A Computational Approach to the Analysis and Generation of Emotion in Text

by

Fazel Keshtkar

Thesis Submitted to the
Faculty of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
For the Degree of Ph.D. of Computer Science (Ph.D.)

Ottawa-Carleton Institute for Computer Science
School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

Sentiment analysis is a field of computational linguistics involving identification, extraction, and classification of opinions, sentiments, and emotions expressed in natural language. Sentiment classification algorithms aim to identify whether the author of a text has a positive or a negative opinion about a topic. One of the main indicators which help to detect the opinion are the words used in the texts. Needless to say, the sentiments expressed in the texts also depend on the syntactic structure and the discourse context. Supervised machine learning approaches to sentiment classification were shown to achieve good results. Classifying texts by emotions requires finer-grained analysis than sentiment classification.

In this thesis, we explore the task of emotion and mood classification for blog postings. We propose a novel approach that uses the hierarchy of possible moods to achieve better results than a standard flat classification approach. We also show that using sentiment orientation features improves the performance of classification. We used the LiveJournal blog corpus as a dataset to train and evaluate our method.

Another contribution of this work is extracting paraphrases for emotion terms based on the six basics emotions proposed by Ekman (*happiness, anger, sadness, disgust, surprise, fear*). Paraphrases are different ways to express the same information. Algorithms to extract and automatically identify paraphrases are of interest from both linguistic and practical points of view. Our paraphrase extraction method is based on a bootstrapping algorithms that starts with seed words. Unlike in previous work, our algorithm does not need a parallel corpus.

In Natural Language Generation (NLG), paraphrasing is employed to create more varied and natural text. In our research, we extract paraphrases for emotions, with the goal of using them to automatically generate emotional texts (such as friendly or hostile texts) for conversations between intelligent agents and characters in educational games.

Nowadays, online services are popular in many disciplines such as: e-learning, interactive games, educational games, stock market, chat rooms and so on. NLG methods can be used in order to generate more interesting and normal texts for such applications.

Generating text with emotions is one of the contributions of our work. In the last part of this thesis, we give an overview of NLG from an applied system's points of view. We discuss when NLG techniques can be used; we explained the requirements analysis and specification of NLG systems. We also, describe the main NLG tasks of content determination, discourse planning, sentence aggregation, lexicalization, referring expression generation, and linguistic realisation. Moreover, we describe our Authoring Tool that we developed in order to allow writers without programming skills to automatically generate texts for educational games.

We develop an NLG system that can generate text with different emotions. To do this, we introduce our pattern-based model for generation. We show our model starts with initial patterns, then constructs extended patterns from which we choose "final" patterns that are suitable for generating emotion sentences. A user can generate sentences to express the desired emotions by using our patterns. Alternatively, the user can use our Authoring Tool to generate sentences with emotions. Our acquired paraphrases will be employed by the tool in order to generate more varied outputs.

Life is a comedy for those who think and a tragedy for those who feel.

HORACE WALPOLE

Anybody can become angry—that is easy. But to be angry with the right person, to the right degree, at the right time, for the right purpose, and in the right way—this is not easy.

ARISTOTLE, *The Nicomachean Ethics*

Acknowledgements

My sincere gratitude goes to my academic supervisor Dr. Diana Inkpen, who helped, encouraged, and guided me toward my academic as well as my personal success. The thesis would not be successful without enormous help and advice from her. She provided me with a substantial amount of help and suggestions throughout my work.

I would like to give special thanks to Dr. Evangelos Millios[1], Dr. James Green[2], Dr. Stan Matwin[3], and Dr. Stan Szpakowicz[3] for being in my thesis committee and for their great advice and comments.

Special gratitude also goes to fellow students of the Natural Language Processing (NLP) group for many helpful discussions and their constant encouragement and support.

I wish to thank Dr. Soo-Min Kim and Prof. Ed Hovy for providing their polarity-tagged lists. I would like to thank Dr. Gilad Mishne for providing his LiveJournal corpus. I wish to thank Dr. Carlo Strapparava for providing the WordNet-Affect emotion data set. Also, I would like to thank Dr. Maria Fernanda Caropreso and Dr. Shahzad Khan for their contribution toward Chapter 5.

I would like to give my sincere thanks to my beloved parents for their support throughout my PhD studies. I am also grateful to all my friends and family who helped me either with suggestions or with positive affirmation.

Especially, I would like to give my special thanks to my wife Masha (Machoura Akmalkhodjaeva) whose patient love enabled me to complete this work. I thank her for all her support, encouraging, and help.

[1]Dalhousie University

[2]Carleton University

[3]University of Ottawa

Dedication

To my parents (Mohammad Hossein Keshtkar and Kobra Haghighat), for all their support in my entire life and career.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Emotions have been widely studied in psychology and behavioral sciences, as they are an important element of human nature. They have also attracted the attention of researchers in computer science, especially in the field of human-computer interaction, where studies have been carried out on facial expressions or on the recognition of emotions through a variety of sensors (e.g., (Picard, 1997)). In computational linguistics, the automatic detection of emotions in texts is becoming increasingly important, i.e., opinion mining and market analysis, affective computing, or natural language interfaces (e-learning environments or educational games) (Strapparava and Mihalcea, 2008). For example, the following represent examples of applicative scenarios in which affective analysis could make valuable and interesting contributions:

- **Sentiment Analysis:** The primary task in sentiment analysis is to classify a given document or text based on the polarity either positive, negative or neutral. Early works applied different machine learning methods to detect the polarity of product reviews and movie reviews(Turney, 2002; Pang and Lee, 2004). Therefore, text categorization based on affective, opinion mining for market analysis, etc., are examples of applications, and already developed by machine learning techniques (Strapparava and Mihalcea, 2008). These techniques mostly focus on identifying positive/negative valence annotation. However, emotion annotation/detection could increase the effectiveness of these applications.

- **Human Computer Interaction:** Affect analysis and generation can be more effective in human-computer interaction to emphasize naturalness and effectiveness. For example, the expression of emotions in embodied conversational agents nowadays is considered an important factor for their trustworthy. So, choosing good affective words is vital for expressive conversations.

Finding methods of automatically classifying and extracting the information from texts is the fundamental challenge in text analysis. Search engines that group similar documents by certain topics and categories, using keywords for topic definition are basic applications. Emotion and sentiment analysis methods extend them to another dimension, to detect the authors attitude toward the topic rather than just determining the topic.

In this thesis, a computational approach to emotion analysis and generation in text is proposed and implemented, in order to automatically determine and generate the author's emotions as expressed in texts, and to generate texts with specific emotions.

