinions expressed on the feature  $f$  in one sentence of  $d$ . Neutral opinions are usually ignored in the result.

## Three Main Technical Problems

Finally, Liu [38] summarizes three main technical problems in opinion mining task:

Problem 1: Extracting objects features, for example, in the sentence “quality of the clothes is good”; *quality of clothes* is considered the object of interest.

Problem 2: Opinions on the feature should be fetched and determine semantic orientation of the opinions (positive, negative, or neutral). The word *good* in the example above is the opinion word, and it should be positive.

Problem 3: Grouping synonyms of features as maybe different opinion holders have different names for the same feature.

## Opinion Summary

We can summarize opinion mining result in many ways. There is an example showed in the Figure 2-1 below, which summarizes opinions on Digital camera 1. CAMERA represents camera itself. Two features **picture quality** and **size** are also showed. In each feature, the number of positive and negative reviews is given. As in the feature of **picture quality**, it has 123 positive reviews and 6 negative reviews. If users want to see the whole sentence where positive or negative reviews are expressed, they can read from the list of the right <individual review sentences>.

*Digital\_camera\_1:*

CAMERA:

Positive: 125 <individual review sentences>

Negative: 7 <individual review sentences>

Feature: **picture quality**

Positive: 123 <individual review sentences>

Negative: 6 <individual review sentences>

Feature: **size**

Positive: 82 <individual review sentences>

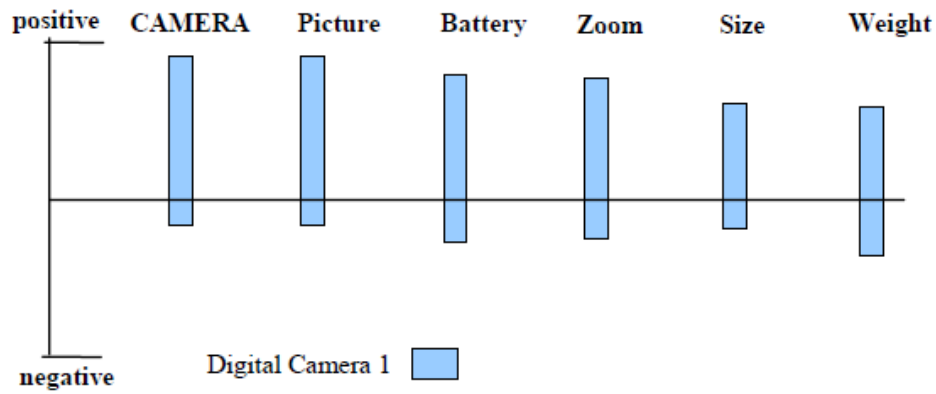
Negative: 10 <individual review sentences>

...

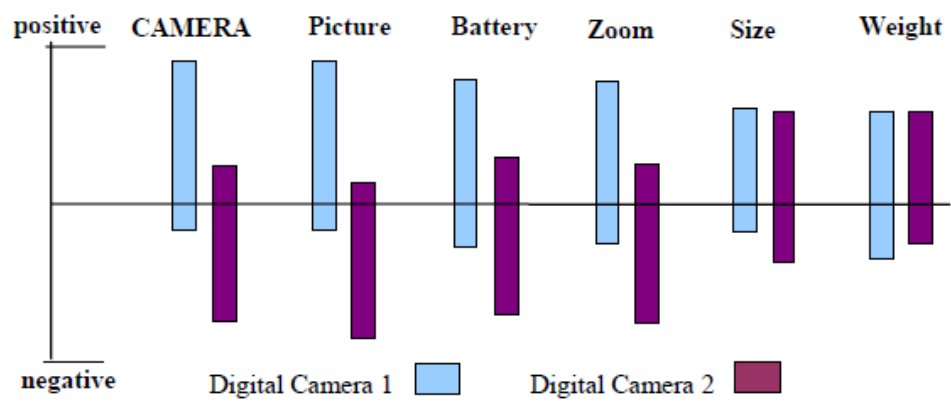
Figure 2- 1: An example of a feature-based summary of opinions [38]

The example also can be visualized using bar chart. The Figure 2-2 below shows the kind of bar chart. In the Figure 2-2 (A), the part above X-axis of each bar gives the number of positive reviews on one feature, the name of which is written on the top. The part below X-axis on the bar means the number of negative reviews. It is more interesting that the Figure 2-2 (B) [39] represents a comparison between two digital cameras. Different colors stand for different cameras. People can see easily the comparison in the same feature in different two cameras.





(A) Feature-based summary of opinions on a digital camera



(B) Opinion comparison of two digital cameras

Figure 2- 2: Visualization of feature-based opinion summary and comparison [38]

## **2.3 The Development of Opinion Mining and Sentiment Analysis**

The area of opinion mining and sentiment analysis has recently enjoys a huge burst of research activities. The early projects on beliefs [32, 33] maybe are seen as forerunners of the area. The year 2001 marks the beginning of popularity of the research on the subject, and then there are hundreds of papers published. Some factors push the development of the subject:

1. The increasing of machine learning methods in natural language processing and information retrieval.
2. The blossoming of World Wide Web provides training datasets for machine learning algorithms.
3. The realization of commercial and intelligent applications that the area provides.

Researches on opinion mining start with identifying opinion words, e.g. beautiful, nice, and ugly. Researchers work on determining semantic orientation of opinion words as well. Some methods are explored: some use linguistic rules to identify opinion words and their orientations from a large corpus; some use a small set of given seed opinion words to find their synonyms and antonyms. Sentiment classification of product reviews at the document level is the next major development, classifying each review document based on sentiment that they express about one object (positive or negative). Some researchers also studied sentence-level sentiment classification. More, some summarize a model of opinion mining, and some work on the problem of mining opinions from comparative sentences, and so on.

## 2.4 The Applications of Opinion Mining and Sentiment Analysis

There are some applications listed below:

Opinion Mining and Sentiment Analysis can be used in review-related websites, which aggregate reviews and solicit reviews. These websites can put review-oriented search engine as the tool to help them gather review information, then solicit these information and fetch usable information with the methods of opinion mining and sentiment analysis, finally summarize them, providing them to users.

As well, Sentiment analysis and opinion mining systems are applied to recommendation systems [27, 28]. For example, items that receive a lot of negative feedback will not be recommended.

Sentiment analysis has been suggested as a technology used in eRulemaking [29, 30, 31], which does automatic analysis of opinions about government regulations and national policy.

Opinion mining and sentiment analysis is also used in business. For example, one company wants to investigate “why they have such a low sale of their products?” Perhaps, they can investigate it on Internet using some information sources like blog, newsgroup, and review-related websites and so on. Later, these information fetched is processed, finally one analysis result is given and answers the question asked above.

Next, we want to present some concrete examples and see how these systems work.

### **Opinion Summarization System**

Hu and Liu in [1] present an opinion summarization system for mining product opinions and opinions sentimental analysis. The input to the system is a product name and reviews of the product; the output is summary of the reviews as showed in the Figure 2-3. The system contains two main steps: opinion identification and opinion sentimental determination. The general architecture of the system is showed in the Figure 2-4. Firstly, the system has the part-of-speech tagging for review words. Then the system finds features of the product that occur most frequently in the reviews. We can say that the most frequent features are the features that reviewers are most interested in. Not all the extracted features in the last step are useful, so the system will prune the features by Compactness pruning that checks the features containing at least two words, and Redundancy pruning that focus on removing redundant features that contain single words. Next, we see the step of opinion word extraction. The system considers a word is opinion word if the word is an adjective and appears in the same sentence as an extracted feature and its position is close to the feature. At the same time, the system considers that it is possible that people are also interested in some infrequent features. So the system extracts the infrequent features with opinion words that have been extracted in the last step. Finally, the system works on opinion

sentence orientation determination (i.e. positive or negative). The semantic orientation of each opinion word extracted is identified with a bootstrapping technique and the WordNet [2]. Then the whole sentence's semantic orientation is determined based on the dominant orientation of the opinion words in the sentence.

```

Digital_camera_1:
picture quality:
    Positive: 253    <individual reviews>
    Negative: 6     <individual reviews>
size:
    Positive: 134   <individual reviews>
    Negative: 10    <individual reviews>
...

```

Figure 2- 3 : Summary of Reviews [1]

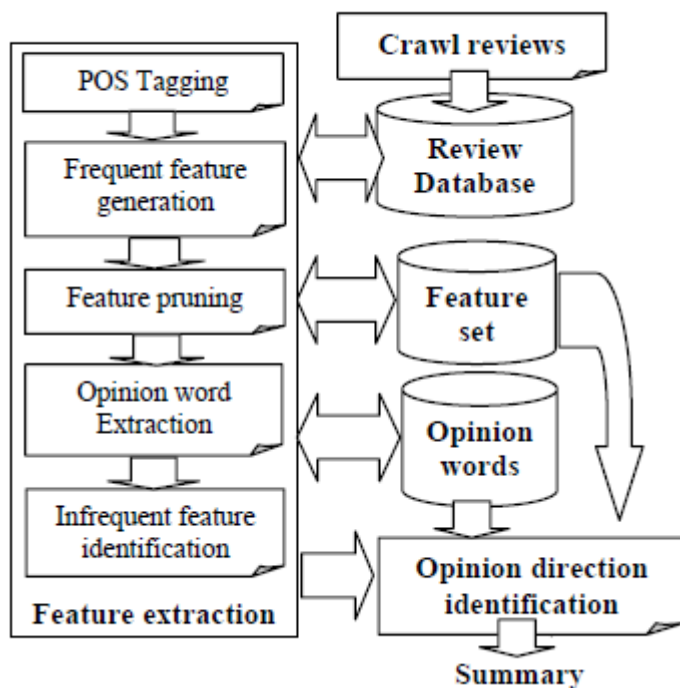


Figure 2- 4: Opinion summarization system

## OPINE

OPINE [3] is an unsupervised information extraction system which extracts product features and their opinions from reviews. It is built on top of KnowItAll, a web-based, domain-independent information extraction system [4]. KnowItAll instantiates relation-specific generic extraction patterns into extraction rules which find candidate facts. KnowItAll's Assessor then assigns a probability to each candidate. The Assessor uses a form of Point-wise Mutual Information (PMI) between phrases that is estimated from web search engine hit counts [5]. The PMI scores computed are converted to binary features for a Naïve Bayes Classifier, which produces a probability associated with each fact.

The system finds explicit features of products by extracting the noun phrases from reviews and retaining those with high frequency and evaluating the remaining noun phrases. The system evaluates the remaining noun phrases by computing the PMI scores between the phrases and discriminators associated with the product class (e.g. “of scanner”, “scanner has”, “scanner comes with”, etc. for the Scanner class).

In finding opinion words, OPINE contains two steps: (1) identify potential opinion words, (2) identify actual opinion words from potential opinion words. The intuition of the way to find potential opinion words is the same as Opinion Summarization System [1] described above. It is just that opinion words appeared nearby product features. But, compared with [1], OPINE adopts a more advanced method, which is the use of extraction rules. Then, by identifying semantic orientation of potential opinion words (i.e. positive or negative semantic orientation), OPINE distinguishes the actual opinion words from the rest.

## Chapter 3 Research Method

The chapter presents the research method “Design Science” for the thesis, and presents how the thesis matches guidelines of the method.

### 3.1 Design Science

The research method used in the thesis is Design Science. The fundamental principle of design science research is that knowledge and understanding of a design problem and its solution is acquired in the building and application of an artifact. The Table 3-1 below from the paper [8] presents design science research guidelines assisting researchers, reviewers, editors, and readers to understand the requirements for effective design science research.

<b>Guideline</b>	<b>Description</b>
<b>Guideline 1: Design as an Artifact</b>	Design science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
<b>Guideline 2: Problem Relevance</b>	The objective of design science research is to develop technology-based solutions to important and relevant business problems.
<b>Guideline 3: Design Evaluation</b>	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
<b>Guideline 4: Research Contributions</b>	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations and/or design methodologies.
<b>Guideline 5: Research Rigor</b>	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
<b>Guideline 6: Design as a Search Process</b>	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
<b>Guideline 7: Communication of Research</b>	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 3- 1: Guideline for Design Science Research [8]

Next, how the thesis matches the seven guidelines will be presented.

- **Guideline 1 requires producing a viable artifact.**

In the thesis, the artifact produced is an opinion mining system for song lyrics used to fetch objects of interest in songs lyrics and opinions about them. Finally, the mining result will be analyzed in some ways.

- **Guideline 2 says that objective of design science research is to develop technology-based solutions to important and relevant business problems.**

The thesis is an exploration of opinion mining in song lyrics. This is a new kind of document explored in the subject opinion mining. Song lyrics often express some sentiment. It will be interesting and can help people do automatic analysis of song lyrics.

- **Guideline 3 requires rigorous and well-executed evaluation methods**

The thesis evaluates results with comparing with manual identification.

- **Guideline 4 requires that design science research can provide clear and verifiable contributions**

The thesis explores opinion mining for song lyrics. Few research focus on opinion mining of song lyrics. This is a new contribution to the subject opinion mining. We also present some interesting analysis from opinion mining result.

- **Guideline 5 requires rigorous research methods in both construction and evaluation of design artifacts.**

In this thesis, some methods or resources for construction of the system are chosen from many literatures published before.

- **Guideline 6 says that the search for an effective artifact requires using available methods to reach desired ends.**

The thesis is evaluated in the end and there are the improvement suggestions given. They serve the iterative process.

- **Guideline 7 says that design science research should be presented not only to technological personnel but also to management audiences.**

Common technical terms will be used and keep intended audiences in mind while writing the thesis.

## Chapter 4 Problem Analysis

The chapter presents a concrete analysis for the problem to be solved in the thesis. It is divided into four parts: Part-of-speech tagging and Fetch Objects of Interest and Fetch Opinion Words and Song Information (Song published time and song musical genre).

### 4.1 Part-of-Speech Tagging

Technique of part-of-speech tagging is often used in previous similar work since objects of interest are often nouns, and opinion words are often adjectives or adverbs.

#### Part-of-speech tagging

Part-of-speech tagging can be seen as the process of assigning a part of speech or other lexical class marker to each word in a corpus [36]. The parts of speech tags divide the words into different categories based on different roles they play in one sentence. The sets of tags can be various, and the most common set of tags contain article, noun, verb, adjective, and preposition, number and proper noun.

Part-of-speech tagging can be used in word sense disambiguation [16] that will be discussed further in the section 4.3.2. Firstly, we judge part of speech tag of one word in its context. Then, other senses of the word not with the part of speech tag are removed. We only consider the senses of the word in the category of the part of speech. Although the word maybe still contains more than one sense, at least the scope is smaller.

#### Part of Speech

Adjectives have been considered as important features by many researchers. The paper [37] revealed a high correlation between the presence of adjectives and sentence subjectivity. The presence or polarity of adjectives is seen in many approaches when deciding the subjectivity or polarity status of textual units.

Adjectives are good indicators of a sentence being subjectivity, but other parts of speech also contribute to the judgment of subjectivity, e.g. verb (“like”, “hope”, “hate”).



## 4.2 Fetch Objects of Interest

When we consider how to fetch objects of interest, we try to learn previous work and we also do exploration for getting a best solution by comparing between different potential methods.

### 4.2.1 TF-IDF SCHEME

TF-IDF Scheme is often used in information retrieval and text mining. It describes how important a word is for a document in a collection. The technique maybe can help us to rank objects of interest and fetch the most interesting objects.

TF-IDF Scheme consists of two parts: TF and IDF. TF represents the occurrence frequency of a term in a document. It provides one measure of how well a term describes document content. IDF represents the inverse occurrence frequency of a term in a collection of documents. The motivation of IDF usage is that terms appearing in many documents in a collection are not very useful to distinguish relevant documents for a query from non-relevant documents.

We can use an example to further illustrate why we adopt TF-IDF Scheme. Suppose we want to find relevant documents for the query “a red hat”. The simplest way is to fetch the documents that contain all the three terms “a”, “red”, “hat”. But it still leaves many documents. Then, we consider TF factor, count the frequency of each term in a document and sum them. We fetch the documents with high term occurrence frequency as relevant documents. However, the term “a” is so common in a collection of documents. This will intend to emphasize documents that contain more “a”, without giving enough weight to more meaningful terms “red” and “hat”. So, the IDF factor is imported to balance the effect, diminish the weight of terms appearing very frequently in a collection, and increase the weight of terms occurring rarely.

Baeza-Yates & Ribeiro-Neto [14] introduces the mathematic formula of TF-IDF scheme in their book.

**Definition:**

$N$  is the total number of documents in a collection;

$n_i$  is the number of documents in which the index term  $k_i$  appears;

$freq_{i,j}$  is the raw frequency of term  $k_i$  in the document  $d_j$ ;

Then, the normalized frequency  $f_{i,j}$  of term  $k_i$  in document  $d_j$  is given by

$$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}} \quad (4.1)$$

where the maximum is computed over all terms appearing in the document  $d_j$ .

Further, let  $idf_i$  inverse document frequency for  $k_i$  be given by

$$idf_i = \log \frac{N}{n_i} \quad (4.2)$$

Finally, the term weight is given by

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i} \quad (4.3)$$

or by a variation of this formula.

## 4.2.2 Existing Libraries for Keyphrase Extraction

Moreover, we also find that there are some existing libraries for keyphrase extraction, which means extracting keywords that can describe the main topic in text analyzed. There are two projects introduced that have explored in the subject.