Below, we look at how humans express their emotions in various manner at different levels:

- **lexical level**: careful choice of words;
- **sentence level**: syntactic and semantic constructs; and
- **text/sub-text level**: discourse, that uses collocation terms (Wiebe et al., 2001).

Current methods concentrate on extracting expression such as words, phrases, and patterns that determine the emotion. The emotional analysis using these methods is not very accurate for all texts due to the ambiguity of natural language, and to the limited disambiguation capability of automatic methods. People are able to perfectly distinguish the expressed emotions because they understand the meaning of the words and phrases. They also are able to generate expressions and sentences for different emotions. However, developing a computer system to analyze, interpret, and generate different emotions is a difficult task.

1.1 What are Emotions?

The question is: what should we consider as the nature and the types of emotions? Following the example of Broad (1995), assume that a person were to say: "I am having an emotion". Then, two questions will arise:(1) "What kind of emotion?", and (2) "Towards what object?". The answer that we could expect for the first question is for example: "One of hatred" or "One of fear". The answer we could expect to the second question. is, for example: "Towards Smith", or "Towards a ghost". Broad (1995) suggests that every emotion is always a cognition, either veridical or wholly or partly delusive, but every emotion might have more than a mere cognition. Based on the Oxford English Dictionary (Plough, 2000), "Emotion" is defined as: "Any agitation or disturbance of mind, feeling, passion or excited mental state."

In this thesis, we focus on the six basic emotions that are: *anger, fear, happiness, sadness, disgust,* and *surprise*. Ekman (1992) believes that each emotion has certain features such as: signal, physiology, and antecedent events. He also proposed that each emotion has characteristics in common with other emotions, such as: rapid onset, short duration, unbidden occurrence, automatic appraisal, and coherence among responses. These common and unique characteristics can be considered in order to distinguish emotions from other affective phenomena such as moods, emotional traits and attitudes, and emotional disorders.

1.1.1 Emotion vs Mood

Emotions are momentary changes that influence the text written by a person. Mood is a medium-term state of a person, which is shifted by the emotions that are expressed. However, according to Ekman (1992) the above mentioned is not the only difference between emotions and moods. We summarize some of these differences as follows:

- Emotions are a matter of seconds, not minutes or hours. However, some researchers such as Frijda et al. (1992) proposed that emotions last between 5 seconds and several hours.

- Mood durations can be for hours or days.
- Moods are highly saturated with one or another emotion, for instance: irritability with anger, dysphoria with sadness, apprehensiveness with fear, euphoria with a type of enjoyment.
- Moods differ from emotions not only in duration, but also in what brings them forth and in their physiology.

1.2 Objectives of this Thesis

The goal of this work is to build a Natural Language Processing (NLP) system that allows flexible text analysis and generation, from the point of view of the emotions and moods expressed. We will achieve this by using automatic text categorization and information extraction techniques for the analysis part, and Natural Language Generation techniques for producing new texts, needed in learning simulations. The high-level architecture of the system is presented in Figure 1.1. Figure 1.2 shows in more detail the components that we implemented for each module of the high-level architecture.

This research is original because it combines the research areas: Natural Language Generation (NLG) and Text Analysis (classification by emotion and mood). Moreover, generating text with personality, including emotions and personalized style, is in fact a new research direction that we are opening in this research.

Previous research on topic-based classification algorithms turned out to be reasonably appropriate when used for emotion and sentiment classification (Pang and Lee, 2008). However, there are fundamental differences between the topic and emotion classification tasks. These differences suggest that significant room for improvement exists on the straightforward application of the traditional topic classification tools for emotion and sentiment analysis in text. For instance, Polanyi and Zaenen (2004) listed 13 contextual valence shifters that could be employed by authors to modify the sentiment expressed by the lexical constituents of the text; these contextual valence shifters have not tradi-

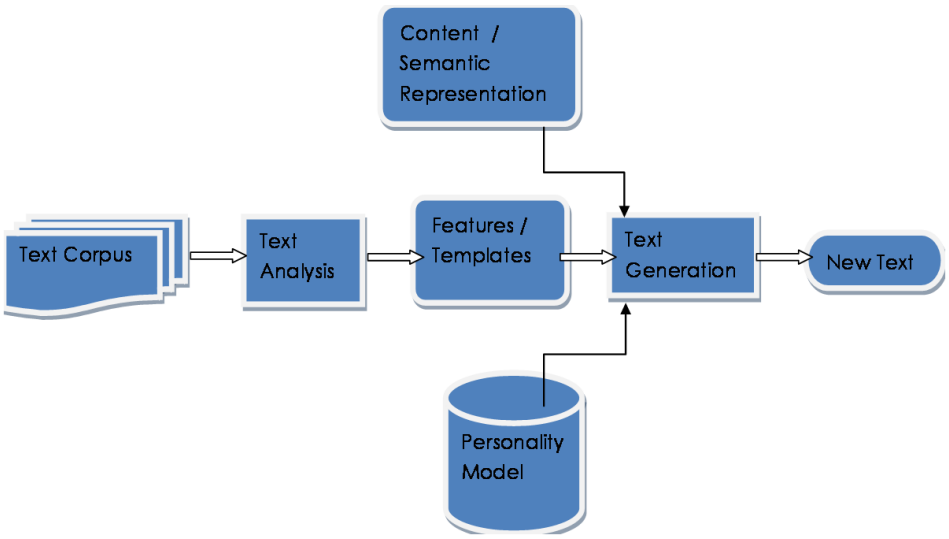

Figure 1.1: The system architecture: features and templates extracted during the text analysis phase will be used to generate text that expresses a given semantic content and has the desired stylistic properties.

tionally been considered important for topic classification (Khan, 2007). Identifying the emotion expression in texts (Strapparava and Mihalcea, 2008; Aman and Szpakowicz, 2007; Rubin et al., 2004; Strapparava and Mihalcea, 2008), emotion (Holzman and Pottenger, 2003; Wiebe et al., 2005), and mood classification (Mishne, 2005; Keshtkar and Inkpen, 2009) are examples of studies that automatically classified emotion and mood in texts. However, even if previous research is significant for emotion classification and emotion annotation based on topics and other semantic orientation features, there are more specific tasks that need to be addressed in emotion analysis.