### Maui Topic Indexing Algorithm

Maui has been developed as a part of Olena Medelyan's PhD project [9]. It combines four software components Kea [10], Weka [11], Jena [12], and Wikipedia Miner [13] with classes created specifically for Maui to form a single topic indexing algorithm. Figure 4-1 shows a general architecture of Maui, we can see that it contains 4 main steps:

1. Generating candidate topics
2. Computing their features
3. Building the topic indexing model
4. Applying the model

The left in the Figure 4-1 depicts how to build the indexing model from manually assigned topics and the right presents the process of applying the model to new documents. Maui implements a supervised machine learning approach, where a small training set provides a model that can be used for fresh documents that had not been seen at training time.

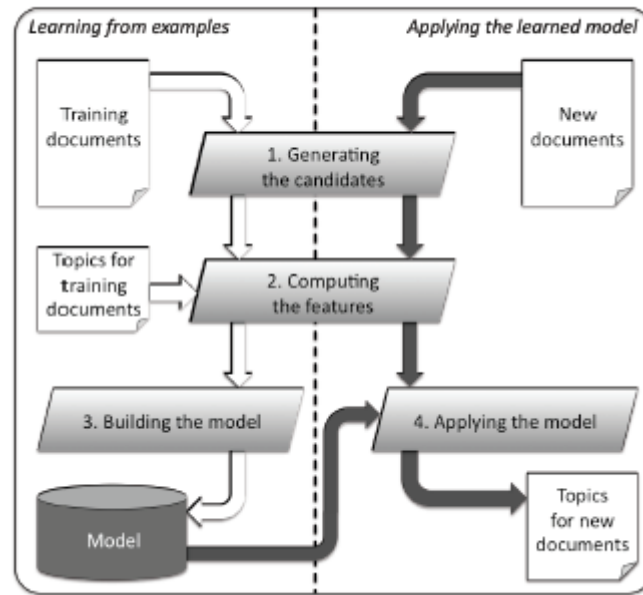


Figure 4- 1: Operation of Maui [9]

### Generating Candidate Topics

For the first step candidate topics generation in the Figure 4-1 above, it contains 4 phases:

**Phase A:** Document text is analyzed to identify initial syntactic boundaries. Maui uses Kea's PhraseFilter algorithm for this.

**Phase B:** Maui extracts all subsequences of tokens of length  $n$  ( $n$ -grams) in each line.

**Phase C:** The  $n$ -grams are conflated to a set of candidate topics.

**Phase D:** Finally, Maui normalizes the occurrence positions by document length and the occurrence frequencies by the number of candidates. These values are stored and the candidates list is passed to the next step: computing the features.

Actually, Maui includes three kinds of different tasks:

**Automatic Tagging** candidate topics of which are sequences that do not begin or end with a stopword in documents analyzed;

**Term Assignment's** candidate topics are from a controlled vocabulary.

**Indexing with Wikipedia,** its candidate topics are from Wikipedia articles.

Maui will be tested with the task of automatic tagging in the exploration of the methods of fetching objects of interest for song lyrics later.

## SmILE Keyphrase Extraction

The sub-component Keyphrase Extraction in the project SmILE [41] is the work described in Master's thesis of Alexander Schutz [40]. There is a simple description below of how the component of keyphrase extraction works.

### Linguistic Preprocessing

The Figure 4-2 below shows the workflow of linguistic preprocessing. It starts the process with *language identifier*, which identifies what kind of language (e.g. English, French, or German) is used in the input document, in order to select right subsequent processing resources. Next, the input text needs to be *tokenized* and *split into sentences*. The *stopword analyser* denotes each token whether it represents a stop word or not, rather than eliminating the token even if the token is stop word. Now it goes to *part-of-speech tagger*, which use different sets of taggers based on different language. Then it arrives the step of *Morphological Analyser*. Now, tokens are enriched with part-of-speech and lemma information. Finally, larger syntactic units are identified by the *noun chunker*, and *frequency analyzer* produces frequency lists of overall word form and lemma occurrence.

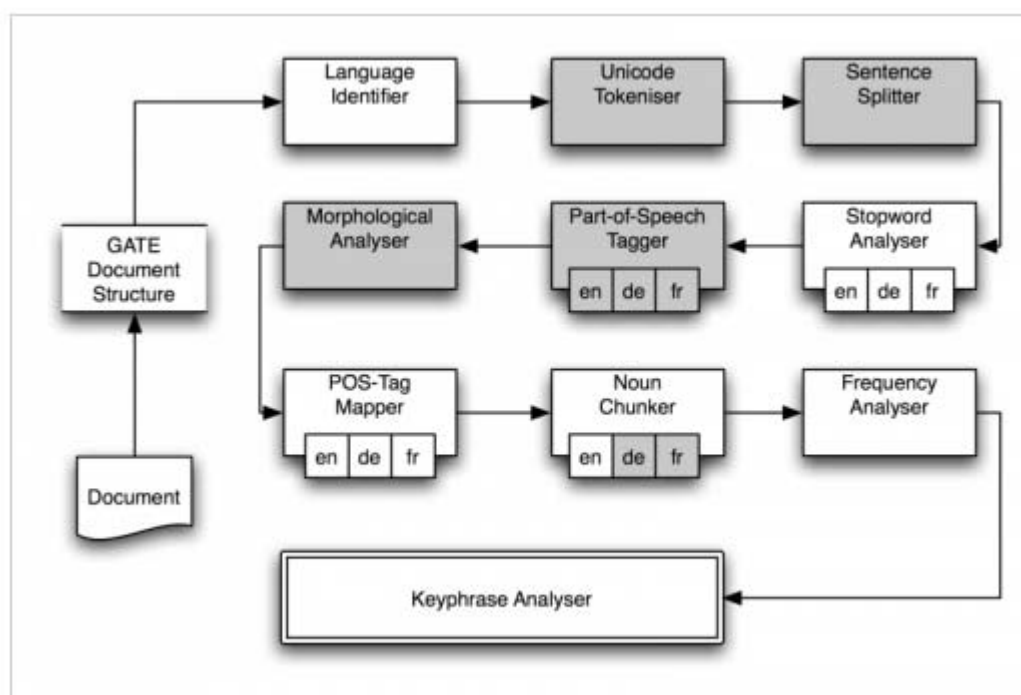


Figure 4- 2: Workflow of Linguistic Preprocessing [40]

### Keyphrase Extraction Procedure

The Figure 4-2 above goes to the final component Keyphrase Analyser, which includes several steps. Firstly, it does the (statistical) lexical analysis to determine the most significant single word terms, which is then extracted including their immediate contexts to form complex terms. Next, it groups similar complex terms, and selects a

representative for each group as a keyphrase candidate, and eventually analyzed the extracted keyphrase candidates in order to determine a confidence score in its context.

It is said that “Due to the nature of the underlying processing steps it is very likely that short documents (<500 words) and not very coherent documents (no *real* sentence structure, many bullet points) will produce suboptimal results.” in the introduction of the method of Keyphrase Extraction in the project SmILE [41]. One Song lyrics is often a short document and less than 500 words, and often contains some not real sentence like chorus as well. So we consider that the method is not suitable for fetching keyphrases of song lyrics.

### **4.2.3 Exploration of Methods**

In the part, we present the process of our exploration for the method of extracting objects of interest (Keywords). We suggest five methods and try to compare them. Five methods are introduced as following:

#### **Fetching Keywords by Songs’ Title**

We can see that many songs’ titles can reflect what the songs talk about. Therefore we want to test the way to see if it is suitable.

#### **Fetching High Local Frequency Nouns**

Usually, the keywords are nouns, so we try to fetch high occurrence frequency nouns in song lyrics.

#### **High Document Frequency Nouns**

We consider that we have to fetch the same objects as many as possible if we want to see the change of opinions about the same objects in different songs. So we want to see which nouns are popular in the collection of song lyrics.

#### **Nouns’ Local Frequency & Nouns’ Document Frequency**

If we only consider local high frequency nouns, then we can not fetch the same objects of interest as many as possible. And, if we only consider nouns’ document frequency, then the nouns to fetch can not reflect well the topics of song lyrics. Therefore, we consider their combination. We compute the weights of nouns in each song lyrics by multiplying nouns’ local frequency by nouns’ document frequency. We fetch nouns with bigger weight value.

#### **Maui [9]**

Maui is an existing method of extracting keywords as mentioned before. Maui firstly builds topics index model through training, then apply the model to analyze new documents.

We implement these methods in the framework of UIMA, where it is interesting to see the comparison of results showed like the Figure 4-3 below. But there is the possibility of overlap, and then we can choose only one type to show as the Figure 4-4, which shows the keywords fetched by the method of high local frequency nouns. Here, for the type KeywordsByTitleIn, it only annotates tokens that are song’s title words in

song lyrics. Of course, song's title words maybe do not appear in song lyrics. When comparing these methods, we still use concrete song title words rather than title words only annotated in song lyrics.

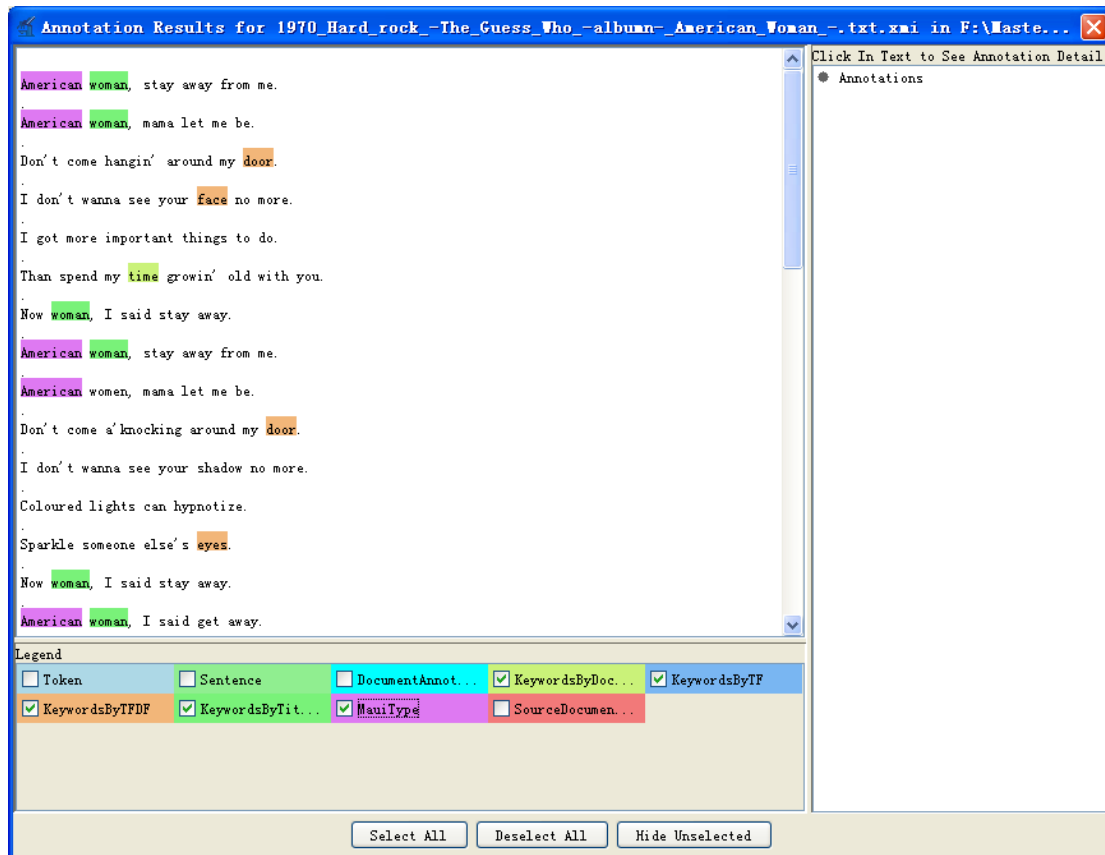


Figure 4- 3: Comparison of Keywords by Different Methods in UIMA

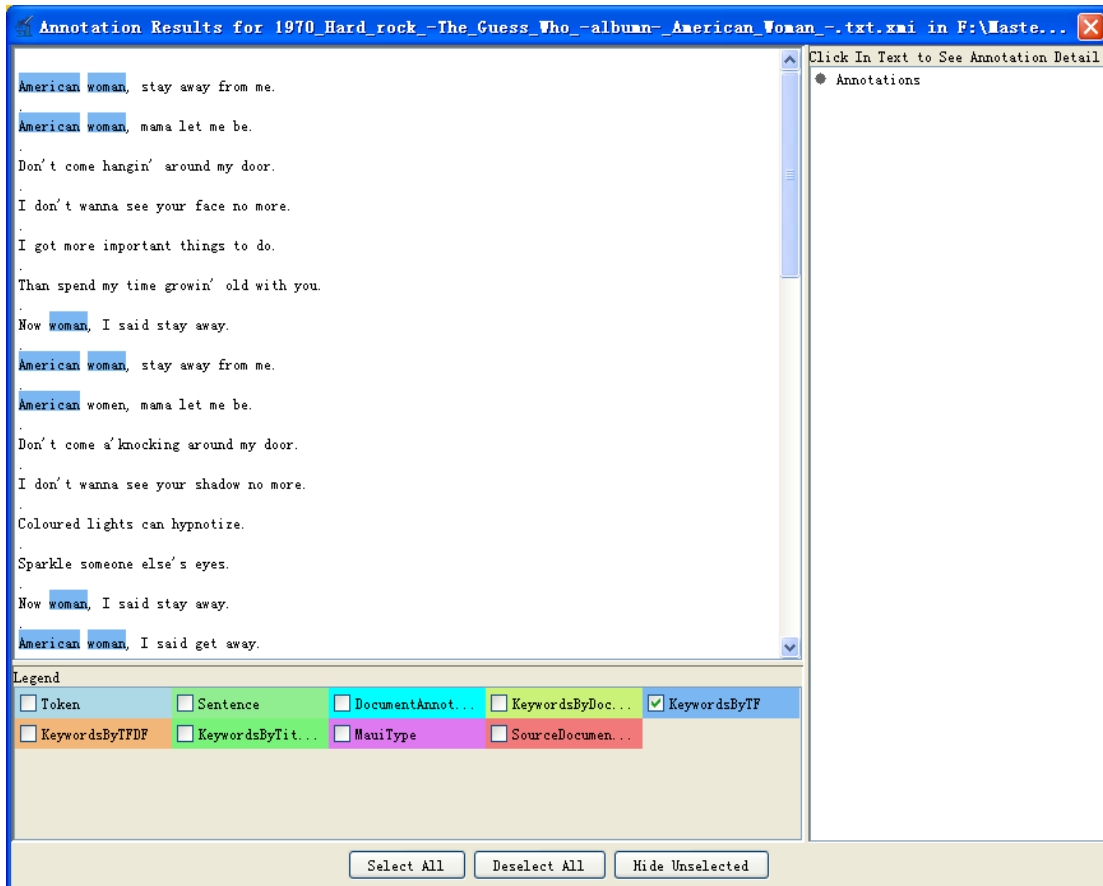


Figure 4- 4: Keywords by the Method of Local High Frequency Nouns

We run the experiment in the collection of 506 song lyrics. Then we randomly extract 10 songs lyrics attached in Appendix B to analyze them, and the analysis result is showed in the Table 4-1, Table 4-2, and Table 4-3 below. In the second column, it is about Song Information including song published time on the first line, song musical type on the second line, artist on the third line, and the name of song album at last. Other columns, from left to right, it is **Manual Identification** which is from our identification for songs' topics, **Title** that is the title of songs, **Local High Frequency Nouns** that is nouns with local high frequency, **High Document Frequency Nouns** that is nouns with high document frequency in the whole collection, **Big Mixed Weight Nouns (TF\*DF)** is nouns with bigger mixed weight value, and **Maui** at the last column that is keywords from the Algorithm Maui.

	<b>Song Information</b>	<b>Manual Identification</b>	<b>Title</b>	<b>Local High Frequency Nouns</b>	<b>High Document Frequency Nouns</b>	<b>Big Mixed Weight Nouns (TF*DF)</b>	<b>Maui</b>
1	1978, Reggae, Bob Marley, Babylon By Bus	love	Is This Love	Love shelter Bed Feeling	NONE	Heads Share Bed	Single bed
2	1984, Folk, The Pogues, The Ultimate Collection	Life philosophies	Streams of Whiskey	Where Streams whiskey	Time Life way	Simple Whiskey Nothing	Whiskey are flowing
3	1989, Grunge, Nirvana, Bleach	Daddy's girl	Negative Creep	Creep Daddy Girl	NONE	Range Daddy	No girl

Table 4- 1: Comparison of Keywords by Different Methods-A



	<b>Song Information</b>	<b>Manual Identification</b>	<b>Title</b>	<b>High Local Frequency Nouns</b>	<b>High Document Frequency Nouns</b>	<b>Big Mixed Weight Nouns (TF*DF)</b>	<b>Maui</b>
4	1992, Grunge, Mudhoney, Superfuzz Bigmuff Plus Early	Chain that door, girl	Chain that door	Girl Door loser	way	Outta	know mean
5	1992, Grunge, Alice in Chains, Dirt	I don't mind.	Angry Chair	Chair Boy Pink Cloud Knees Time Mind Lost Can't I-I-I	Way Time	Walls Boy Cloud	boy knees pray
6	1998 Metal, System of A Down, System of A Down	Jesus Philosophy	Suite-pee	Philosophy Die Christ	way	Floor Motherfucker Needs	Jesus thy Christ

Table 4- 2: Comparison of Keywords by Different Methods-B

	<b>Song Information</b>	<b>Manual Identification</b>	<b>Title</b>	<b>High Local Frequency Nouns</b>	<b>High Document Frequency Nouns</b>	<b>Big Mixed Weight Nouns (TF*DF)</b>	<b>Maui</b>
7	1999, Country, Wilco, Summerteeth	In a future age	In a future age	Page Future age	NONE	Shakes Dares Mark	shakes bend barking
8	2002, Country, Dixie Chicks, Home	Wedding Ring	White Trash Wedding	Ring Hand way	way	Nip Gin	afford ring wearing
9	2002, Slow Rock, Coldplay, A Rush of Blood to the head	whisper	A Whisper	Whisper Sound Ticking Clocks	none	Clocks Remember Questions	whisper ah
10	2003, Pop, Britney Spears, In the zone	shadow	Shadow	Shadow Nobody Arriving	None	NONE	Love Shadow Bright
			6/10	8/10	0/10	1/10	5/10

Table 4- 3: Comparison of Keywords by Different Methods-C

### **Analysis of Experiment Results**

At the last row of the Table 4-3, we summarize the number of songs, topics of which are fetched correctly (including the situation of correctly partially). There are 6 songs' topics judged correctly with the title information, 8 songs judged correctly with the method of high local frequency nouns, 0 songs for high document frequency nouns, 1 song for mixed weight method, 5 songs for Maui. Therefore, we can see that the most correct methods to fetch song topics are title and high local frequency nouns. Document frequency and Mixed Method do not reflect the real topics of songs very well, since both of them have to consider the document frequency of terms. Maui's result is also not as good as the title method and the way of nouns' local frequency. As showed in the first song in the Table 4-1 above (1978, Reggae, Bob Marley, Babylon By Bus, Is This Love), the real song keywords is "love", the method of Title presents us the keywords "Is This Love" and The method of high local frequency nouns presents us the keywords "love, shelter, bed, feeling", but keywords from Document Frequency is none, keywords from Mixed Method is "heads, share, bed", and keywords from Maui is "Single bed".

## **4.3 Fetch Opinion Words**

In the part, we analyze how to fetch opinion words. Previous work has developed some lexicon for opinion words or lexicon containing polarity description of words. Some lexicon is based on WordNet that is introduced next. Word Sense Disambiguation is also very important in the area since one word maybe contains several different senses, which means having different sentiment polarity.

### **4.3.1 WordNet**

#### **What is WordNet?**

WordNet is a large lexical database of English and was created and is being maintained at the Cognitive Science Laboratory of Princeton University under the direction of psychology professor George A. Miller. Its design is inspired by psycholinguistic theories of human lexical memory and its development began in 1985 [20]. The initial idea was to provide an aid to use in searching dictionaries conceptually rather than alphabetically. Over the years, the project received funding from government agencies interested in machine translation.

In WordNet, nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called synsets, each of which expresses a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet can be used online [18] as showed in the Figure 4-5 below and is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

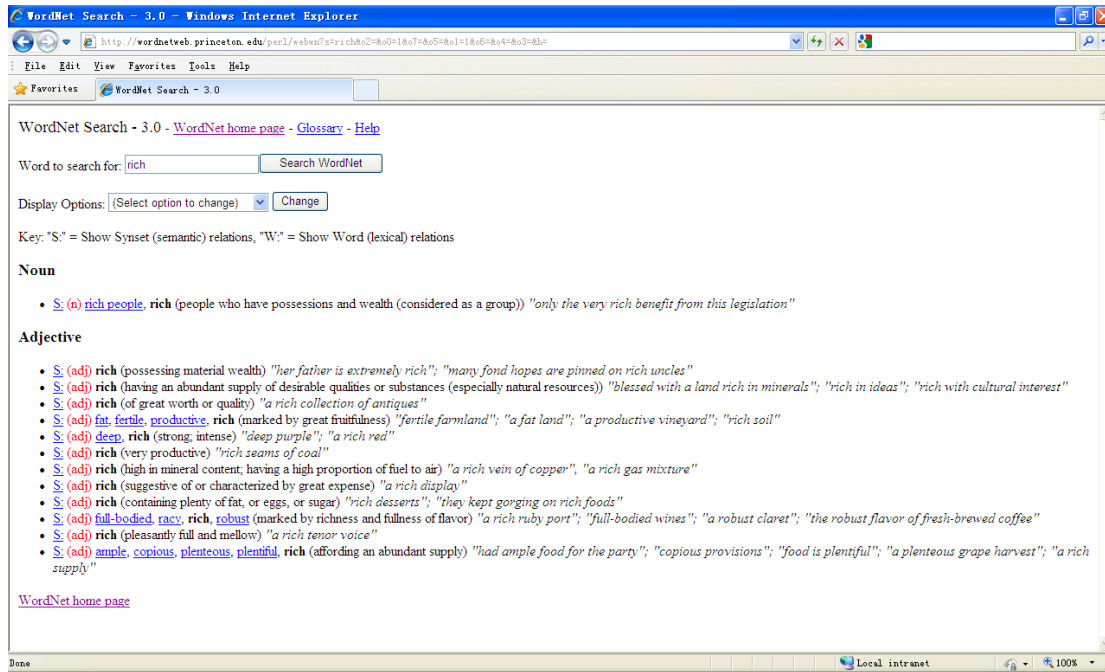


Figure 4- 5: WordNet Online

### Nouns in WordNet

The paper [21] said that a superordinate term and distinguishing features that definitions of common nouns typically give provides the basis for organizing nouns files in WordNet. The superordinate relation (hyponymy) generates a hierarchical semantic organization. For example, there is a lexical tree like this: oak->tree->plant->organism. We can see that there is the relationship “is a kind of” between them. Distinguishing features are entered in order to create a lexical inheritance system, in which each word inherits the distinguishing features of all its superordinates. There are three types of distinguishing features discussed: attributes (modification), parts (meronymy), and functions (predication), but only meronymy is implemented in nouns files that is said from the report of 1993. We can see an example: *canary* with distinguishing features:

- (1) *Attributes: small, yellow*
- (2) *Parts: beak, wings*
- (3) *Functions: sing, fly*

### Adjectives in WordNet

The paper [22] talks that WordNet divides adjectives into two major classes: descriptive and relational. Descriptive adjectives are what one usually thinks of when adjectives are mentioned. A descriptive adjective ascribes a value of an attribute to a noun. For example, *heavy* is a value of the attribute *weight*, which is the weight of *package*. Relational Adjectives are associated with some nouns and play a role similar to that of a modifying noun. For example, the adjective word *atomic*, we can see that both *atomic bomb* and *atom bomb* are admissible.

### 4.3.2 Word Sense Disambiguation (WSD)

Word Sense Disambiguation is the process of disambiguating senses of one word since one word maybe contains several senses. In the thesis, WSD will be used for getting one word’s right sentiment polarity since different word senses have different sentiment polarity. We will present some methods that have been explored in the area so far. We can study from these methods and apply some of them to our thesis.

#### Usage of Part-of-speech tagging in WSD

The paper [16] presents relationship between part-of-speech tagging and word sense disambiguation through some experiments. Firstly, there is the experiment on the lexicon LDOCE, a dictionary of English designed for students which contains about 36,000 word types. The senses for each word type are grouped into *homographs*. If there is only one homograph for that word type, we call the situation *monohomographic*, otherwise, we call *polyhomographic*. For example, the word “bank” has more than one homograph as showed in the Figure 4-6 below.

<p>bank <i>n</i></p> <p>1 land along the side of a river, lake, etc.</p> <p>2 earth which is heaped up in a field or a garden, often making a border or division.</p> <p>3 a mass of snow, mud, clouds, etc. <i>The banks of dark cloud promised a heavy storm.</i></p> <p>4 a slope made at bends in a road or race-track, so that they are safer for cars to go round.</p> <p>5 SANDBANK: <i>The Dogger Bank in the North Sea can be dangerous for ships.</i></p> <p>bank <i>v</i></p> <p>1 (of a car or aircraft) to move with one side higher than the other, esp. when making a turn</p> <p>bank <i>n</i></p> <p>1 a place where money is kept and paid out on demand, and where related activities go on.</p> <p>2 a place where something is held ready for use, esp. organic product of human origin for medical use</p> <p>3 (a person who keeps) a supply of money or pieces for payment or use in a game of chance</p> <p>4 break the bank to win all the money that the BANK 4(3) as in a game of chance</p>
--

Figure 4- 6: Homographs of “Bank” [16]

Then, the paper summarizes three categories, to one of which each LDOCE word type can be assigned on the level of homograph by part of speech. The three categories are as following:

#### 1. Guaranteed Disambiguation:

Each homograph that one word type contains is associated with different grammatical category. The word type will be disambiguated if its part-of-speech in a text is known.

e.g. a word with 3 homographs with grammatical categories adj, noun and verb.

#### 2. Possible Disambiguation:

There is at least one homograph associated only with one grammatical category. Other homographs can be associated with more than one grammatical category. e.g. a word with 3 homographs with grammatical category noun, noun, and verb. The word's sense will be disambiguated when the word's grammatical category in a text is verb.

### 3. No Disambiguation:

Each grammatical category that one word type contains has more than two or two homographs. This kind of word type is never fully disambiguated only by part-of-speech.

e.g. a word with 4 homographs with grammatical categories noun, noun, verb and verb.

Further, they examine each word type in LDOCE and find that 98.6% word types are guaranteed disambiguation and 99.4% are possible disambiguation over all word types.

Finally, they give a practical experiment to disambiguate the real text, which are five articles from *Wall Street Journal*, containing around 1700 words in total. The process of the disambiguation only uses part-of-speech tag. Brill tagger [17] is chosen as part-of-speech tags. Tags from Brill tagger are manually mapped to part-of-speech tags used in LDOCE. When tags suggest several homographs to one token in text, they decide to choose the first sense in the list. Because they think the first sense occurs most frequently and possibly is the best guess. The result shows that 92% of the content word tokens were tagged with the correct homograph compared with manual tagging of the same five articles.

### Bayesian Hierarchical Disambiguator (BHD)

The paper [6] presents a system called Bayesian Hierarchical Disambiguator (BHD) to disambiguate adjectives using probabilistic network. Firstly, the Equation (4.4) showed below is introduced.

$$\max_i \left( \frac{\Pr(\text{adj}, \text{noun}, \langle \text{NFs} \rangle | \text{adj}\#i) \times \Pr(\text{adj}\#i)}{\Pr(\text{adj}, \text{noun}, \langle \text{NFs} \rangle)} \right) \quad (4.4)$$

The term “adj#i” represents an adjective word with its  $i^{\text{th}}$  sense. We consider that the word's current sense is just the sense that makes value of the equation maximum. The term “ $\Pr(\text{adj}\#i)$ ” is called prior term, which represents how frequently a sense of an adjective word is used without any contextual information. But the likelihood term  $\Pr(\text{adj}, \text{noun}, \langle \text{NFs} \rangle | \text{adj}\#j)$  represents how frequently the sense of the adjective word is used with some contextual information, a part of which is here described with semantic features of the noun word in the pair (the adjective word, noun after it) with ISA hierarchy relationship of WordNet. For example, the noun-adjective pair “great hurricane”, we can see that there is the ISA hierarchy of hurricane in WordNet that

hurricane ISA cyclone ISA windstorm ISA violent storm....These nouns (cyclone, windstorm, violent storm....) are used as semantic features of “hurricane”. And these nouns and the noun “hurricane” compose of context information of the adjective “great”.

How to compute the prior term  $Pr(adj\#i)$  and the likelihood term  $Pr(adj,noun,<NFs> | adj\#j)$  ? For the prior term, the paper presents the method containing both automatic parts and manual parts. Firstly, 5000 nouns is collocated for each adjective, then a search engine is used to search these noun-adjective pairs for getting their occurring frequency. Then they are sorted and the top 100 nouns are fetched. Different senses are assigned to the adjective word in the top 100 noun-adjective pairs. For example, the top 10 nouns for “great” are “deal”, “site”, “job”, “place”, “time”, “way”, “American”, “page”, “book”, and “work”. They are all classified to the great #2 except the last one, as defined in WordNet. Therefore, the prior term is computed by dividing the mount of all the noun-adjective pairs with the occurring number of a sense. The likelihood term is computed with probabilistic network that is constructed with semantic features of nouns associated with an adjective.

The system BHD is also evaluated between with and without the prior terms and baselines that means putting the first sense in WordNet as the sense. The evaluation result can be seen in the Table 4-4 [6] below. “+SP” means the selectional preference model namely probabilistic network model considering the semantic features of nouns associated with an adjective as context. We can see in the column “1<sup>st</sup> noun sense” the accurate rate when considering the prior term is the highest and slightly higher than Baseline. Baseline’s result is better than the method without prior.

	Context	1 <sup>st</sup> noun sense	all noun senses
Without prior	Noun only	56.3%	53.3%
	+SP	60.0%	60.0%
With Prior	Noun only	77.8%	77.8%
	+SP	80.0%	81.4%
Baseline		75.6%	75.6%

Table 4- 4: Accuracy Comparison of Adjective Sense Determination [6]

To sum up, the system provides a probabilistic and statistic approach to solve the problem of word sense disambiguation and proves a good improved result with proper evaluation. These methods such as using search engines on Internet to collect data and analyzing probability are helpful for designing our system. However, the system BHD has to involve manual process.

## **A Knowledge-Driven Framework for WSD**

The paper [7] introduces a knowledge-driven framework for sense disambiguation of nouns. The idea behind the framework is that noun senses in a given context must be related through a certain relationship. The Framework contains the following five elements:

- i. a representation for senses, which is provided by the knowledge source,
- ii. a clustering algorithm for grouping related sense representations,
- iii. a match function for comparing a sense cluster with the textual context,
- iv. a filtering function for selecting sense clusters relying on the previous function,
- v. a stopping criterion for ensuring the termination of the disambiguation process.

### **4.3.3 Lexical Resources for Opinion Words**

In the paper [43] “Opinion Mining and Sentiment Analysis”, there is a list of lexicon about opinion words. We explore them and compare them for considering if it is possible to combine them into the thesis.

#### **OpinionFinder’s Subjectivity Lexicon**

The list of subjectivity clues is part of OpinionFinder and from several resources, can be downloaded on <http://www.cs.pitt.edu/mpqa/> in the part of Subjectivity clues, and is used in the paper [23]. The subjectivity clues contains sentiment words like “like”, “hate”, “beautiful”, including some subjectivity words “think”, “feel” as well. We can use them to judge subjective sentences or fetch some sentiment words.

#### **SentiWordNet**

SentiWordNet is lexical resource for opinion mining and can be acquired by the website <http://sentiwordnet.isti.cnr.it/>. Each synset of WordNet is assigned three scores: positive score, negative score and objective score. We can use the resource to judge if one word is sentiment word since we have the positive score or negative score or objective score from the resource. The sum of three scores for each synset is 1.0. If objective score of one word is 1.0, then it can be determined non-opinion word. But before that, we have to obtain the part of speech that the word plays in one sentence.

#### **General Inquirer**

<http://www.wjh.harvard.edu/~inquirer/>, the website provides some lists of positive or negative sentiment words or more detailed category like “pleasure”, “pain”. The data is dispersed and not completed compared with Opinion Finder’s subjectivity clues and SentiWordNet.



### **NTU Sentiment Dictionary**

The dictionary lists the polarities of many Chinese words. A user register form is used on <http://nlg18.csie.ntu.edu.tw:8080/opinion/userform.jsp>. Since the thesis only consider English song lyrics, then the dictionary is not useful here for us.

## **4.4 Song Lyrics, Songs Time Information and Song Genres**

In the part, we try to learn songs lyrics features and songs published time information and songs' musical types. Because the goal of the thesis requires us to analyze opinion mining result finally in terms of songs published time and songs musical type.

### **Song Lyrics and Title**

Lyrics aren't poetry. Lyrics have everything to do with music, but poetry has nothing to do with music and is a simple language game that has many word rules. It is more important for good lyrics to fit the music nicely and neatly than to have amazing metaphors or glamorous word combinations. It is extremely important that lyrics have an interesting topic, which means finishing almost half the job. Each song has title, which often contain topics of the song.

Good lyrics should be as simple as possible, no big words, no difficult-to-understand metaphors. This makes listeners to understand the music easily. As well, good lyrics should not go off topic.

Most of the songs include the title in the chorus, which is a general explanation of song topic. And usually the same word is not repeated more than two times (or Max. three) in chorus although the chorus itself is repeated.

### **Songs Time Information**

Here, songs time information can be the published time of songs, which will be used in results of opinion mining in the thesis. We see some change over sentiment or topics of songs in different years.

### **Songs Genre**

Often, we have song genres like rock, pop, country, folk, R&B, hip-hop and so on. Later, opinion mining result will be analyzed with song genres. Perhaps, it can present us some song genres contain more negative words than positive words, something like this.

## Chapter 5 Solution

The chapter presents how the opinion mining system for song lyrics is realized. It will describe architecture of the system and its components and the processing steps of the system like the process of part-of-speech tagging, extraction of opinion words, and so on.