Regarding text generation, there are numerous online systems for: e-learning, intelligent games, question-answering, machine translation, or in specific domains such as medical systems, customer service, stock markets, and so on, which use intelligent agent characters. For example, when using computer programs to generate content, the expectation is that a computer game can achieve the illusion of infinite variety of expression without the high human labour cost. Therefore, designing and implementing a Natural Language Generation System to modify natural language text to express emotions and

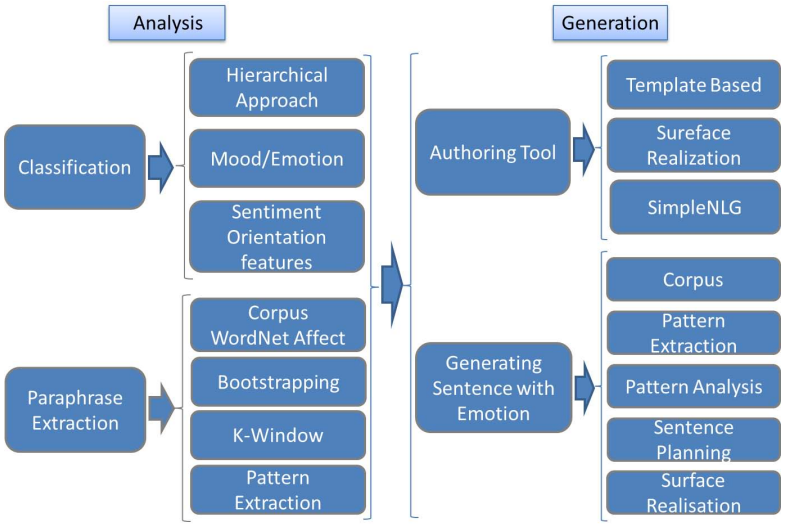

Figure 1.2: The components of the system architecture for analysis and generation in Figure 1.1

moods is an important task in online agent systems.

In this thesis, we will focus on the development of a system that can help the process of emotion analysis and generation that can be used in many products. At the same time, this system has interesting research achievements in itself, for other uses. This thesis studies the interplay of emotions expressed by lexical terms in the scope of emotion analysis and generation. We outline the objectives of this thesis in the following subsections.

1.2.1 Text Analysis

One of the objectives of this thesis is to provide a computational approach to text analysis that could improve the emotion classification accuracy. We design and implement methods for extracting desired features from text collections. To achieve this, we design and implement the following methods.

Feature Extraction for Emotions and Moods by Using Sentiment Orientation Scores

We categorize text by emotions and moods by using Sentiment Orientation Features to improve the accuracy of classifications. We extract features and templates for emotions and moods to be used for classification. We use machine learning algorithms to classify texts by the emotions they express. In particular Support Vector Machines (SVM) and Naive Bayes, which were shown to work well with texts, are used. The features that are strongly associated with each class of emotions will be selected for use in generation in the next step. In addition, we use text corpora (such as blogs) to extract features in an unsupervised manner. In this way, we will exploit any labeled dataset that is available, and supplement with un-labeled corpora, which are easily available.

A Hierarchical Approach to Text Classification for Moods and Emotions

We categorize text by emotions and moods by using hierarchical classification, in order to improve the accuracy of classification. We explore the task of mood classification for blog postings. Our novel approach uses a hierarchy of possible moods to achieve better results than a standard flat machine learning approach. We also show that using sentiment orientation features improves the performance of classification. We use the Livejournal blog corpus as a dataset to train and evaluate our method.

Extracting Paraphrases for Emotions

Another objective is to study the linguistic relation between emotion terms and their paraphrases, and quantify the utility of such patterns for leveraging these associations for paraphrase extraction. Both side of pair of paraphrases can be considered as pattern. Specifically, we want to see if the extracted paraphrases for emotions could improve emotion analysis. We design and implement a novel method that can extract paraphrases for emotion terms. We automatically learn rules and patterns that can be applied to the generation of emotional text in NLG systems.

1.2.2 Text Generation

Intelligent agents need text generation capabilities. For example, suppose a company uses business interview games that can be used by a human resource manager who is trained in a simulated environment. Creating the dialogue for these interviews will be a bottleneck for a company. Using a Natural Language Generation System that can create realistic dialogue for the intelligent conversational agents can alleviate this bottleneck. With such a technological solution, a writer can sketch out the skeleton of the dialogue and then the NLG system can create the myriad of different responses, based on a defined set of rules.

We develop methodologies to generate text with different emotions. For this task, we design and implement a method for modifying natural language text, to express emotion and mood, in the NLG system. We will refer to this mechanism as the "affective text" generation module.

Text Generation to Express Different Emotions

This step is a new direction of research, with very little previous work to build on. We design a way to generate text that has different emotion style. We make use of resources such as WordNet Affect (Strapparava and Valitutti, 2004), and we employ directly the expressions and patterns that we extracted. Then, for the actual sentence realization, we use the SimpleNLG (Gatt and Reiter, 2009) module.

Literature review and background research

We extensively studied the previous research in emotion analysis and generation area. We explore it and we find that our research is unique. Our work extends previous emotion analysis and classification research plus generation, by bridging the gaps identified in the previous research. The literature review is divided in two parts: *I*) sentiment, emotion, and mood analysis and classification, and *II*) Natural Language Generation System. More details of literature review is discussed in Chapter 2.

Data Collection

In this research we used four datasets. We briefly describe the datasets, as follows.

1. **LiveJournal blog dataset**: We used the blog corpus that Mishne collected for his research (Mishne, 2005). The corpus contains 815,494 blog posts from LiveJournal[1], a free weblog service used by millions of people. This corpus contains total of $69,149,217$ words. In LiveJournal, users are able to optionally specify their current emotion or mood. To select their emotion/mood users can choose from a list of 132 provided moods.

2. **Text Affect Dataset**: The second dataset (Strapparava and Mihalcea, 2007) consists of newspaper headlines that were used in the SemEval 2007-Task 14. It includes a development dataset of 250 annotated headlines, and a test dataset of 1000 news headlines.

3. **Fairy Tales Dataset**: This dataset consists in 1580 annotated sentences (Alm et al., 2005), from tales by the Grimm brothers, H.C. Andersen, and B. Potter. The annotations used the extended set of nine basic emotions of Izard (1971) (anger, disgust, fear, guilt, interest, happiness, sadness, shame and surprise).

4. **Annotated Blog Dataset**: We also used the dataset provided by Aman and Szpakowicz (2007). Emotion-rich sentences were selected from personal blogs, and annotated with the six emotions (as well as a non-emotion class, that we ignore here).

Adaptation of LiveJournal Corpus

We have updated the LiveJournal corpus to support linguistic processing. Even though the LiveJournal corpus (Mishne, 2005) is one of the main emotion classification datasets in our research and in the research community, it was not amenable to linguistic processing because the LiveJournal dataset is noisy and unstructured text (see Chapter 3 for

[1]http://www.livejournalinc.com

more details). This corpus was regenerated from the original XML files. We extracted the corpus based on the 132 moods classes according to corpus definition.

Feature Extraction and Feature Selection for Classification

We have used the features from (Mishne, 2005). We used similar features to those used by (Mishne, 2005), with some extensions. The following features are used from our corpora dataset.

- ***Frequency Counts***: Bag-of-Words (BoW) is the most common feature representation used in automatic text classification. We represent the words by their frequencies.

- ***Length-related Features***: Since, blog posts vary in length, we consider length features such as: the length of the document, the number of sentences, and the average number of words.

- ***Part-of-Speech tags***: We use both sequence of Part-of-Speech (PoS) tags and individual PoS for verbs, nouns, adjectives and adverbs.

- ***Special Symbols***: We used special symbols called emoticons (emotional icons), that represent human emotions or attitudes. We used 9 most popular emoticons as features.

- ***Sentiment Orientation Features***: For emotion and mood classification, the sentiment orientation of some words can be a useful feature. Several sources are predictors for sentiment orientation. We calculate the total and the average orientation score for each document based on the words that are from the following resources: General Inquirer (Stone et al., 1966), Kim-Hovy list of opinion-mining words (Kim and Hovy, 2004), and (Turney and Littman, 2003).

We explain in detail these features and the way that we use them in our research in Chapters 3 and Chapter 4.

Classification based on Hierarchical Approach

We proposed a Hierarchical Approach for classification of moods and emotions. We classified our data in hierarchical levels (5 Levels) for 132 moods. Our results show that this method works well (see Chapter 3).

Text Categorization by Emotions and Extracting Templates for Emotions

We applied our hierarchical method to the classification of six basic emotions (happiness, sadness, fear, surprise, disgust and anger). Then we used the features that we extracted from this method to extract templates for emotions. By this, we learned that many features can apply to different emotions. We called these features cross-emotion features. We used these significant features in our paraphrase extraction task.

Paraphrase Extraction for Emotions

Paraphrasing is one of the crucial tasks in natural language understanding and generation. We introduced a novel technique of extract paraphrases for emotion terms, from non-parallel corpora. We present a bootstrapping technique for identifying paraphrases, starting with a small number of seeds. WordNet Affect emotion words are used as seeds. The bootstrapping approach learns extraction patterns for six classes of emotions. We use annotated blogs and other datasets as texts from which to extract paraphrases, based on the highest-scoring extraction patterns. The results include lexical and morpho-syntactic paraphrases, that we evaluate with human judges. We discuss this in Chapter 4.

Authoring Tool for Generation of Templates

Natural Language Generation (NLG) systems can make data accessible in an easily digestible textual form; but using such systems requires sophisticated linguistic and sometimes even programming knowledge. We have designed and implemented an environment for creating and modifying NLG templates that requires no programming knowledge, and can operate with a minimum of linguistic knowledge. It allows specifying templates with

any number of variables and dependencies between them. It internally uses SimpleNLG to provide the linguistic background knowledge. We tested the performance of our system in the context of an interactive simulation game (Caropreso et al., 2009a,c,b).

The purpose of this research was to study how we can use NLG system to generate text without programming knowledge. Secondly, since there are many NLG systems in literature, we needed to decide which system is the best to use in our research. Also, we needed a system that allows us to use both linguistic background knowledges and has capability toward template-based model for generation. To obtain this, we have done a lot of study and we decide to use the SimpleNLG system (Gatt and Reiter, 2009). In our research we used SimpleNLG library to do this task. We explained the ability and advantages of SimpleNLG and we describe how we used SimpleNLG in our research in Chapter 5.

1.2.3 Generating Text to Express Emotions

Natural Language Generation (NLG) can be viewed as the inverse of Natural Language Understanding (NLU), as NLG maps from meaning to text, while NLU maps from text to meaning. An NLG system will achieve its goal by performing different tasks such as selecting terminology and producing grammatically-correct sentences. It will go through several stages in order to generate text which looks natural (similar to text that would be generated by a human being to express the given concepts).

There are two widely adopted approaches to NLG, the "deep-linguistic" and the "template-based" (Gagn and Briggs, 1997). The "deep-linguistic" approach attempts to build the sentences up from a logical representation. The "template-based" NLG systems provide scaffolding in the form of templates that contain a predefined structure and perhaps some of the final text. The "deep-linguistic" approach to NLG is designed to be flexible and should be able to express any sentence, given a valid input logical form. Wide adoption of these systems has been constrained by the sophistication of the grammar system required, and the steep learning curve for writing logical forms. An example of this type of system is KMPL (Bateman, 1997).

We decided to use the template-based approach, while incorporating as much linguistic knowledge as needed. We use SimpleNLG (Gatt and Reiter, 2009) for sentence realization; it allows the user to specify a sentence by giving its content words and their grammatical roles (such as subject or verb). SimpleNLG is implemented as a Java library and it requires Java programming knowledge to be used. SimpleNLG automates several tasks, such as orthography, morphology, and grammatical realization. We will discuss this NLG system and other systems in detail in Chapter 2.

1.3 Contributions

During the research toward this thesis, we have made several contributions to emotion analysis and generation. Our methods are domain-independent and robust. The contributions are listed below:

1. **Proposed a hierarchical approach for mood classification, using sentiment orientation features:**

   We have tested how the sentiment orientation features and the open class lexical terms (adjectives, nouns, verbs and adverbs) help in classification of texts by emotions. We proposed a new method for classifying text by mood and emotion based on a hierarchical approach that works better than a flat classification approach. Based on the hierarchical classification, we constructed a domain of valuable features which can be used in emotion analysis and generation in NLG.

2. **Designed an Algorithm to Extract Paraphrases for Emotions**

   We based our method for paraphrase extraction on the assumption that phrases which appear in similar contexts are paraphrases. Here, contexts are groups of the words that surround a phrase. The bootstrapping algorithm learns which contexts are good predictors of paraphrases by analyzing the contexts surrounding identical words in aligned contexts. These contexts are used to extract new paraphrases, which in turn are used to learn more contexts. Our algorithm produces phrasal

and single word lexical paraphrase, as well as syntactic paraphrases. Our results show that the algorithm extracts paraphrases with high accuracy.