### 5.1 System Architecture

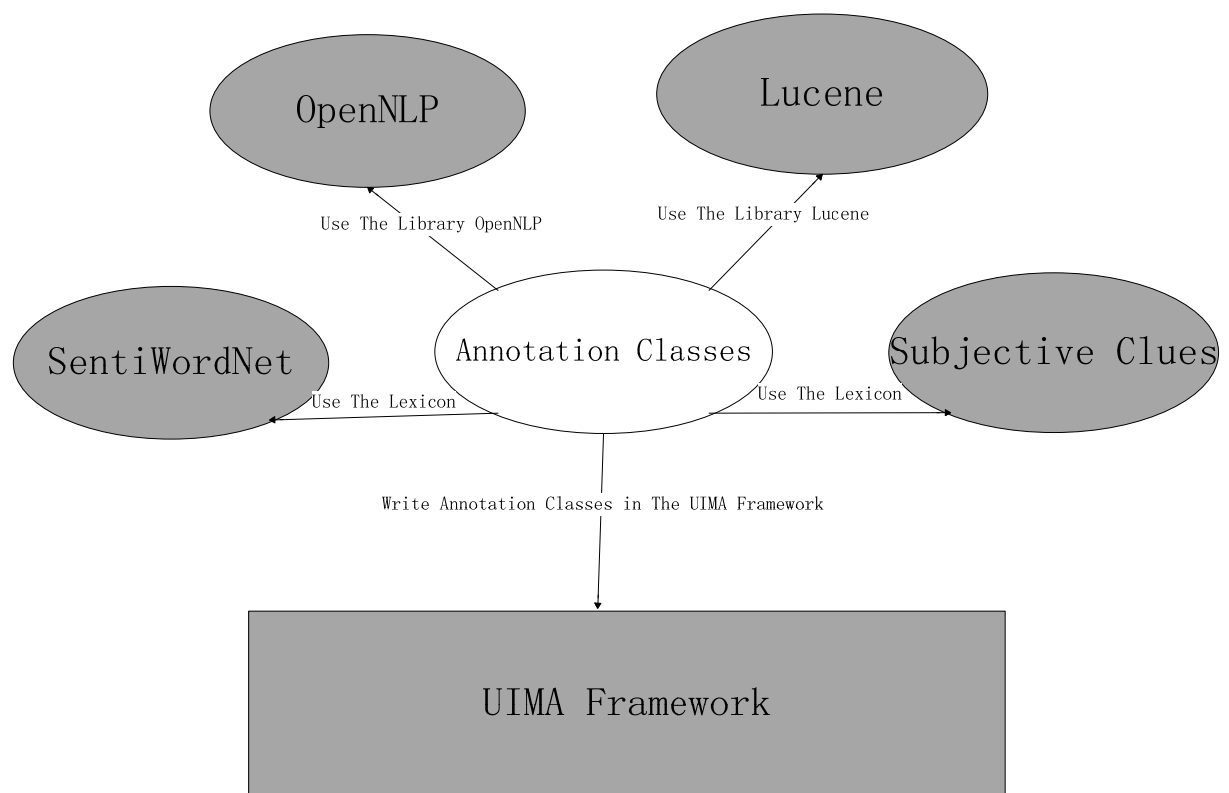


Figure 5- 1: System Architecture

As presented in the Figure 5-1 of system architecture, we write our annotation classes like the class of fetching keywords from song lyrics, or the class of fetching opinion words about these keywords in the framework of **UIMA**. These annotation classes use the library of **OpenNLP** for part-of-speech tagging of song lyrics, and use the library of **Lucene** for counting term document frequency in a collection of song lyrics. We use two lexicons **SentiWordNet** and **Subjective Clues** to fetch opinion words. There is the introduction of these components used in the system in the next section.

## 5.2 System components

The part will introduce system components appearing in the system architecture above.

### 5.2.1 Annotation Classes

The part is the key part in the system, and we write our own annotation classes here. The annotation classes include the classes of fetching objects of interest and fetching opinion words. In the exploration of methods of fetching objects of interest mentioned in the last chapter, we have written these annotation classes of methods of fetching objects of interest (keywords) in the component as showed in the Figure 5-2 below.

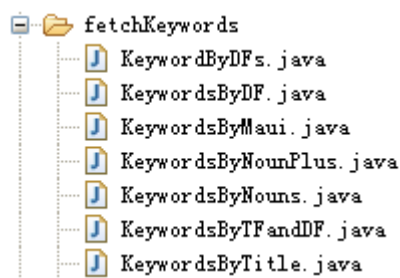


Figure 5- 2: Java Classes of Fetching Keywords

KeywordsByDF.java fetches keywords by document frequency;  
KeywordsByMaui.java fetches keywords by Maui Topic Indexing Algorithm; (in the final source code, the file name is changed).

KeywordsByNouns.java fetches keywords by ranking nouns;

KeywordsByTFandDF.java fetches keywords by multiplying term frequency and document frequency;

KeywordsByTitle.java fetches keywords by song title information.

The component of annotation classes actually belong to the framework UIMA as introduced next. The code example of annotation classes is showed in Appendix C: KeywordsByNouns.java. Annotation classes can help us mark special annotation in text as showed in the Figure 5-3 below. It marks some token types like Keywords and Sentiment Words.

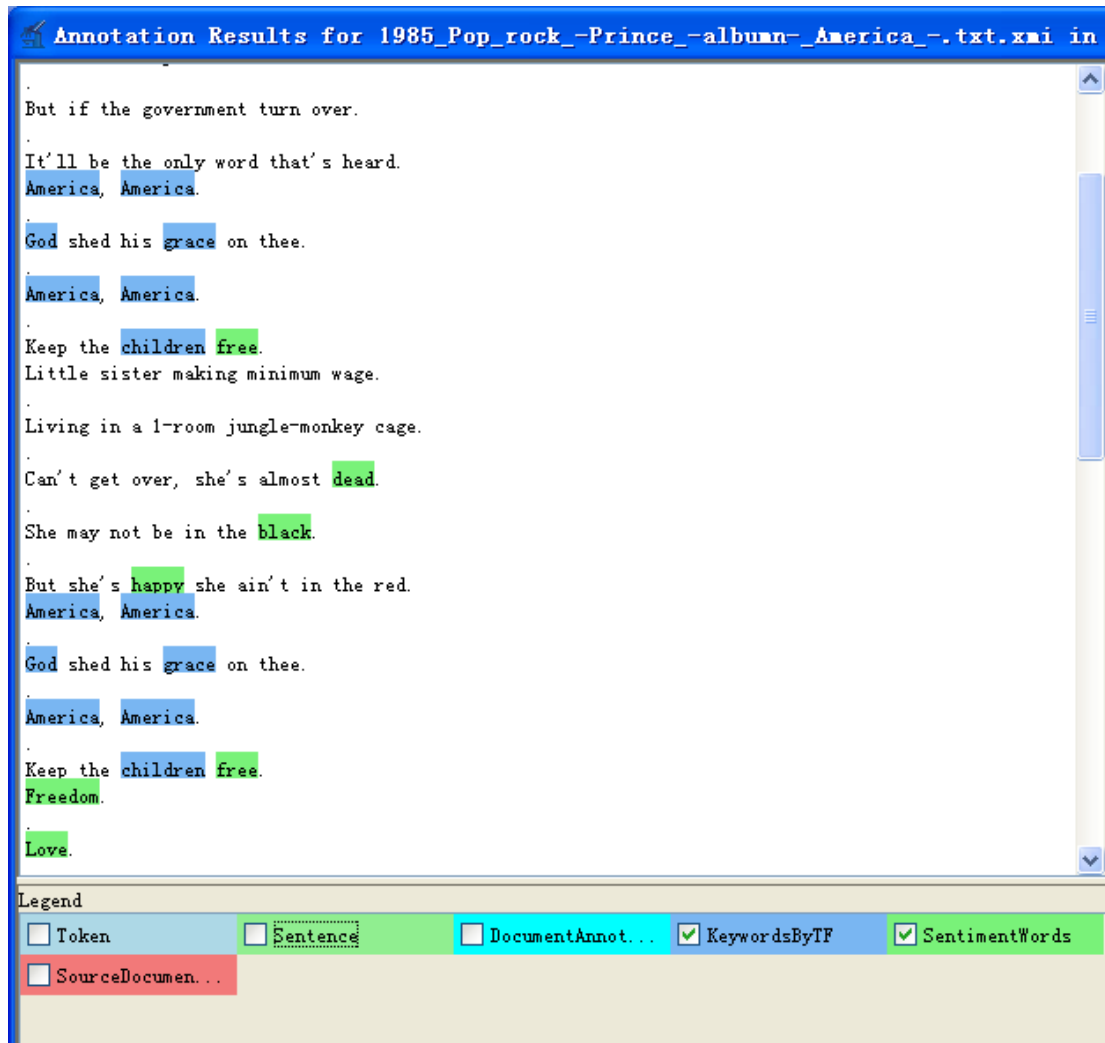


Figure 5- 3: Annotation in Text with UIMA

## 5.2.2 UIMA

Now, we introduce UIMA, the framework adopted in the thesis.

### What is UIMA?

UIMA (Unstructured Information Management Architecture) [42] is the framework that is used to analyze massive unstructured information for marking some special kinds of information like adjectives, nouns, adverbs, and verbs in a text.

In UIMA, Developers can write their own annotator, which is the component doing actual analyzing work of unstructured information, e.g. the annotator of language identification, the annotator of fetching topics from one document, the annotator of fetching opinion words. As well, we can run a pipeline of several annotators together. The framework is available for both Java and C++ now.

### 5.2.3 OpenNLP

OpenNLP is used for part of speech tagging. OpenNLP (<http://opennlp.sourceforge.net/index.html>) is an organizational center for open source projects related to natural language processing (NLP). In the thesis, we integrate OpenNLP into the UIMA, and how to integrate the both can be seen in Appendix A. OpenNLP contains various java-based NLP tools such as sentence detection, tokenization, part-of-speech tagging (POST), chunking and parsing.

#### Sentence Detection

Sentence Detection is used for separating sentences. For example, if song lyrics are input like below:

*Don't you understand, what I'm trying to say?  
Can't you see the fear that I'm feeling today?  
If the button is pushed, there's no running away,  
There'll be noone to save with the world in a grave,  
take a look around you, boy, it's bound to scare you, boy,  
but you tell me over and over and over again my friend,  
ah, you don't believe we're on the eve of destruction.*

Then, the output after the process of Sentence Detection is: (each color stands for one sentence.)

*Don't you understand, what I'm trying to say?  
Can't you see the fear that I'm feeling today?  
If the button is pushed, there's no running away,  
There'll be noone to save with the world in a grave,  
take a look around you, boy, it's bound to scare you, boy,  
but you tell me over and over and over again my friend,  
ah, you don't believe we're on the eve of destruction.*

#### Tokenization

The process of tokenization is used for breaking down a sentence into tokens, which are separated by spaces. For example, if the input is:

*Don't you understand, what I'm trying to say?*

Then, we can see that the output is:

*Do n't you understand , what I 'm trying to say ?*

We can see that “don’t” is split into “do” and “n’t”, and punctuations like “,” and “?” are also split into separate tokens. Usually, tokens should be words.

## Part-of-Speech Tagging (POST)

The part-of-speech tagging is used for tagging each token as verb, or adverb, or adjective, or.....The tags conform to “Pen Treebank Style”, the list of which at the word level is showed below:

<b>CC</b> - Coordinating conjunction	<b>CD</b> - Cardinal number
<b>DT</b> – Determiner	<b>EX</b> - Existential there
<b>FW</b> - Foreign word	
<b>IN</b> - Preposition or subordinating conjunction	
<b>JJ</b> – Adjective	<b>JJR</b> - Adjective, comparative
<b>JJS</b> - Adjective, superlative	<b>LS</b> - List item marker
<b>MD</b> – Modal	<b>NN</b> - Noun, singular or mass
<b>NNS</b> - Noun, plural	<b>NNP</b> - Proper noun, singular
<b>NNPS</b> - Proper noun, plural	<b>PDT</b> - Predeterminer
<b>POS</b> - Possessive ending	<b>PRP</b> - Personal pronoun
<b>PRP\$</b> - Possessive pronoun (prolog version PRP-S)	<b>RB</b> - Adverb
<b>RBR</b> - Adverb, comparative	<b>RBS</b> - Adverb, superlative
<b>RP</b> – Particle	<b>SYM</b> - Symbol
<b>TO</b> – to	<b>UH</b> - Interjection
<b>VB</b> - Verb, base form	<b>VBD</b> - Verb, past tense
<b>VBG</b> - Verb, gerund or present participle	<b>VBN</b> - Verb, past participle
<b>VBP</b> - Verb, non-3rd person singular present	
<b>VBZ</b> - Verb, 3rd person singular present	
<b>WDT</b> - Wh-determiner	<b>WP</b> - Wh-pronoun
<b>WP\$</b> - Possessive wh-pronoun (prolog version WP-S)	
<b>WRB</b> - Wh-adverb	

For example, if the input is:

*We love our mother nations.*

Then the output is: (symbols after slash stand for part-of-speech tags assigned)

*We/PRP love/VBP our/PRP\$ mother/NN nations/NNS ./.*

### 5.2.4 Lucene

**Apache Lucene** is an open source information retrieval software library containing text indexing and searching, originally created in Java by Doug Cutting. It is supported by the **Apache Software Foundation** and is released under the **Apache Software License**. Lucene has been ported to other programming languages including Delphi, Perl, C#, C++, Python, Ruby and PHP.

Lucene was initially available for downloading from Doug Cutting’s home at the SourceForge web site. It joined the Apache Software Foundation’s Jakarta family of high-quality open source Java products in September 2001 and became its own top-level Apache project in February 2005.

Lucene has been widely recognized for its utility in the implementation of Internet search engines and local, single-site searching. And it can index various file formats' documents like PDF, TXT, MS WORD, HTML as long as text can be extracted from documents.

### **Indexing with Lucene**

There are some fundamental Lucene classes for indexing text like **IndexWriter**, **Analyzer**, **Document**, and **Field**.

**IndexWriter** is used to create a new index and to add Documents to an existing index.

**Analyzer** is responsible for extracting indexable tokens out of text to be indexed, and eliminating the rest. Lucene contains a few different Analyzer implementations. Some of them are used for removing stopwords (frequently-used words that don't help distinguish one document from the other, such as "a," "an," "the," "in," "on," etc.), some help convert all tokens to lowercase letters so that searching is not case-sensitive, and so on.

An index consists of a set of Documents, and each **Document** consists of one or more **Fields**. Each Field has a name and a value. Think of a Document as a row, and Fields as columns in that row.

There is a code example of index showed below:

```
// text is the text to index with Lucene
String text = "This is the text to index with Lucene";
// indexDir is the directory that hosts Lucene's index files
File indexDir = new File("F:/Master Thesis/luceneIndex");
// analyzer is used for pre-processing the text;
Analyzer analyzer = new StandardAnalyzer();
IndexWriter writer= new IndexWriter(indexDir, analyzer, true);
Document document = new Document();
document.add(Field.Text("fieldname", text));
writer.addDocument(document);
writer.close();
```

### **Reading Index**

We can use the class **IndexReader** to read out index data from existing index, such as getting document frequency of a term in a collection of documents.

### 5.2.5 SentiWordNet

SentiWordNet [15] is built on WordNet [18] and assigns three numerical scores  $Obj(s)$ ,  $Pos(s)$  and  $Neg(s)$  to each WordNet synset. These three scores describe respectively how objective, positive and negative words contained in the synset are. There is a visualization representation given in SentiWordNet as showed in the Figure 5-4 below. We can see the figure shows three polarities of Positive, Negative and Objective. It describes the change between Positive and Negative on horizontal direction and the change from Subjective to Objective. The ball stands for a sense of one word. The position where it is represents the sentiment orientation the word sense has.

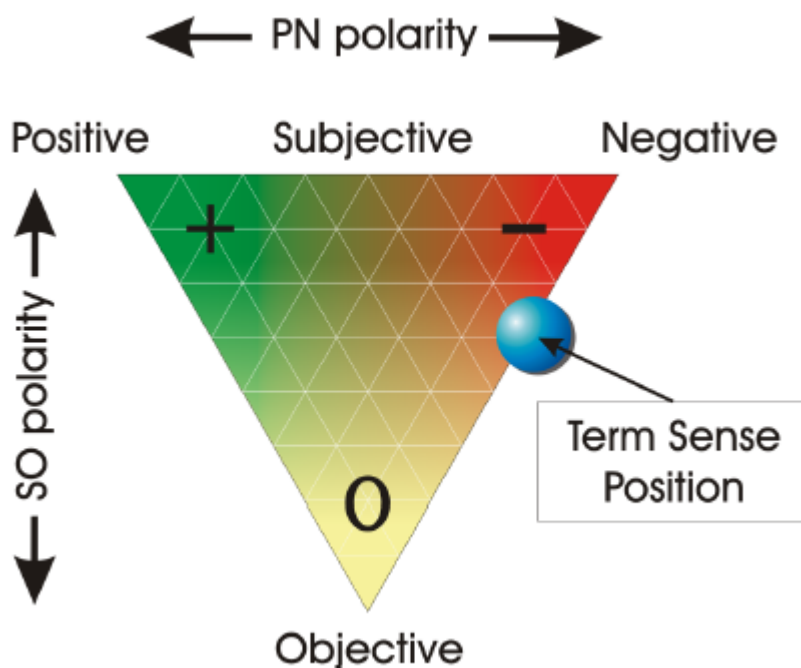


Figure 5- 4: Visualization Representation of Term Polarity of SentiWordNet [15]

We can search one word's different senses' scores of Pos, Neg, Obj on <http://sentiwordnet.isti.cnr.it/>. For example, if we input the word “ugly”, it shows the result like the Figure 5-5 below. We can see positive score is 0, objective score is 0.625, and the negative score is 0.375 when “ugly” is with the first sense (#1).



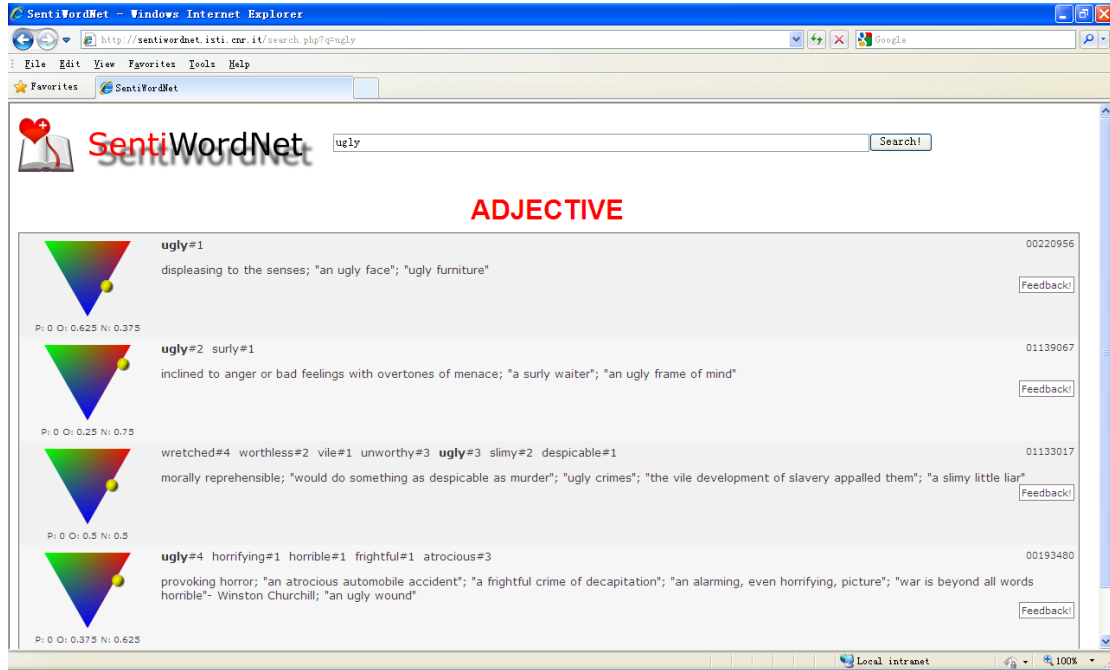


Figure 5- 5: SentiWordNet Online

One java class is given in Appendix C: SWN3.java, which shows an example of how to reads data from the lexicon resource.

## 5.2.6 Subjective Clues

We also integrate a list of subjective clues, which is used in [23]. These clues are collected from some sources. A majority of the clues are collected in the work reported in [24], some are from manually developed resources, and others are identified automatically using both annotated and not-annotated data. The list of subjective clues is organized in a file with some formats, for example, each line presents one subjective clue like below:

*type=weaksubj len=1 word1=accept pos1=verb stemmed1=y priorpolarity=positive*

There is the explanation of each attribute as the following:

**type** - either strongsubj or weaksubj

A clue that is subjective in most contexts is considered strongly subjective (strongsubj), and those that may only have certain subjective usages are considered weakly subjective (weaksubj).

**len** - length of the clue in words

All clues in this file are single words.

**word1** - token or stem of the clue

**pos1** - part of speech of the clue, may be anypos (any part of speech)

**stemmed1** - y (yes) or n (no)

Is the clue word1 stemmed? If stemmed1=y, this means that the clue should match all unstemmed variants of the word with the corresponding part of speech. For example, "abuse" with part-of-speech tag "verb", will match "abuses" (verb), "abused" (verb), "abusing" (verb), but not "abuse" (noun) or "abuses" (noun).

**priorpolarity** - positive, negative, both, neutral

The prior polarity of the clue means the polarity of the clue when without context.

Code example of Reading the lexicon is attached Appendix C: SubLexicon.java.

## 5.3 Processing Steps of System

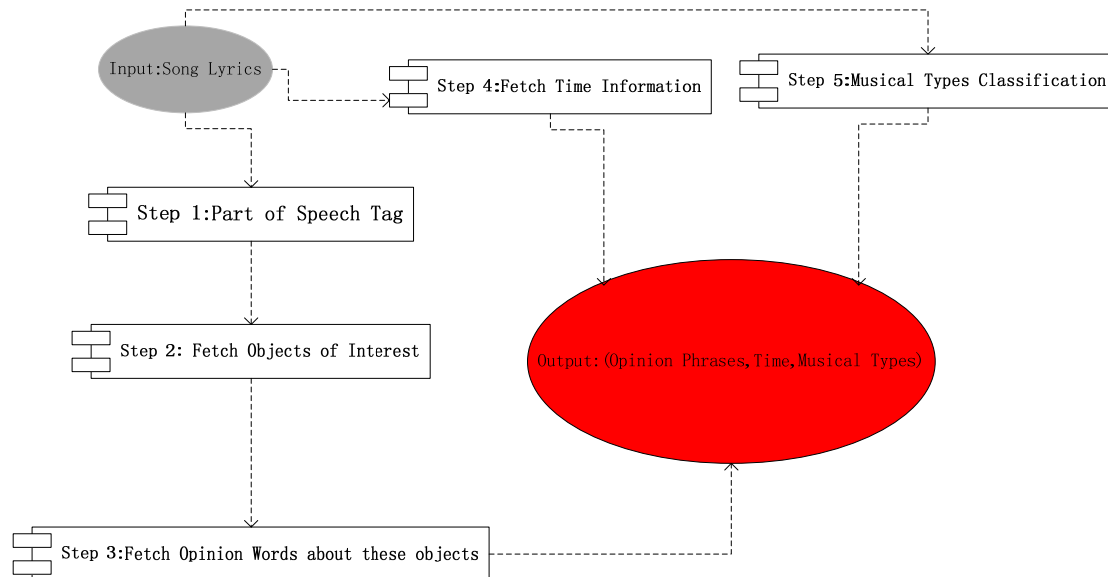


Figure 5- 6: Processing Steps of System

The Figure 5-6 above describes processing steps of the system we build. The oval in grey means the input of the system that is song lyrics. The oval in red is the output of the system that should be opinion phrases of song lyrics with songs' published time and their musical types such as rock, classical, pop. The opinion phrases mean the pairs containing opinion words and objects of interest, for example, "crazy world", "Great man".

The processing steps of the system are described as following:

### **Input: Song Lyrics.**

**Step 1: Do Part-of-Speech Tagging.** It is used for tagging each token or word in song lyrics as noun, or verb, or adjective, under help of the component of OpenNLP.

### **Step 2: Fetch Objects of interest.**

We fetch nouns with high frequency (three most high frequency nouns, but sometimes it maybe is more or less than three because maybe there is the situation of equal frequency or no enough nouns in song lyrics), and combine with title words that does not appear in the nouns fetched with high frequency before. Title words should be filtered with some articles (a, an, etc.), conjunctions (and, or, but), prepositions (in, on, etc.), be verbs (was, is, etc.) and others. We list these stop words in Appendix D.

### **Step 3: Fetch opinion words about these objects.**

Two components **SentiWordNet** and **Subjectivity Clue** are used in the process of fetching opinion words. Final opinion words are the result after filtered by the two components.

#### **The process of Fetching Opinion Words with SentiWordNet**

We firstly choose potential opinion words in the process of Fetching Opinion Words with SentiWordNet. The adjectives or adverbs close to the high frequent nouns (the distance can be 2 tokens before or after the nouns) are considered as potential opinion words. Further, if the potential opinion words are not sentiment words, i.e. neither negative nor positive, they should be removed. The remaining words are kept as opinion words.

We consider that one word is opinion word if the word has sentiment polarity rather than objective. In order to judge if the word has sentiment polarity, we firstly do the word's sense disambiguation as the following:

One word in WordNet can have several different senses; different senses correspond to different numerical score of Neg, Pos, Obj. So we have to consider how to disambiguate word senses. We decide to process with two steps:

Step 1: Through part-of-speech tagging, the system find the grammatical category of one word in text, this can filter other senses of the word with other grammatical categories.

Step 2: we choose the first sense under the right grammatical category since senses in WordNet are generally ordered from most to least frequently used, with the most common sense numbered 1. Frequency of use is determined by the number of times a sense is tagged in the various semantic concordance texts. Senses that are not semantically tagged follow the ordered senses. This is introduced in WordNet Online [18]. We also can see the first sense is applied in several papers [6, 19] as baseline system.

After the process of Word Sense Disambiguation, we have the positive and negative and Objective scores of one potential opinion word under one sense from SentiWordNet. If its objective score is 1, then we consider the word does not have sentimental polarity, and should be removed from the list of opinion words. The remaining opinion words are chosen as opinion words. And if positive score is more than negative score, we judge the word in the context to be with positive sentiment; otherwise, it is negative sentiment.

#### **The process of fetching opinion words with Subjectivity Clues**

Next, we will test each token with Subjective Clues in the same sentence as keywords. If one token with the corresponding part-of-speech tag appears in the list of subjective clues and the token's polarity is not neutral (Subjective Clues provide polarity of each word, which we can use directly), we fetch the token as opinion words of the

keywords.

**Step 4: Fetch published time of songs.** How to fetch songs published time depends on how the published time is given when input. In the thesis, the published time is given in the file name of each song lyrics.

**Step 5: Fetch types of songs.** How to fetch songs genres also depends on how songs genre is given when input. In the thesis, song genre is given in the file name of each song lyrics.

**Output:** The system outputs the result in the form of (objects of interest and their opinion words, time, musical type). For example, (crazy word, 1976, rock).

## Chapter 6 Experiments and Evaluation

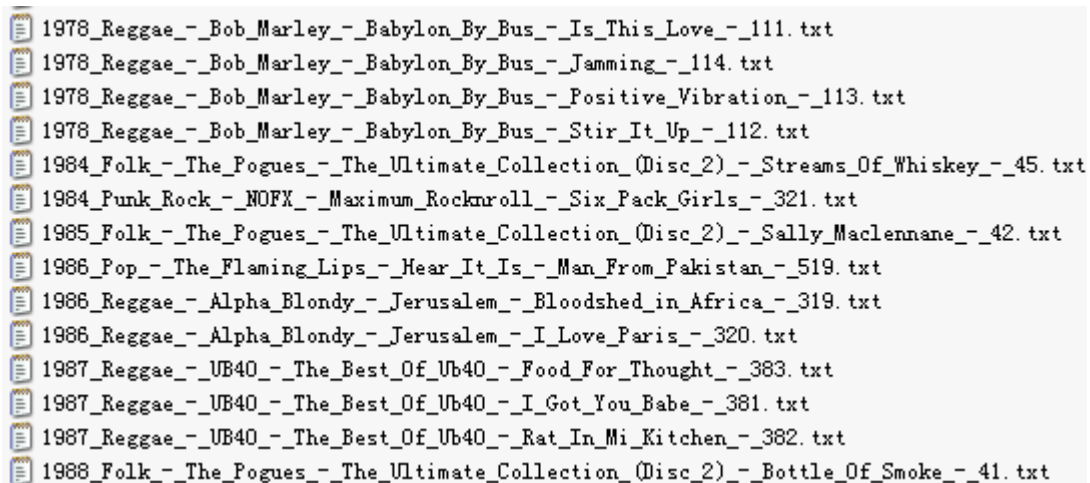
The chapter presents some experiments on the opinion mining systems for song lyrics and their results and the results analysis.

### 6.1 Data Preparation

We have two parts of song lyrics for experiment; one part is the collection of Song Lyrics in hand containing 506 lyrics; the other part is the collection that we collect on Internet on our own.

#### Collection of Song Lyrics in Hand

We have one collection of song lyrics in hand in .txt file format as showed below in the Figure 6-1, each file's name of which contains one song's published time, musical types, artist, album name, and song title in the order from left to right, e.g. in the first file, the published time is "1978", the song genre is "Reggae", the artist is "Bob Marley", the album is "Babylon By Bus", the song title is "Is This Love". As well, we show some examples of song lyrics in Appendix B.



```
1978_Reggae_-_Bob_Marley_-_Babylon_By_Bus_-_Is_This_Love_-_111.txt
1978_Reggae_-_Bob_Marley_-_Babylon_By_Bus_-_Jamming_-_114.txt
1978_Reggae_-_Bob_Marley_-_Babylon_By_Bus_-_Positive_Vibration_-_113.txt
1978_Reggae_-_Bob_Marley_-_Babylon_By_Bus_-_Stir_It_Up_-_112.txt
1984_Folk_-_The_Pogues_-_The_Ultimate_Collection_(Disc_2)_-_Streams_Of_Whiskey_-_45.txt
1984_Punk_Rock_-_NOFX_-_Maximum_Rocknroll_-_Six_Pack_Girls_-_321.txt
1985_Folk_-_The_Pogues_-_The_Ultimate_Collection_(Disc_2)_-_Sally_Maclennane_-_42.txt
1986_Pop_-_The_Flaming_Lips_-_Hear_It_Is_-_Man_From_Pakistan_-_519.txt
1986_Reggae_-_Alpha_Blondy_-_Jerusalem_-_Bloodshed_in_Africa_-_319.txt
1986_Reggae_-_Alpha_Blondy_-_Jerusalem_-_I_Love_Paris_-_320.txt
1987_Reggae_-_UB40_-_The_Best_Of_UB40_-_Food_For_Thought_-_383.txt
1987_Reggae_-_UB40_-_The_Best_Of_UB40_-_I_Got_You_Babe_-_381.txt
1987_Reggae_-_UB40_-_The_Best_Of_UB40_-_Rat_In_Mi_Kitchen_-_382.txt
1988_Folk_-_The_Pogues_-_The_Ultimate_Collection_(Disc_2)_-_Bottle_Of_Smoke_-_41.txt
```

Figure 6- 1: Collection of Song Lyrics

Besides, we need to do some processing with these data. We add period at the end of each line of song lyrics if there is no period, since we analyze song lyrics sentence by sentence as mentioned in the section 5.2. Actually, it is also necessary if we want to fetch exactly the sentiment words close to nouns. It can make sense only if sentiment words fetched are the words that appear in the same sentence with the relative nouns.

At the same time, we search songs' information (e.g. published time) on Internet if missing the information in files of songs' lyrics.

## Collecting Song Lyrics on Internet

As well, we collect song lyrics on the website <http://www.absolutelyrics.com/> and save each of them in a .txt file with file name containing artist and song title. In the process, we collect 20,067 song lyrics. Then we use the artist and song title information to search on Wikipedia.org for the information of year and genre and so on. Because, luckily, we find that there is information box in the introduction of song on Wikipedia.org as showed in the Figure 6-2. In the process, we run some of 20,067 song lyrics obtained in the last process and obtain 1,102 song lyrics with their published years at least. Both processes mentioned are programmed and done automatically by computers. We import the library form Cobra Tool Kit on <http://lobobrowser.org/cobra.jsp> to parse HTML file. There are two java classes attached in Appendix C: SearchWikipedia.java that is used for searching information about year or genre or others on Wikipedia.org with the keywords input of song artist and song title, and FetchSongLyrics.java that is used to fetch song lyrics from the website <http://www.absolutelyrics.com/>, and save each song lyrics into separate .txt file. Through the method, we can collect thousands of song lyrics very easily, but not for all the song lyrics, the information of their year or genre or others can be found on Wikipedia. Moreover, genre information found maybe contains several genres name that actually is the genre of the whole album that one song belongs to. Finally, we still need to process the collection like above for separating sentences.

**"Hey Jude"**



**Single by The Beatles**

<b>B-side</b>	"Revolution"
<b>Released</b>	26 August 1968
<b>Format</b>	7"
<b>Recorded</b>	31 July 1968 at Trident Studios, London
<b>Genre</b>	Rock
<b>Length</b>	7:20 (mono) 7:11 (stereo)
<b>Label</b>	Apple
<b>Writer(s)</b>	Lennon/McCartney <sup>[1]</sup>
<b>Producer</b>	George Martin
<b>Certification</b>	4x Platinum (RIAA)

Figure 6- 2: Information Box of Song on Wikipedia

## 6.2 Experiment and Evaluation

### Experiment 1: Fetch Topics

We fetch topics with the method of ranking nouns frequency in song lyrics, combining with the title information. We run the experiment in the collection of song lyrics containing 506 song lyrics prepared in the last section Data Preparation.

We randomly choose 10 song lyrics (the same as the experiment of exploring the methods of fetch keywords done in Chapter 4). We compare the machine result with manual Identification to see how accurate the system fetches topics. The Table 6-1 and the Table 6-2 below presents the result.

We can see that topics of 7 songs lyrics are fetched correctly totally, topics of 2 songs lyrics are fetched correctly to some extent, topics of only one song lyrics is not fetched correctly. We analyze the song that is not fetched topics correctly and find that creator of the song lyrics does not express the meaning of the song very straightly. It leads to the situation that the topics words do not appear very frequently.