3. **Automatic Generation of Narrative Content for Digital Games**
   We developed an NLG system, for interactive simulation games that is used for training (Caropreso et al., 2009c,a). This kind of system usually requires a large amount of coherent narrative content. An effective and efficient solution to the narrative content creation problem is to use NLG techniques. The use of NLG systems, however, requires sophisticated linguistic and sometimes programming knowledge. For this reason, NLG systems are typically not accessible to game designers who write the narrative content of the games. We have designed and implemented a visual environment for creating and modifying NLG templates that requires no programming knowledge, and can operate with a minimum of linguistic knowledge. It allows specifying templates with any number of variables and dependencies between them. It automatically generates all the sentences that follow the created template. Our NLG system uses SimpleNLG (Gatt and Reiter, 2009) to provide the linguistic background knowledge. We tested the performance of our system in the context of an interactive simulation game. We have identified the need for an NLG Template Authoring Environment that allows game content designers without linguistic and programming background to experiment with and finally design language templates.

4. **Pattern-based Model to Generate Text with Emotions**
   We developed a pattern-based model for generating emotion sentences. Our model extracts syntactic patterns from the data set, then it extends these patterns to be good candidates for sentence generation. This was done by tokenization, extracting patterns, and analyzing the patterns. Then we developed a Sentence Planning module which provides rules and constraints for our model. Also, we developed a Surface Realization module that produces final output text. The realizer generates sentences according to sentence planner (microplanner) module. The realizer is

used to generate sentences from our pattern-based model.

1.4 Outline of this Thesis

This thesis is composed of 7 chapters. Chapter 1 introduces the problem we are addressing. Chapter 2 summarizes much of the work done on text analysis and generation toward emotion and NLG systems. Chapter 3 describes the feature extraction and classification for emotions and moods. In this chapter we introduce the sentiment orientation features as well as the hierarchical approach to emotion classification. Chapter 4 outlines algorithm and experiments that we performed for paraphrases extraction and pattern identification for emotions. Chapter 5 covers Natural Language Generation, and our Template Authoring Environment Tool. In Chapter 6, we explain our approach and model for generating text to express emotions (using our patterns, or using our Authoring Tool and the extracted paraphrases). Chapter 7 concludes this thesis and proposes avenues for future work.

Portions of this thesis have been published in Keshtkar and Inkpen (2011c), Keshtkar and Inkpen (2011b), Keshtkar and Inkpen (2011a), Keshtkar and Inkpen (2010a), Keshtkar and Inkpen (2010b), Keshtkar and Inkpen (2009), Inkpen et al. (2009), Caropreso et al. (2009a), Caropreso et al. (2009b), and Caropreso et al. (2009c).

Chapter 2

Literature Review

In this chapter we describe previous research, focusing on two aspects that are related to our research, namely 1) sentiment, emotion, and mood analysis and classification, and 2) Natural Language Generation systems. We also discuss some related work in more detail in the later chapters when it is relevant to specific problems that we consider in our research.

2.1 Sentiment Analysis and Classification

Many researchers have become interested in sentiment analysis, as more people learn of the scientific challenges posed, and the scope of new applications enabled, by the processing of subjective language. The papers studied by Qu et al. (2004) and Esuli (2006) are a relatively early representative sample of research in the area. In this section, we review a range of methods, algorithms, or approaches, in sentiment analysis tasks.

In general, automatic sentiment classification can focus on words, sentences, or documents. There are two approaches of sentiment classification methods for documents: *lexicon-based* and *corpus-based*. *Lexicon-based* methods compute a sentiment score for texts, according to the scores of the words in the texts that are also in a sentiment lexicon. Turney (2002) estimated the sentiment orientation of customer reviews using the semantic orientation scores of the constituent adjectives. The orientation of the adjec-

tives was measured by their co-occurrence frequency on the Web with several positive or negative seed adjectives. Kim and Hovy (2004) assigned polarity scores to a large list of words, based on their WordNet distance from positive and negative seed words. Hiroshi et al. (2004) used deep language analysis techniques for machine translation to extract sentiment scores for words in documents. Kennedy and Inkpen (2006) determined the sentiment of customer reviews by counting positive and negative terms, taking into account contextual valence shifters, such as negations and intensifiers. A computable metric of positive or negative polarity in financial news text was proposed by (Devitt and Ahmad, 2007).

Corpus-based methods use a corpus of documents that are labelled with polarity to train a sentiment classifier. Various classification models and linguistic features were introduced to improve the classification performance: Pang and Lee (2008), Mullen and Collier (2004), Wilson et al. (2005), Read (2005). McDonald et al. (2007) explored a structured model for jointly classifying the sentiment of text at different levels of granularity. Blitzer et al. (2007) investigated domain adaptation for sentiment classifiers, focusing on online reviews for different types of products. Andreevskaia and Bergler (2008) integrated a corpus-based classifier with a *lexicon-based* classifier, using precision-based weighting, in order to increase classification performance across domains.

Several researchers focused on leveraging rich English resources for sentiment analysis into other languages. For the Romanian language, Banea et al. (2008b) applied standard Naive Bayes and SVM classifiers to subjectivity classification, and the results indicated that automatic translation is viable for the construction of resources and tools in a new language. Wan (2008) used both Chinese and English lexicons to improve Chinese sentiment analysis. Work on Chinese sentiment classification uses *lexicon-based* or *corpus-based* methods, similar to those described above for English (Tsou et al., 2005; Ye et al., 2006; Li and Sun, 2007).

However, according to Pang and Lee (2008), the sentiment analysis and classification of text mainly focuses on *polarity classification*, *opinion mining and summarization* and *subjectivity detection*. A brief explanation of these follows.

2.1.1 Polarity classification

As mentioned, in the past many researchers were attracted by sentiment polarity classification, which has been focused on document classification to identify authors' sentiment expression as positive or negative.

For example, categorization of entire documents based on sentiment has been done by Hearst (1992); Sack (1994). Huettner and Subasic (2000); Das and Chen (2001); Tong (2001) proposed construction of discriminant-word lexicons (manual or semi-manual). On the other hand, Das and Chen (2001) presented a methodology for real time sentiment extraction in the field of finances; Web-based stock message boards, attempts to automatically label each message as a "buy", "sell", or "neutral" recommendation. Das and Chen (2001) classifier achieves an accuracy of 62% and the human agreement rate was 72%. In contrast, with automatically labeled data collected from the web, Pang and Lee (2008) addressed the task of determining sentiment polarity in movie reviews via supervised learning approaches.