	<b>Song Information</b>	<b>Manual Identification</b>	<b>Machine result</b>	<b>Correct?</b>
1	1978, Reggae, Bob Marley, Babylon By Bus, <b>Is This Love</b>	love	love, Shelter, Bed	Yes
2	1984, Folk, The Pogues, The Ultimate Collection, <b>Stream of Whiskey</b>	Life philosophies	Where, Streams, Whiskey	No
3	1989, Grunge, Nirvana, Bleach, <b>Negative Creep</b>	Daddy's girl	Negative, creep, Daddy, girl	Yes
4	1992, Grunge, Mudhoney Superfuzz Bigmuff Plus Early, <b>Chain that door</b>	Chain that door, girl	Chain, door, loser	Yes

Table 6- 1: Evaluation of Fetching Song Topics-A



	<b>Song Information</b>	<b>Manual Identification</b>	<b>Machine Result</b>	<b>Correct?</b>
5	1992, Grunge, Alice in Chains, Dirt, <b>Angry Chair</b>	I don't mind	Angry, chair, Boy, Pink, cloud, knees, Time, Mind	Correct to some extent
6	1998 Metal System of A Down System of A Down <b>Suite-pee</b>	Jesus Philosophy	Philosophy, Motherfucker, Christ Suite-pee	Correct to some extent
7	1999 Country Wilco Summerteeth <b>In a future age</b>	In a future age	Future, age, page	Yes
8	2002 Country Dixie Chicks Home <b>White Trash Wedding</b>	Wedding Ring	Ring, hand, white, mamma, baby,way, Trash Wedding	Yes
9	2002 Slow Rock Coldplay A Rush of Blood to the head <b>A Whisper</b>	whisper	Whisper, sound, ticking, clocks	Yes
10	2003 Pop Britney Spears In the zone <b>Shadow</b>	shadow	Shadow, nobody, room	Yes

Table 6- 2: Evaluation of Fetching Song Topics-B

## Experiment 2: Fetch Opinion Words

We run the experiment based on the last experiment, and try to find opinion words that describe topics fetched in the last experiment. Later, we will give evaluation of the experiment results and see if the opinion words are fetched properly. We also use the same 10 songs lyrics as before and list the Table 6-3 and the Table 6-4 about the result. In the last column, we present the format like (topic, opinion words...). One topic is with some opinion words describing the topic.

	<b>Song Information</b>	<b>Manual Identification of topics</b>	<b>Machine result of topics</b>	<b>(topic, opinion words)</b>
1	1978, Reggae, Bob Marley, Babylon By Bus, <b>Is This Love</b>	love	love, Shelter, bed	(bed, shelter)
2	1984, Folk, The Pogues, The Ultimate Collection, <b>Stream of Whiskey</b>	Life philosophies	Where, Streams, Whiskey	No opinion words for topics
3	1989, Grunge, Nirvana, Bleach, <b>Negative Creep</b>	Daddy's girl	Negative, creep, Daddy, Girl	(girl, little) (creep, negative)
4	1992, Grunge, Mudhoney Superfuzz Bigmuff Plus Early, <b>Chain that door</b>	Chain that door, girl	Chain, door, loser	No opinion words for topics
5	1992, Grunge, Alice in Chains, Dirt, <b>Angry Chair</b>	I don't mind.	Angry,chair, Boy, Pink, cloud, knees, Time, Mind	(chair, angry) (boy, mistake, dull) (time, pray)

Table 6- 3: Evaluation of Fetching Opinion Words-A

	<b>Song Information</b>	<b>Manual Identification of Topics</b>	<b>Machine Result of Topics</b>	<b>(topics, opinion words)</b>
6	1998 Metal System of A Down System of A Down <b>Suite-pee</b>	Jesus Philosophy	Philosophy, Motherfucker, Christ, Suite-pee	(philosophy, try)
7	1999 Country Wilco Summerteeth <b>In a future age</b>	In a future age	Future, age, page	No opinion words for topics
8	2002 Country Dixie Chicks Home <b>White Trash Wedding</b>	Wedding Ring	Ring, hand, white, mamma, baby, way, wedding, trash	(ring, afford) (mamma, approve)
9	2002 Slow Rock Coldplay A Rush of Blood to the head <b>A Whisper</b>	whisper	Whisper, sound, ticking, clocks	No opinion words for topics
10	2003 Pop Britney Spears In the zone <b>Shadow</b>	shadow	Shadow, nobody, room	(shadow, never)

Table 6- 4: Evaluation of Fetching Opinion Words-B

We can see that 6 of 10 songs lyrics present opinion words about topics or some of topics. At least, we can know some about what songs lyrics express through these phrases (topics, opinion words). Obviously, it is better like this than only topic words. We also can say that opinion words are fetched adequately. Only in four songs lyrics, no opinion words are fetched. Some of them maybe just do not have opinion words. Of course, we also find some problem. In number 8 song “2002, Country, Dixie Chicks, Home, White Trash Wedding”, opinion words of the topic “mamma” is only given “approve”, in fact, it is “do not approve”. Therefore, the problem that how to fetch the negative elements like the words “does not” arises.

### **Experiment 3: Fetch Common Topics**

We run the experiment on the same collection of song lyrics as before, we use the same method as Experiment 1 to fetch topics, and then ranking document frequency of all the topics appeared and fetch about top 10 topics. We consider the top topics are the most common.

Experiment results presents us the most common, in which we can see that there are topics like “love”, “life”, “world” and so on. We can imagine that these topics will never change. People always talk about their love, their lives and the world no matter when and no matter where and no matter how the world changes.

At the same time, we also find that some interesting things, in the topic “world”, the concept “green world” appears in the song *O Green World* (published time: 2005, musical type: pop, singer: Gorillaz, album: Demon Days). It matches the fact that the concept attracts the attention of more and more people in recent years. We all should save our earth.

As well, we find that the sentiment words like “suicide” “pressure” starts to appear in the topic “life” in the song of Imn (published time: 2005, musical type: Metal, singer: Mudvayne, album: Lost And Found). In the songs before 2005, we don’t find these sentiment words. It matched the fact that the world we meet is more and more competitive, which leads to more pressure people have to meet.

#### **Baby, Time, Way, Mind**

Unfortunately, in these common topics fetched, there are some words like “baby”, “time”, “way”, and “mind”. We consider that these words often do not make sense to the meaning that songs express although they appear very frequently. Perhaps, some of them often appear in chorus of songs.

## Experiment 4: Comparison between Different Musical Types

We run the experiment on the same collection of song lyrics as the last experiments. Songs' topics and opinion words about them are fetched. We divide opinion words into two polarities: Positive and Negative. And we use both two lexicons of Subjective Clue and SentiWordNet to judge one word's polarity. Subjective Clue provides us subjective vocabularies with polarity. In SentiWordNet, we consider if positive score of one word is more than negative score of one word, the word is judged positive, otherwise, it is negative.

Finally, the experiment presents us the result as the following:

```
Metal:  
negative:328 positive:326  
Pop:  
negative:278 positive:126  
Folk:  
negative:590 positive:337  
R&B:  
negative:1103 positive:433  
Reggae:  
negative:380 positive:353  
Grunge:  
negative:782 positive:152  
Slow Rock:  
negative:331 positive:99  
Country:  
negative:177 positive:249  
Hip-hop:  
negative:236 positive:328  
Punk:  
negative:953 positive:417
```

It is interesting to see that negative words are more than positive words in some musical types like Grunge, Punk, Slow-Rock, and R&B. We can understand that rock music (e.g. mentioned above like Grunge, Punk, and Slow-Rock) should contain more negative words than positive words. As well, we see in the result that Hip-hop appears more positive words than negative words, which tells us that Hip-hop is often up. Country music also shows more positive words. Reggae has almost the same positive words and negative words, and Metal is also.

## Experiment 5: Comparison between Different Years

We run Experiment 5 on 20 song lyrics with the title containing America or American. And we can imagine that these songs maybe is about American or America. Indeed, the experiment result presents us one common topic “American” with opinion words by different year as showed in the Table 6-5 and the Table 6-6 below. In the first column, it is song published year; it shows song’s title and its artist and perhaps includes song genre in the second column, and the last column is opinion words with part-of-speech tags. It is ordered by year increasing from up to down. Interestingly, we can find that there is one opinion word “war” in the song published on 1970, when it is the period of cold war. After 1990s, the songs do not appear it, but some opinion words like ‘love’, ‘frustration’, ‘dream’ and so on.

	<b>Songs’ Pulished Year</b>	<b>Song Title (perhaps including song genre), Artist</b>	<b>Opinion words with part-of-speech tag</b>
1	1970,	American Woman (Hard rock), The Guess Who	important#adj, war#noun, ghetto#noun
2	1973	American Tune, Simon and Garfunkel	uncertain#adj
3	1977	American Girl (Rock), Goo Goo Dolls	help#verb, great#adj
4	1993	American Honky(Country), Garth Brooks	try#verb, rectify#verb, mind#verb, welfare#noun, great#adj, love#verb, frustration#noun
5	2000	American_Psycho, Treble_Charger	[freak#noun]
6	2002	American Girls (Rock), Counting Crows	fall#verb, wish#verb, right#adj, well#adverb, cry#verb, please#verb, try#verb
7	2003	American Tune, Eva Cassidy	uncertain#adj, right#adj
8	2003	American Life (pop), Madonna	mess#noun, wrong#adj, sympathy#noun, little#adj, friend#noun, dream#noun
9	2003	American Child (Country), Phil Vassar	dirt#noun, love#verb, thank#verb, dreams#noun, wild#adj

Table 6- 5: Opinion Change over Common Topic “American”-A

10	2003	American Soldier (Country), Toby Keith	liberty#noun, jeopardy#noun, right#adj
11	2007	American X (Rock), Black Rebel Motorcycle Club	bliss#noun, cut#verb
12	2010	American Honey (Country), Lady Antebellum	strong#adj, love#noun, slow#adj, weed#noun, ready#adj, innocent#adj, pure#adj, sweet#adj, wild#adj, friend#noun, miss#verb, lose#verb
13	2010	American Saturday Night (Country), Brad Paisley	kiss#noun

Table 6- 6: Opinion Change over Common Topic “American”-B

## Experiment 6: Hot Topics in Different Year

We run the experiment in the collection of about 1,600 song lyrics with the published year at least. Hot topics means these topics are often as the topics of song lyrics published in one year. We list the result in the Table 6-7 and Table 6-8 below in the order from now to past.

We can find that the most popular topics are still “love”, “life”, “world”, and “girl” and so on. It is interesting to see the topic “Chance” and “Peace” in 1969; “One Love” in 1976; “One World” in 1982; “Fighting” in 1974; “Merry Christmas” in 1944. The word “Time” occur very frequently but it is considered non-hot topics, because the word is really too common. The result depends on very much chosen song lyrics running the experiment. Larger amount of song lyrics can show us more significant result.

Year	Hot Topics
2010	Time
2009	Time, Head
2008	Time
2007	Time
2006	Time
2005	Time, Love, Life
2004	Time
2003	Time, Love
2002	Heart, Love
2001	Time, Love
2000	Time
1999	Everything
1998	Time, Nothing, Heart
1997	Soul, Heart
1996	Life, Love
1995	Heaven, World, Love, Light, Girl
1994	Something, Life, Love
1993	Day, Love
1992	Baby, Door, Love
1991	Time, Man
1990	Eyes, Honey, Bridge
1989	Love, Girl
1988	Time, Heart
1987	People, Race, Love, Night
1986	Love, Heartbeat, Nation, Daddy, Adventure, Girl, Kingdom, Angels, Dreamers

Table 6- 7: Hot Topics-A



<b>Year</b>	<b>Hot Topics</b>
1985	Love, Words
1984	Year, Christmas
1983	Fun, Girls
1982	One, World
1981	Night
1980	Road, Night
1979	Love, Time, Heart
1978	Rock, Strangers, Eyes, Love, Horror, Kisses
1977	Sun, Girl
1976	One, Love
1975	Home. Life, Darkness, Music, Heart, Miles
1974	Timing, Fighting, Fact
1973	Road, Money
1972	Nothing, Cruise, Love, Summertime, Wine, Heart, Skies, Friend, Face, Gold, Blues, Afternoon
1971	Case, Rain
1970	Stars, Girl, Rain, Fire
1969	Chance, Peace
1968	Baby, Heart
1967	Sun, Rivers, Roads
1965	Time, Sea
1964	Change, Sun, Time
1963	One, Nothing
1962	Mistake, Notion, Cheer, Today, Places, Girl, Name, Friend, Care
1961	Dream, World
1960	Arms, Fool
1959	Music
1958	Guess
1956	Arms
1955	Prison, Train, Folsom
1944	Merry, Christmas
1942	Mamma, God, News, Door, Papa, Chile, Crowding, Friends, Money, Bible
1937	Late

Table 6- 8: Hot Topics-B

## **Chapter 7 Conclusion and Further Work**

The chapter presents a conclusion of the thesis and some improvement suggestions for the current work.

### **7.1 Conclusion**

Until now, we have built an opinion mining system for song lyrics, which fetch objects of interest and opinion words about them. We randomly collect some song lyrics to do test for evaluation of how good the system works. The result from the system is compared with manual identification. From the result from Experiment 1 and Experiment 2, we find that the system basically can present topics of one song lyrics and opinion words about the topics. Moreover, the system can run a collection of song lyrics and present some interesting results, e.g. fetching most common topics, presenting the number of polarity words for each musical type or different year. As well, we can analyze opinion change on some common topics as time changes. All these mentioned above meets the requirement from problem definition for the thesis.

In the process of constructing the system, we explore carefully, refer to many previous works, compare between different methods (e.g. explore how to fetch objects of interest). As well, we have done a lot of programming work to realize our idea, applying theory into practice, doing experiment to evaluate our result, finally analyzing experiment result and present some interesting things.

Besides, we have developed a program in Java for collecting song lyrics on Internet from one website. The program can help us collect thousands of song lyrics (we have collected about 20,000 song lyrics) and search information of song publishing year or musical genre on Wikipedia.org (we have obtained about 1,000 song lyrics with their published years at least).

The work in opinion mining for song lyrics is few at present. The thesis finishes an exploration in the subject. We believe the exploration is valuable and useful for further work since it is not perfect.

## 7.2 Further Work

Next, we present some points on which we think the current work can be improved further.

1. Now the system is only for English Lyrics, so we consider if it is possible for other languages' lyrics.
2. The system only fetches objects of interest on one-word level, although we can construct a phrase with opinion words describing the objects of interest. Like the phrase "American Idol", the word "idol" can be fetched, but "American" maybe not considered as opinion words in the system, then we miss the important phrase. So it is better if we can fetch objects of interest on phrase-level.
3. The word "not" is not considered as a situation when fetching opinion words. We can imagine that sometimes the word "not" determines two kinds of opposite sentiment polarities.
4. In collecting song lyrics on Internet, we did not fetch the information of song genre very well, so that we do not have enough songs with the information for analyzing hot topics for different song genres. Moreover, it also should be better if analyzing sentiment polarity (the number of positive words and the number of negative words) for different song genres in bigger collection of song lyrics.
5. We should collect more song lyrics with their published year at least. If the number can reaches more than 10,000, It should be better to fetch hot topics in different year. Now, only 1,608 song lyrics with the published year, we run the experiment and find it is not enough to see some interesting change of hot topics in different year. Its result is also similar with the experiment running on about 500 song lyrics. A larger amount of song lyrics can ensure fetching hot topics in different year more exactly.

## References

- [1] M. Hu and B. Liu, "Mining opinion features in customer reviews," in *Proceedings of AAAI*, pp. 755–760, 2004.
- [2] Miller, G., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K.1990. "Introduction to WordNet: An on-line lexical database," *International Journal of Lexicography*, 3(4):235-312.
- [3] A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 2005.
- [4] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study." *Artificial Intelligence*, 165(1):91–134, 2005.
- [5] P. D. Turney, "Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL." In *Procs. of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, Freiburg, Germany, 2001.
- [6] Gerald Chao and Michael G.Dyer, "Word Sense Disambiguation of Adjectives Using Probabilistic Networks", in *Proceedings of the 17th International Conference on Computational Linguistics*, Saarbrücken, pages 152-158, 2000.
- [7] Anaya-S\_anchez, H., Pons-Porrata, A., Berlanga-Llavori, R., "Word sense disambiguation based on word sense clustering," In *J.S. Sichman et al. (Eds.), Lecture Notes in Artificial Intelligence*, vol. 4140, 472-481, Springer, 2006
- [8] Hevner, March and Jinsoo, "Design Science in Information Systems Research," *MIS Quarterly* Vol. 28 2004, pp. 75-105.
- [9] Olena Medelyan, "Human-competitive automatic topic indexing," partial fulfillment for the degree of Doctor of Philosophy in Computer Science at The University of Waikato, Hamilton, New Zealand, July 2009
- [10] Witten, I. H., G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: Practical automatic keyphrase extraction," In *Proc. ACM Conf. on Digital Libraries*, Berkeley, CA, US. New York, NY: ACM Press, pp.254–255, 1999.
- [11] Witten, I. H., and E. Frank, "Data mining: Practical machine learning tools and techniques with Java implementations," 2nd edition. San Francisco, CA:Morgan Kaufmann, 2005.
- [12] McBride, B., "Jena: Implementing the RDF Model and Syntax Specification," In *S. Staab et al. (eds.): Proc. 2nd Int. Workshop on the Semantic Web, SemWeb'01*, Hong Kong, China, 2001.
- [13] Milne, D., "An open-source toolkit for mining Wikipedia," In *Proc. New Zealand Computer Science Research Student Conf.*, NZCSRSC'09, Auckland, New Zealand, 2009.
- [14] Baeza-Yates and Ribeiro-Neto, *Modern Information Retrieval*. Pages: 29-30, Addison-Wesley. 1999.
- [15] A. Esuli and F. Sebastiani, "SentiWordNet: A publicly available lexical resource for opinion mining," in *Proceedings of Language Resources and Evaluation (LREC)*,

2006.

- [16] Y. Wilks and M. Stevenson, "The grammar of sense: Using part-of-speech tags as a first step in semantic disambiguation," *Journal of Natural Language Engineering*, vol. 4, pp. 135–144, 1998.
- [17] E. Brill, "Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging," *Computational Linguistics*, December 1995.
- [18] Princeton University Cognitive Science Laboratory, *WordNet [online]*, Available from: <http://wordnet.princeton.edu/>.
- [19] Antonio Toral, Oscar Ferrández, Eneko Agirre, Rafael Muñoz, "A study on Linking Wikipedia categories to Wordnet synsets using text similarity," *International Conference RANLP 2009 - Borovets, Bulgaria*, pages 449–454.
- [20] G. A. Miller et al, "Introduction to WordNet: An On-line Lexical Database," Technical Report, Cognitive Science Laboratory, Princeton University, 1993.
- [21] G. A. Miller, "Nouns in WordNet: A Lexical Inheritance System," Report, Cognitive Science Laboratory, Princeton University, 1993.
- [22] Christiane Fellbaum, Derek Gross, Katherine Miller, "Adjectives in WordNet," Report, Cognitive Science Laboratory, Princeton University, 1993.
- [23] Theresa Wilson, Janyce Wiebe and Paul Hoffmann, "Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis," in *Proceedings of HLT/EMNLP*, Vancouver, Canada, 2005.
- [24] Ellen Riloff and Janyce Wiebe. "Learning extraction patterns for subjective expressions." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*.
- [25] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of WWW*, pp. 519–528, 2003.
- [26] B. Liu, "Web data mining; Exploring hyperlinks, contents, and usage data," *Opinion Mining*. Springer, 2006.
- [27] J. Tatemura, "Virtual reviewers for collaborative exploration of movie reviews," in *Proceedings of Intelligent User Interfaces (IUI)*, pp. 272–275, 2000.
- [28] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "PHOAKS: A system for sharing recommendations," *Communications of the Association for Computing Machinery (CACM)*, vol. 40, pp. 59–62, 1997.
- [29] C. Cardie, C. Farina, T. Bruce, and E. Wagner, "Using natural language processing to improve eRulemaking," in *Proceedings of Digital Government Research (dg.o)*, 2006.
- [30] N. Kwon, S. Shulman, and E. Hovy, "Multidimensional text analysis for eRulemaking," in *Proceedings of Digital Government Research (dg.o)*, 2006.
- [31] S. Shulman, J. Callan, E. Hovy, and S. Zavestoski, "Language processing technologies for electronic rulemaking: A project highlight," in *Proceedings of Digital Government Research (dg.o)*, pp. 87–88, 2005.
- [32] J. Carbonell, "Subjective Understanding: Computer Models of Belief Systems," PhD thesis, Yale, 1979.

- [33] Y. Wilks and J. Bien, “Beliefs, points of view and multiple environments,” in *Proceedings of the international NATO symposium on artificial and human intelligence*, pp. 147–171, USA, New York, NY: Elsevier North-Holland, Inc., 1984.
- [34] S. Das and M. Chen, “Yahoo! for Amazon: Extracting market sentiment from stock message boards,” in *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*, 2001.
- [35] R. M. Tong, “An operational system for detecting and tracking opinions in on-line discussion,” in *Proceedings of the Workshop on Operational Text Classification (OTC)*, 2001.
- [36] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.
- [37] V. Hatzivassiloglou and J. Wiebe, “Effects of adjective orientation and gradability on sentence subjectivity,” in *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2000.
- [38] Bing Liu, “Opinion Mining”, *Encyclopedia of Database Systems*, 2008.
- [39] Liu, B., Hu, M. and Cheng, J, “Opinion Observer: Analyzing and Comparing Opinions on the Web,” in *Proceedings of International World Wide Web Conference (WWW’05)*, 2005.
- [40] Alexander Thorsten Schutz, “Keyphrase Extraction from Single Documents in the Open Domain Exploiting Linguistic and Statistical Methods,” Master Thesis in National University of Ireland, 2008.
- [41] SmILE, <http://smile.deri.ie/>, accessed on 21th Oct, 2010.
- [42] UIMA, <http://uima.apache.org/>, accessed on 22th Oct. 2010
- [43] Bo Pang and Lillian Lee, “Opinion Mining and Sentiment Analysis,” *Foundations and Trends® in Information Retrieval*: Vol. 2: No 1–2, pp 1-135, 2008.

## Appendix A: Integration of OpenNLP and UIMA in Eclipse

Actually, we integrate OpenNLP into the project uimaj-examples, which is contained in the source file of UIMA Java Framework & SDK that can be downloaded on <http://uima.apache.org/downloads.cgi>. To finish integration, we have to do the following steps:

1. **Download OpenNLP Tools Package**, go to <http://opennlp.sourceforge.net/> and download the OpenNLP Tools Package 1.3, which should contain the source files, the class files, java documentations and so on.
2. **Compile OpenNLP Tools Package** into jar file as the library that is imported into the project uimaj-exmaple later.
3. **Download the Model files**, go to <http://opennlp.sourceforge.net/> and choose “Models” at the bottom, you can choose model as you want, for example, the tags library file for part-of-speech tagging.
4. Set **the UIMA Wrappers for OpenNLP** as the source file of the project uimaj-example, UIMA Wrappers for OpenNLP is actually named “opennlp-wrappers” in a sub-directory of the project “uimaj-example”.

5. **Input locations of model files** in the matched opennlp-wrappers XML descriptor. For example, now we open the descriptor: opennlp-wrappers/descriptors/OpenNLPPostagger.xml as showed in the Figure Appendix-1 below

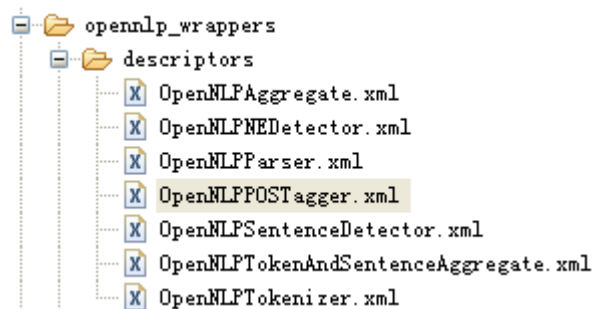


Figure Appendix- 1: XML Descriptors of UIMA Wrappers for OpenNLP

Then, we click the tab “Parameter Settings” and see the Figure Appendix-2 showed below, now input the location of the model file for part-of-speech tag in the right of the figure.

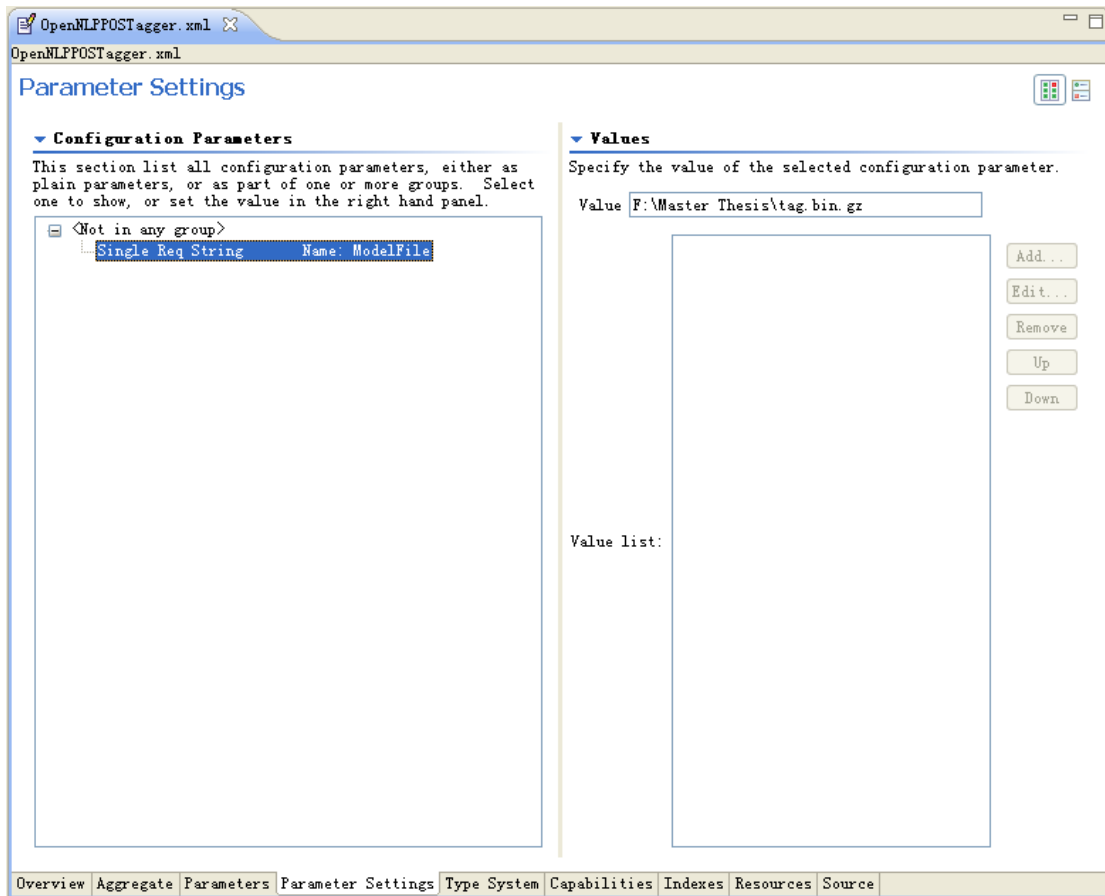


Figure Appendix- 2: Parameter Settings in XML Descriptors



## Appendix B: Song Lyrics

NUMBER 1: 1978\_Reggae\_-\_Bob\_Marley\_-\_Babylon\_By\_Bus\_-\_Is\_This\_Love\_-\_111.txt

I wanna love you and treat you right  
I wanna love you every day and every night  
We'll be together with a roof right over our heads  
We'll share the shelter of my single bed  
We'll share the same room, yeah, but JAH provide the bread

Is this love, is this love, is this love  
Is this love that I'm feelin'?  
Is this love, is this love, is this love  
Is this love that I'm feelin'?

I wanna know, wanna know, wanna know now  
I got to know, got to know, got to know now  
I, I, I, I, I, I, I, I, I, I, I, I, I'm willing and able  
So I throw my cards on your table

I wanna love you, I wanna love and treat, love and treat you right  
I wanna love you every day and every night  
We'll be together yeah, with a roof right over our heads  
We'll share the shelter yeah, oh yeah, of my single bed  
We'll share the same room yeah, but JAH provide the bread

Is this love, is this love, is this love  
Is this love that I'm feelin'?  
Is this love, is this love, is this love  
Is this love that I'm feelin'?  
Wo-o-o-oah!

Oh yes I know, yes I know, yes I know now  
Oh yes I know, yes I know, yes I know now  
I, I, I, I, I, I, I, I, I, I, I, I, I'm willing and able  
So I throw my cards on your table

See I wanna love ya, I wanna love and treat ya, love and treat ya right  
I wanna love you every day and every night  
We'll be together with a roof right over our heads  
We'll share the shelter of my single bed  
We'll share the same room yeah, but JAH provide the bread  
We'll share the shelter of my single bed.

**NUMBER2:**

**1984\_Folk\_-\_The\_Pogues\_-\_The\_Ultimate\_Collection\_(Disc\_2)\_\_-\_Streams\_Of\_Whiskey\_-\_45.txt**

Last night as I slept  
I dreamt I met with Behan  
I shook him by the hand and we passed the time of day  
When questioned on his views  
On the crux of life's philosophies  
He had but these few clear and simple words to say

I am going, I am going  
Any which way the wind may be blowing  
I am going, I am going  
Where streams of whiskey are flowing

I have cursed, bled and sworn  
Jumped bail and landed up in jail  
Life has often tried to stretch me  
But the rope always was slack  
And now that I've a pile  
I'll go down to the Chelsea  
I'll walk in on my feet  
But I'll leave there on my back

Because I am going, I am going  
Any which way the wind may be blowing  
I am going, I am going  
Where streams of whiskey are flowing

Oh the words that he spoke  
Seemed the wisest of philosophies  
There's nothing ever gained  
By a wet thing called a tear  
When the world is too dark  
And I need the light inside of me  
I'll walk into a bar  
And drink fifteen pints of beer

I am going, I am going  
Any which way the wind may be blowing  
I am going, I am going  
Where streams of whiskey are flowing

I am going, I am going  
Any which way the wind may be blowing

I am going, I am going  
Where streams of whiskey are flowing  
Where streams of whiskey are flowing  
Where streams of whiskey are flowing.

**NUMBER 3: 1989\_Grunge\_-\_Nirvana\_-\_Bleach\_(Remastered)\_-\_Negative\_Creep\_-\_499.txt**

This is out of our range  
This is out of our range  
This is out of our range  
no  
This is getting to be  
This is getting to be  
This is getting to be  
gross  
I'm a negative creep  
I'm a negative creep  
I'm a negative creep  
and I'm stoned  
I'm a negative creep  
I'm a negative creep  
I'm a negative creep  
and I'm  
I'm  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more

This is out of our range  
This is out of our range  
This is out of our range  
no  
This is getting to be  
This is getting to be  
This is getting to be  
gross  
I'm a negative creep  
I'm a negative creep  
I'm a negative creep  
and I'm stoned  
I'm a negative creep  
I'm a negative creep

I'm a negative creep  
and I'm  
I'm  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more

Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more  
Daddy's little girl ain't a girl no more.

**NUMBER 4: 1992\_Grunge\_-\_Mudhoney\_-\_Superfuzz\_Bigmuff\_plus\_Early\_-\_Chain\_That\_Door\_-\_228.txt**

Where do you think we've seen that girl?  
Chain that door, I'm outta your world

We know  
We know  
We know what I mean

You made me feel like a big-time loser  
Chain that door, it's all over  
I said  
You made me feel like a big-time loser  
Chain that door, girl, it's all over

We know  
We know  
We know what I mean

See that I'm walking the way I am  
Why you always hang around?

We know  
We know  
We know what I mean.

NUMBER 5: 1992\_Grunge\_-\_Alice\_in\_Chains\_-\_Dirt\_-\_Angry\_Chair\_-\_463.txt

Sitting on an angry chair  
Angry walls that steal the air  
Stomach hurts and I don't care

What do I see across the way, hey  
See myself molded in clay, oh  
Stares at me, yeah I'm afraid, hey  
Changing the shape of his face, aw yeah

Candles red I have a pair  
Shadows dancing everywhere  
Burning on the angry chair

Little boy made a mistake, hey  
Pink cloud has now turned to gray, oh  
All that I want is to play, hey  
Get on your knees, time to pray, boy

I don't mind, yeah  
I don't mind, I-I-I  
I don't mind, yeah  
I don't mind, I-I-I  
Lost my mind, yeah  
But I don't mind, I-I-I  
Can't find it anywhere  
I don't mind

Corporate prison, we stay, hey  
I'm a dull boy, work all day, oh  
So I'm strung out anyway, hey

Loneliness is not a phase  
Field of pain is where I graze  
Serenity is far away

Saw my reflection and cried, hey  
So little hope that I died, oh  
Feed me your lies, open wide, hey  
Weight of my heart, not the size, oh

I don't mind, yeah  
I don't mind, I-I-I  
I don't mind, yeah

I don't mind, I-I-I  
Lost my mind, yeah  
But I don't mind, I-I-I  
Can't find it anywhere  
I don't mind, I-I-I

Pink cloud has now turned to gray  
All that I want is to play  
Get on your knees time to pray, boy.

**NUMBER 6: 1998\_Metal\_-\_System\_Of\_A\_Down\_-\_System\_Of\_A\_Down\_-\_Suite-pee\_-\_231.txt**

I had an out of body experience  
The other day  
Her name was Jesus  
And for her everyone cried  
Everyone cried, everyone cried

Try her philosophy, try her philosophy  
Try her philosophy, try  
You die for her philosophy  
Die for her philosophy  
Die her philosophy die

Crossed and terrored ravages of architecture  
Lend me thy blades  
We're crossed and terrored ravages of architecture  
Hoist around the spade

Try her philosophy, try her philosophy  
Try her philosophy, try  
You die for her philosophy  
Die for her philosophy  
Die her philosophy, die

Die, die, die, why

Lie naked on the floor  
And let the Messiah go through our souls  
Lie naked on the floor  
And let the Messiah go all through our souls

Die, like a motherfucker  
Die, like a motherfucker

Die, like a motherfucker  
Why, like a motherfucker

I want to fuck my way to the garden  
'Cause everyone needs a mother fucker

The following of a Christ, the following of a Christ  
The following of a Christ, the following of a Christ  
The falling of Christ, the falling of Christ  
The falling of Christ, the falling of Christ.

**NUMBER 7: 1999\_Country\_-\_Wilco\_-\_Summerteeth\_-\_In\_A\_Future\_Age\_-\_139.txt**

Genuine  
Day will come  
When the wind  
Decides to run  
And shakes the stairs  
That stab the wall  
And turns the page  
In a future age

Some trees will bend  
And some will fall  
But then again  
So will us all  
Lets turn our prayers  
Into outrageous dares  
And mark our page  
In a future age

High above  
The sea of cars  
And barking dogs  
In fenced-in yards.