2.1.2 Subjectivity Detection

Research in polarity classification usually considers the input documents to be opinionated. In most of the applications they might want to know whether a given document contains subjective information or to determine which portions of the document are subjective. Research in this area by Hatzivassiloglou and Wiebe (2000) investigated the effects of adjective orientation on sentence subjectivity. Their goal was to establish whether a given sentence is subjective or not by considering the adjectives used in the sentence.

Much research addresses sentence-level or sub-sentence-level subjectivity detection in different domains (Wiebe et al., 2001; Wiebe and Wilson, 2002; Yu and Hatzivassiloglou, 2003; Riloff and Wiebe, 2003; Pang and Lee, 2004; Beineke et al., 2004; Kim and Hovy, 2005a; Wilson et al., 2005). A comprehensive examination of the use of different clues and features for recognizing subjectivity in text was also presented by Wiebe et al.

(2005).

2.1.3 Summarization and Opinion Mining

Readers might not be interested in off-topic passages in opinion documents. To address this Hurst and Nigam (2004) introduced a sentence-level classifier to extract sentiment polarity on a given topic, in order to identify topical sentences. For reviews covering several aspects aspects of the subject matter, it may be useful to summarize the opinions grouped by the specific aspects addressed. Work has been done to explore identifying features and opinions associated with these features from product reviews (Yi et al., 2003; Hu and Liu, 2004; Kobayashi et al., 2004; Popescu and Etzioni, 2005; Yi and Niblack, 2005).

2.1.4 Perspectives and Viewpoints

Some research in non-factual-based text analysis deals with perspectives and viewpoints (Wiebe and Rapaport, 1988; Wiebe, 1994). The MPQA corpus contains annotations of the subjective opinions and emotion in language, and a system that can be trained to detect such expressions would be able to answer questions about opinions. Low-level opinion annotations were developed and evaluated, which facilitated the study of a number of interesting problems, such as identifying the opinion holder and analyzing opinions at phrase level (Cardie et al., 2003; Stoyanov et al., 2004; Choi et al., 2005; Wilson et al., 2005; Kim and Hovy, 2005b). There has also been work that focused on specific pairs of perspectives such as, identifying Israeli versus Palestinian viewpoints (Lin et al., 2006).

2.1.5 Other Non-Factual Information in Text

Work has been done to explore identifying features and opinions associated with these features from product reviews identifying the genre of text that addresses non-topic-based categorization is another related area of research in sentiment analysis (Karlgren and Cutting, 1994; Kessler et al., 1997; Stamatatos and Kokkinakis, 2000; Finn et al.,

2002; Lee and Myaeng, 2002; Finn and Kushmerick, 2006). Subjective genres, such as "editorial", are one of the categories in this area; these are more closely related to subjectivity detection. There has also been research concentrating on classifying documents according to their source or source style, with statistically-detected stylistic variation (Koppel et al., 2002a) providing important cues; authorship identification is perhaps the most significant. A classic example is a Bayesian study of the authorship of the Federalist Papers (Mosteller and L.Wallace, 1984).

Since sentiment analysis reaches deeper into Natural Language Processing (NLP), we briefly mention other recent applications of sentiment analysis for: question answering (sentiment questions) (Yu and Hatzivassiloglou, 2003; Kim and Hovy, 2004; Stoyanov et al., 2004) determining public opinions through blogs (for Business Intelligence) (Durant and Smith., 2006). (Koppel and Shtrimberg, 2004), facilitating the consumer decision process (Turney, 2002; Pang and Lee, 2008; Turney and Littman, 2003; Kushal et al., 2003; Pang and Lee, 2004), and quantifying subjective attitudes encoded in free texts (ĎMello et al., 2005). Currently, most techniques for detecting emotions focus on physiological and visual cues (Kaliouby, 2005). Polanyi and Zaenen (2004) contributed the Missing Definitive Affect Lexicon, which identifies the most salient clues about attitude provided by the lexical choices of the author.

2.1.6 Emotion Lexicons

General Inquirer Dictionary

Grefenstette et al. (2004) conducted an evaluation of the coverage of the Harvard General Inquirer, a well-known affect dictionary. From this analysis we can conclude that the definitive affect lexicon has not yet been created and that there is room for improvement in all existing lexicons. The General Inquirer Dictionary was created by Stone et al. (1966), and in 2004 the dictionary contained 1,915 words marked as generally positive and 2,291 as negative. These words were marked with the affect classes: (i.e Active, Passive, Strong, Weak, Pleasure, Pain, Feeling (other than pain/pleasure), Arousal, Virtue, Vice,

Overstated, or Understated). In addition to these labels, an open-ended set of semantic labels has been defined, including Human, Animate, Region and Route. These semantic labels are of limited use for affect identification, though they may be helpful for genre specific studies. Promising Affect Lexicon Generation Initiatives, (Hatzivassiloglou and McKeown, 1997; Wiebe et al., 2005; Wilson et al., 2005) focused on the generation of sentiment-charged words rather than the analysis of sentiment.

WordNet-Affect

Strapparava and Valitutti (2004) presented a linguistic resource for a lexical representation of affective knowledge, which they called WordNett-Affect. They developed this from WordNet, through the selection and labelling of the *synsets* representing affective concepts.

In their research on affective lexicon, they focused on producing a resource that contains a set of affective concepts correlated with affective words. They used WordNet database (Felbaum, 1998), an important resource to begin with. The *synset* model is simple enough to provide an intrinsic correlation between a concept and the corresponding words. We explain how we used this important lexicon in our research in Chapter 4.

LIWC: Linguistic Inquiry and Word Count

How individuals talk and write provides a view of their emotional and cognitive worlds (Lepore and Smyth, 2002). Many studies have found that having individuals write or talk about deeply emotional experiences is associated with improvements in mental and physical health (Lepore and Smyth, 2002). Text analysis based on these studies indicates that those who benefit most from writing tend to use relatively high rates of positive emotion words, a moderate number of negative emotion words, an increasing number of cognitive words, and they switch their use of pronouns from writing session to writing session (Campbell and Pennebaker, 2003).

Pennebaker et al. (2007) developed a text analysis application called Linguistic Inquiry and Word Count (LIWC), to provide an efficient and effective method for studying

the various emotional, cognitive and structural components present in individuals' verbal and written speech samples. The first LIWC dictionary was produced as part of an exploratory study of language and disclosure. The second version (LIWC2001) updated the original with an expanded dictionary and modernized software design. The latest version (LIWC2007) has significant revisions in both the dictionary and the software options and it supports multiple languages.