**NUMBER 8: 2002\_Country\_-\_Dixie\_Chicks\_-\_Home\_-\_White\_Trash\_Wedding\_-\_253.txt**

You can't afford no ring  
You can't afford no ring  
I shouldn't be wearing white  
and you can't afford no ring

You finally took my hand  
You finally took my hand  
It took a nip of gin

but you finally took my hand

You can't afford no ring  
You can't afford no ring  
I shouldn't be wearing white  
and you can't afford no ring

Mamma don't approve  
Mamma don't approve  
Daddy says he's the very best  
And mamma don't approve

You can't afford no ring  
You can't afford no ring  
I shouldn't be wearing white  
and you can't afford no ring

Baby's on its way  
Baby's on its way  
Say I do and kiss me quick  
'Cause baby's on its way

I shouldn't be wearing white and you can't afford no ring!

**NUMBER 9: 2002\_Slow\_Rock\_-\_Coldplay\_-\_A\_Rush\_Of\_Blood\_To\_The\_Head\_-\_A\_Whisper\_-\_480.txt**

A whisper, whisper, whisper, whisper  
A whisper, whisper, whisper, whisper

I hear the sound of the ticking of clocks  
Who remembers your face  
Who remembers you when you are gone  
I hear the sound of the ticking of clocks  
Come back and look for me  
Look for me when I am lost  
And just a whisper, whisper, whisper, whisper

Just a whisper, whisper, whisper, whisper

Night turns to day  
And I still have these questions  
Bridges will break  
Should I go forwards or backwards  
Night turns to day  
And I still get no answers



Just a whisper, whisper, whisper, whisper  
A whisper, whisper, whisper

(just a whisper, whisper, whisper, whisper)  
I hear the sound of the ticking of clocks  
Who remembers your face  
Who remembers you when you are gone

I hear the sound of the ticking of clocks  
Come back and look for me  
Look for me when I am lost  
And I am just a whisper, a whisper, a whisper, a whisper

Just a whisper, whisper, whisper, whisper

Oh ha ah ah ah ah ah ah.

**NUMBER 10: 2003\_Pop -\_Britney\_Spears -\_In\_The\_Zone -\_Shadow -\_550.txt**

Your body's warm but you are not  
You give a little not a lot  
You coup your love until we kiss  
You're all I want but not like this

I'm watching you disappear  
But you, you were never here

It's only your shadow, never yourself  
It's only your shadow, nobody else  
It's only your shadow, filling the room  
Arriving too late, and leaving too soon  
And leaving too soon...

Your body gives but then holds back  
The sun is bright, the sky is black  
Can only be another sign  
I cannot keep what isn't mine

You left and it lingers on  
But you, you were almost gone

It's only your shadow, never yourself  
It's only your shadow, nobody else  
It's only your shadow, filling the room

Arriving too late, and leaving too soon  
And leaving too soon...

I cannot tell if you mean what you say  
You say it so loud, but you sound far away  
Maybe I had just a glimpse of your soul  
Or was that your shadow I saw on the wall

I'm watching you disappear  
But you, you were never here

It's only your shadow, never yourself  
It's only your shadow, nobody else  
It's only your shadow, filling the room  
Arriving too late  
No, no, no  
It's only your shadow  
It's only your shadow, nobody else  
It's only your shadow  
Arriving too late and leaving too soon

It's only your shadow.

## Appendix C: Code Examples

### KeywordsByNouns.java

This is a typical annotation class in our source code for annotating keywords or sentiment words in text. It runs sentence by sentence in text.

```
public class KeywordsByNouns extends JCasAnnotator_ImplBase {

    public TextFileIndexer index=new TextFileIndexer();
    /**
     * Initialize the Annotator.
     *
     * @see JCasAnnotator_ImplBase#initialize(UimaContext)
     */
    public void initialize(UimaContext aContext) throws
        ResourceInitializationException {
        super.initialize(aContext);

        try {
            index.buildIndex();
            System.out.println("INDEX DONE");
        } catch (Exception e) {
            throw new ResourceInitializationException(e);
        }
    }

    /**
     * Process a CAS.
     *
     * @see JCasAnnotator_ImplBase#process(JCas)
     */
    public void process(JCas aJCas) throws AnalysisEngineProcessException
    {

        ArrayList<Token> tokenList = new ArrayList<Token>();
        ArrayList wordList = new ArrayList();
        HashMap<Object,Integer> tokenList1 = new HashMap<Object,Integer>();
        AnnotationIndex sentenceIndex =
            aJCas.getAnnotationIndex(Sentence.type);
        AnnotationIndex tokenIndex = aJCas.getAnnotationIndex(Token.type);

        // iterate over Sentences
        FSIterator sentenceIterator = sentenceIndex.iterator();
```

```

// iterator each sentence;
while (sentenceIterator.hasNext()) {
    Sentence sentence = (Sentence) sentenceIterator.next();

    // iterate over Tokens
    FSIterator tokenIterator = tokenIndex.subiterator(sentence);

    while (tokenIterator.hasNext()) {

        Token token = (Token) tokenIterator.next();
        tokenList.add(token);
        String posTag = token.getPosTag();

        if(posTag!=null&&posTag.equals("NN")||posTag.equals("NNS")||posTag.equals("NNP")){

            if(tokenList1.get(token.getCoveredText()) != null){
                tokenList1.put(token.getCoveredText(),tokenList1.get(token.getCoveredText()+1); }
            else{
                tokenList1.put(token.getCoveredText(),1);
            }
        }
    }
}

// ranking noun term frequency
Object[] value = tokenList1.values().toArray() ;
Arrays.sort(value);

for(Object o:tokenList1.keySet()){
    if(tokenList1.get(o).equals(value[value.length-1])||tokenList1.get(o).equals(value[value.length-2])||tokenList1.get(o).equals(value[value.length-3])){

        for(int i=0; i<tokenList.size();i++){
            if(tokenList.get(i).getCoveredText().equals(o)){
                // annotate KeywordsByTF in text
                KeywordsByTF keywords=new KeywordsByTF(aJCas);
                keywords.setBegin(tokenList.get(i).getBegin());
                keywords.setEnd(tokenList.get(i).getEnd());
                keywords.addToIndexes();    } }} }}

```

## SearchWikipedia.java: Search on Wikipedia

The class is used for searching information about year or genre or others on Wikipedia.org with the keywords input of song artist and song title. Among, we import the library form Cobra Tool Kit on <http://lobobrowser.org/cobra.jsp> to parse HTML file.

```
import java.io.File;
import java.io.IOException;
import org.lobobrowser.html.domimpl.HTMLDocumentImpl;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class SearchWikipedia {

    String genre;
    String year;
    String TEST_URI2 ;

    public SearchWikipedia(String artist, String title) throws
IOException, SAXException{

        TEST_URI2 =URLUtils.addParameter("http://en.wikipedia.org/",
"search", artist+" "+title);
        HTMLDocumentImpl document2 =ParsingURL.getHTML(TEST_URI2);
        NodeList uls=document2.getElementsByTagName("ul");
        String TEST_URI3=null;
        HTMLDocumentImpl document3=null;

        for(int k=1;k<uls.item(2).getChildNodes().getLength();k=k+2){

            boolean flag=true;
            String[] titles=title.split(" ");

            for(int t=1;t<titles.length;t++){

if(!uls.item(2).getChildNodes().item(k).getChildNodes().item(0).toStr
ing().contains(titles[t])&&!uls.item(2).getChildNodes().item(k).getCh
ildNodes().item(0).toString().contains(titles[t].toLowerCase()))
                {
                    flag=false;
                    break;
                }
            }
        }
    }
}
```

```

        }
    }

    if(flag==true)
    {

TEST_URI3="http://en.wikipedia.org"+uls.item(2).getChildNodes().item(
k).getChildNodes().item(0);
        if(TEST_URI3!=null){
            document3 =ParsingURL.getHTML(TEST_URI3);
            break;
        }
    }
}

NodeList tables=null;
if(document3!=null)
    tables=document3.getElementsByTagName("table");

if(tables!=null){
    for(int i = 0; i < tables.item(0).getChildNodes().getLength(); i++)
    {

if(tables.item(0).getChildNodes().item(i).getTextContent().contains("
Released"))

year=tables.item(0).getChildNodes().item(i).getTextContent().replace(
"Released", "");

if(tables.item(0).getChildNodes().item(i).getTextContent().contains("
Genre")){

genre=tables.item(0).getChildNodes().item(i).getTextContent().replace
("Genre", "");
        }
    }
}

public String getYear(){
    return year;
}

```

```

public String getGenre(){
    return genre;
}

public String getURL(){
    return TEST_URI2;
}

public static void main(String[] args) throws Exception {

    File fileDir = new File("F:/Master Thesis/newLyrics/A");
    File[] textFiles = fileDir.listFiles();

    for(int k=0; k<textFiles.length;k++){

String[] name=textFiles[k].getName().replace(".txt","").split("-");
        String artist=name[0];
        String title=name[1];

        SearchWikipedia search=new SearchWikipedia(artist,title);

        if(search.getYear()!=null){

            char[] year=search.getYear().toCharArray();
            String reyear=null;
            for(int i=0;i<year.length;i++){

if((i+1)<year.length&&(i+2)<year.length&&(i+3)<year.length){

if(Character.isDigit(year[i])&&Character.isDigit(year[i+1])&&Character
.isDigit(year[i+2])&&Character.isDigit(year[i+3]))

reyear=Character.toString(year[i])+Character.toString(year[i+1])+Ch
aracter.toString(year[i+2])+Character.toString(year[i+3]);
                }
            }

            File file;
            if(search.getGenre()!=null)
                file=new File("F:/Master
Thesis/new/"+reyear+"_"+search.getGenre().substring(2,search.getGen
re().length()-1)+"_"+"-"+artist+"-"+albumn+"-"+title+"-"+".txt");
            else

```

```

        file=new File("F:/Master
Thesis/new/"+realyear+"_"+"Genre"+"_"+"-"+"artist"+"-"+"albumn"+"-"+"tit
le+"-"+" .txt");
        textFiles[k].renameTo(file);
    }
}

System.out.println("DONE");

}
}

```

## FetchSongLyrics.java

The class is used to fetch song lyrics from the website <http://www.absolutelyrics.com/>, and save each song lyrics into separate .txt file.

```

import org.lobobrowser.html.domimpl.*;
import org.w3c.dom.*;
import java.util.HashMap;
import java.io.*;

public class FetchSongLyrics {

    public static void main(String[] args) throws Exception {

String[] letters=
{"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q",
"r","s","t","u","v","w","x","y","z"};

        HashMap<String,Integer> links=new HashMap<String,Integer>();
        links.put("a",106 );
        links.put("b",115 );
        links.put("c",102 );
        links.put("d",97 );
        links.put("e",43 );
        links.put("f",74 );
        links.put("g",67 );
        links.put("h",93 );
        links.put("i",153 );
        links.put("j",29 );
        links.put("k",21 );
        links.put("l",111 );

```



```

links.put("m",98 );
links.put("n",58 );
links.put("o",56 );
links.put("p",68 );
links.put("q",5 );
links.put("r",64 );
links.put("s",212 );
links.put("t",205 );
links.put("u",22 );
links.put("v",12 );
links.put("w",123 );
links.put("x",2 );
links.put("y",43 );
links.put("z",3 );

String uri=null;

for(String letter:letters){

    for(int p=1;p<=links.get(letter);p++){

uri="http://www.absolutelyrics.com/lyrics/songlist/"+letter+"/"+p+"/"
;

        HTMLDocumentImpl document = ParsingURL.getHTML(uri);
        NodeList lis=document.getElementsByTagName("li");
        for(int k=0;k<lis.getLength();k++) {
            String lyrics = null;
            String TEST_URI1=
"http://www.absolutelyrics.com"+lis.item(k).getChildNodes().item(0);

            HTMLDocumentImpl document1 = ParsingURL.getHTML(TEST_URI1);
            Element div= document1.getElementById("realText");
            NodeList titles=document1.getElementsByTagName("title");
            Node title=titles.item(0);
            String[] strings=title.getTextContent().split("::");

for(int i=0;i< div.getChildNodes().getLength();i++){
    for(int t=0;
        t<div.getChildNodes().item(i).getChildNodes().getLength();t++)

lyrics+=div.getChildNodes().item(i).getChildNodes().item(t).getTextCo
ntent()+"\r\n";

        }

```

```

        File filename = null;
        filename = new File("F:/Master
Thesis/newLyrics/F/"+strings[0]+"-"+strings[1].replace("Lyrics",
"+"+".txt");

        if (!filename.exists())
            filename.createNewFile();

BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(filename));

        if(lyrics!=null)
            bufferedWriter.write(lyrics.replace("null",
""));

            bufferedWriter.flush();
            bufferedWriter.close();

    } }}}}

```

## SubLexicon.java

The class read data from the list of Subjective Clues.

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.HashMap;
import java.util.Vector;

public class SubLexicon {

    private String path =
"resources/SubLexicon/subjclueslen1-HLTEMNLP05.txt";
    public double negative;
    public double positive;

    private HashMap<String, Vector<String>> _dict;

    public SubLexicon(){

        _dict = new HashMap<String, Vector<String>>();

        try{

```

```

        BufferedReader csv = new BufferedReader(new
FileReader(path));
        String line = "";
        while((line = csv.readLine()) != null)
        {
            String[] words = line.split(" ");

            String[] types=words[0].split("=");
            String type=types[1];

            String[] lens=words[1].split("=");
            String len=lens[1];

            String[] wordls=words[2].split("=");
            String wordl=wordls[1];

            String[] posls=words[3].split("=");
            String posl=posls[1];

            String[] stemmedls=words[4].split("=");
            String stemmedl=stemmedls[1];

            String[] priorpolaritys=words[5].split("=");
            String priorpolarity= priorpolaritys[1];

            Vector<String> vector=new Vector<String>();
            vector.add(type);
            vector.add(len);
            vector.add(stemmedl);
            vector.add(priorpolarity);

            _dict.put(wordl+"#+posl, vector);

        }

    }

    catch(Exception e){e.printStackTrace();}
}

public Vector<String> extract(String word, String pos)

```

```

{
    return _dict.get(word+"#+pos);
}

public String getType(String word, String pos){

if(_dict.get(word+"#+pos)!=null)
    return _dict.get(word+"#+pos").elementAt(0);
else
    return null;

}

public String getLen(String word, String pos){

if(_dict.get(word+"#+pos")!=null)
    return _dict.get(word+"#+pos").elementAt(1);
else
    return null;

}

public String getStem(String word, String pos){

if(_dict.get(word+"#+pos")!=null)
    return _dict.get(word+"#+pos").elementAt(2);
else
    return null;

}

public String getPolarity(String word, String pos){

if(_dict.get(word+"#+pos")!=null)
    return _dict.get(word+"#+pos").elementAt(3);
else
    return null;

}

```

```

    public static void main(String args[]) throws Exception {
        System.out.println("!!!start!!!");
        SubLexicon sub=new SubLexicon();
        System.out.println("!!!"+sub.extract("like", "verb"));
        System.out.println("!!!"+sub.getType("like", "verb"));
        System.out.println("!!!"+sub.getLen("like", "verb"));
        System.out.println("!!!"+sub.getStem("like", "verb"));
        System.out.println("!!!"+sub.getPolarity("like", "verb"));
    }
}

```

## **SWN3.java**

The class reads data from the lexicon resource of SentiWordNet. The code refers to sample code on home page of SentiWordNet <http://sentiwordnet.isti.cnr.it/>.

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.HashMap;
import java.util.Vector;

public class SWN3 {

    private String pathToSWN = "resources/SentiWordNet_3.0.0.txt";
    private HashMap<String, Vector<Double>> _dict;

    public SWN3(){

        _dict = new HashMap<String, Vector<Double>>();

        try{
            BufferedReader csv = new BufferedReader(new
FileReader(pathToSWN));
            String line = "";
            while((line = csv.readLine()) != null)
            {

                if(!line.startsWith("#")){
                    String[] data = line.split("\t");
                    String[] words = data[4].split(" ");

```

```

        for(String w:words)
        {
            String[] w_n = w.split("#");
            if(w_n[1].equals("1")){
                Vector<Double> vector=new Vector<Double>();
                vector.add(Double.parseDouble(data[2]));
                vector.add(Double.parseDouble(data[3]));
                w_n[0] += "#"+data[0];
                _dict.put(w_n[0], vector);
            }
        }
    }
}

    catch(Exception e){e.printStackTrace();}
}

public Vector<Double> extract(String word, String pos)
{
    return _dict.get(word+"#+pos);
}

public static void main(String args[]) throws Exception {
    System.out.println("!!!start!!!");
    SWN3 swn=new SWN3();
    System.out.println("!!!"+swn.extract("little", "a"));
}
}
}

```

## **Appendix D: the List of Stop Words for Song's Title**

If these words in the list of stop words showed below appear in song's title, they will be filtered and will not be considered as keywords of song.

The list of stop words is:

a  
an  
and  
are  
at  
be  
but  
by  
do  
for  
in  
if  
into  
is  
it  
no  
not  
such  
of  
on  
or  
that  
the  
their  
then  
there  
these  
they  
this  
to  
was  
were  
will  
with  
you