The selection of words in LIWC2007 involved a number of steps over several years. The general idea was to determine a group of words that convey basic emotional and cognitive dimensions for studies in social, health, and personality psychology. The domain of LIWC2007 categories expanded in the following steps: Step 1: Word Collection, Step 2: Judges' Rating Phases, Step 3: Psychometric Evaluation, and Step 4: Updates and Expansions.

Emotions Lexicon by using Mechanical Turk

The Amazon Mechanical Turk (MTurk)[1] is one of the suites in Amazon Web Services, a crowd-sourcing Internet marketplace that enables researchers (known as Requesters) to coordinate the use of human intelligence to perform tasks which computers are incapable of. Recently, Mohammad and Turney (2010) showed how using the Mechanical Turk, they were able to create a high-quality, moderate-sized emotion lexicon. They also showed how the inclusion of word choice questions can discourage malicious data entry, help identify instances where the annotator may not be familiar with the target term (allowing to reject such annotations), and help obtain annotations at sense level (rather than at word level). They perform an analysis of the annotations to better understand the distribution of emotions evoked by terms of different parts of speech. They identified which emotions tend to be evoked simultaneously by the same term and showed that certain emotions indeed go hand in hand.

[1] https://www.mturk.com/mturk/welcome

2.2 Affect and Emotions

People have considered various affect types and emotions in their research. For example, the six "universal" emotions (Ekman, 1992): *anger, disgust, fear, happiness, sadness*, and *surprise* were used in emotion classification ((Subasic and Huettner, 2001; Liu et al., 2003b; Alm et al., 2005), as well as in computational approaches to humour recognition and generation (Mihalcea and Strapparava, 2006). Many interesting aspects of texts, such as 'emotion' and 'mood', are also being explored in the context of weblogs and other informal text resources (Nicolov et al., 2006).

2.2.1 Dimensional and Hierarchical Structure of Affect

Researchers like Green et al. (1993) challenged the idea that *"positive affect"* and *"negative affect"* (PA and NA) are huge, uncorrelated dimensions. They claimed that positive and negative emotional activation, such as happiness and sadness, have a "largely bipolar structure of affect," when random and non-random errors (positive vs negative) are taken into account. Tellegen et al. (1999) believed that "happiness" and "sadness" have a vast unidimensional bipolar structure, but PA and NA are relatively independent. Their analyses show a three-level hierarchy incorporated in one structure: 1) a general bipolar happiness-versus-unhappiness dimension, 2) the relatively independent PA and NA dimensions a level below, and 3) discrete emotions. They hope that psychologists will finally take the hierarchical affect models into account, and exploit their heuristic value. For example, a hierarchical structure allows researchers to determine whether, from an affective perspective, more than a general happiness-versus-unhappiness index is required to account for life satisfaction. Also, Watson and Tellegen (1985) constructed a two-dimensional map (see Figure 2.1), which presents PA-NA, Pleasantness-Versus-Unpleasantness and Engagement-Versus-Disengagement (or high-versus-low Activation or Arousal) dimensions. Figure 2.1 shows the variety of emotions; some of them can be found in the LiveJournal hierarchy of moods.

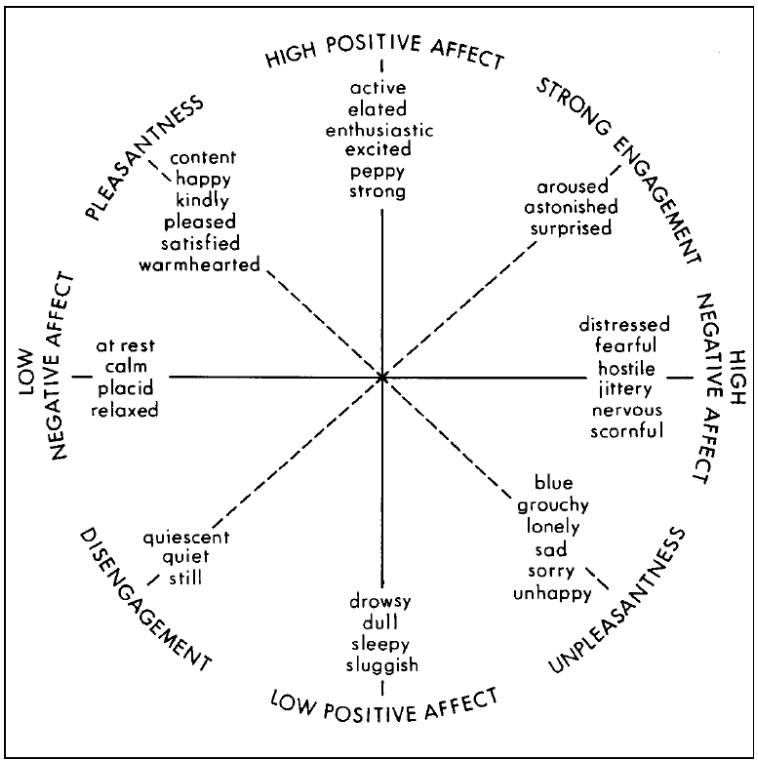

Figure 2.1: Two-dimensional map of emotion by Watson and Tellegen (1985).

2.3 Manual Identification of Emotions in Text

2.3.1 Annotating Expressions of Opinions and Emotions in Language

Read (2004) studied emotion in private states in a broader framework, and more research on private states was done by Wiebe et al. (2005). A private state frame includes the source of the private state (i.e., whose private state is being expressed), the target (i.e., what the private state is about), and various properties involving intensity, significance, and type of attitude (Wiebe et al., 2005). They investigated whether previous research, such as that of (Pang and Lee, 2004; Turney and Littman, 2003; Riloff and Wiebe, 2003; Hatzivassiloglou and Wiebe, 2000), focused on sentiment or subjectivity classification at the document or sentence level. Document classification tasks include, for example, distinguishing editorials from news articles, and classifying reviews as positive or negative.

However, for many applications, identifying only opinionated documents or sentences may not be sufficient.

Wiebe et al. (2005) studied 10,000 news article sentences from the MPQA (Multi-Perspective Question Answering) corpus, and manually-annotated them for emotions, opinions, sentiments, speculations, evaluations and other private states in language. The resulting corpus annotation scheme is described, with examples of its use. The main goal of Wiebe et al. (2005) was to investigate the use of opinion and emotion in language through a corpus annotation study. To achieve this, they proposed a detailed annotation scheme that identifies key components and properties of the above (emotions, opinions, sentiments, speculations, evaluations and other private states). Read (2004) found that internal states cannot be directly observed by others. Wiebe et al. (2005) also proposed a fine-grained annotation scheme, annotating text at the word and phrase-level, rather than the level of document or sentence. They defined a private state frame for every expression of private state in each sentence.

2.3.2 Affective Text Dataset with Annotated Emotions

The Text Affect dataset (Strapparava and Mihalcea, 2007) was used in the SemEval 2007-Task 14. A test set of 1,000 newspaper headline annotations were released after the workshop. In task 14, Strapparava and Mihalcea (2007) presented an 'Affective Text' task, which focused on the classification of emotions and valence (positive/negative polarity) in news headlines, and was intended as an exploration of the connection between emotions and lexical semantics. They prepared a corpus consisting of news titles extracted from news websites (e.g. Google news, CNN) and/or newspapers, and 1,000 titles of web sites. Objective: they provided a set of six predefined emotion labels (i.e. Anger, Disgust, Fear, Joy, Sadness, Surprise) and had six human judges annotate the titles with the appropriate emotion label, and/or with a valence indication (positive/negative). In their paper, they described the dataset used in the evaluation, and the results obtained by the participating systems. A development dataset of 250 newspaper headlines were also annotated by the six annotators. The agreement between the annotators (correla-

EMOTIONS	Agreement
Anger	49.55
Disgust	44.51
Fear	63.81
Happiness	59.91
Sadness	68.19
Surprise	36.07
VALENCE	
Valence	78.01

Table 2.1: Pearson correlation for inter-annotator agreement (Strapparava and Mihalcea, 2007).

tion) ranged from 0.36 to 0.68 for the six emotions annotated with intensity levels. This dataset used the six classes of basic emotions proposed by Ekman (1992).

Unlike previous annotations of sentiment or subjectivity, e.g., (Wiebe et al., 2005; Pang and Lee, 2008), which typically relied on binary (subjective/objective or positive/negative) annotations, Strapparava and Mihalcea (2007) decided to use a finer-grained scale which allowed the annotators to select different degrees of emotional load. They conducted inter-tagger agreement studies for each of the six emotions, and for the valence annotations. The agreement evaluations were carried out using the Pearson correlation measure (see Table 2.1). To measure the agreement among the six annotators, they first measured the agreement between each annotator and the average of the remaining five annotators, followed by an average over the six resulting agreement figures.

Based on the results, they concluded that indicating the emotion annotation is difficult. The gap between the results obtained by the systems, and the upper bound represented by the annotator agreement, suggests that there is room for future improvements.

2.3.3 Annotating Expressions of Emotion in Text

Aman and Szpakowicz (2007) attempted to study expression of emotions in text. They conducted an emotion annotation task to identify emotion category, emotion intensity and the words/phrases that indicate emotion in text. They computed the annotation agreement in a corpus of blog posts. The average agreement for labelling sentences as emotion or non-emotion was 0.76. The agreement on emotion categories was between 0.60 and 0.79, and for emotion indicators it was 0.66. Aman and Szpakowicz (2007) produced 2,090 sentences annotated with the same six emotions (i.e. anger, disgust, fear, joy, sadness and surprise), from 173 weblog posts. Their automatic classification experiments of emotion or non-emotion class achieved an accuracy of 73.89%, significantly higher than the baseline of the most frequent class. Their experiments focused on binary classification: whether a sentence contains an emotion or not.

2.3.4 Annotation of Emotional Content in Text

Alm et al. (2005) used the nine basic emotions proposed by Izard (1971) : happiness, interest, shame, guilt, sadness (distress), anger, disgust, surprise and fear. In their paper, they discuss the emotional distributions in child-directed texts. They provided statistical evidence for the relevance of emotions and evaluated trends of emotional story development, based on the annotation statistics of 22 Grimm brothers fairy tales, which are part of a larger, on-going text-annotation project that was also introduced. The study is motivated by the need to explore features for text-based emotion prediction at the sentence-level, for use in expressive text-to-speech synthesis of children's stories.

Their text-annotation project targets the annotation of emotional contents (i.e., the six emotions of Ekman) in three sets of children's stories, by Beatrix Potter, the Grimm brothers and Danish author H. C. Andersen.

Table 2.2 presents inter-annotator statistics for a preliminary data set. The two annotators from Group A addressed a set of 20 Grimm brothers stories (1,357 sentences), whereas Group B annotated a different set of 22 Grimm brothers stories (1,581 sentences).

	Kappa b	$P(b_{i=j})$	$P(m_{i=j})$	$P(t_{i=j})$
Gr A	0.51	0.64	0.73	0.76
Gr B	0.24	0.45	0.49	0.50

Table 2.2: Inter-annotator statistics for 2 groups of paired annotators.Group A's $a_i(i = 1, 2)$ and group B's $b_i(i =, 1, 2)$ annotated 20 and 22 Grimm brothers stories, respective (Alm et al., 2005)

The kappa statistics are reported for inter-annotator agreement, as well as the percent overlap between the paired annotators. For the latter measure, classes were also combined into their semantically intuitive super-classes, where t refers to the top level (whether a sentence is neutral or non-neutral), m to whether an emotional sentence has neutral, positive or negative valence, and b to the basic level emotion category (e.g., angry, happy), including neutral.

2.3.5 Affect Expressed for Online Communication Language

In another research project, the Affect Analysis Model was designed based on the compositionality principle by Neviarouskaya et al. (2007). They also used the nine basic emotions Izard (1971): *anger, disgust, fear, guilt, interest, happiness, sadness, shame* and *surprise*, as the basis for affective text classification. The proposed rule-based approach processes each sentence in sequential stages, including symbolic cue processing, detection and transformation of abbreviations, sentence parsing, and word/phrase/sentence-level analyzes, and the system showed promising results in affect recognition in sentences extracted from diary-like blog posts and fairy tales. However, the system has limitations, such as the strong dependency on the source lexicon and affect database.

2.4 Automatic Learning of Emotion from Texts

2.4.1 Learning to Identify Emotions in Text

In the SemEval 2007-Task 14, the three teams that participated used unsupervised methods for automatic classification of emotion on text and obtained low classification accuracy on the test set (Strapparava and Mihalcea, 2008). The SemEval dataset was annotated for the six basic emotions of Ekman (1992). More sophisticated unsupervised methods for this task were proposed by Strapparava and Mihalcea (2008). They evaluated several knowledge-based and corpus-based methods to automatically identify these emotions in the text. Mihalcea and Liu (2006) focused their work on two particular emotions: *happiness and sadness*. They worked on blog posts, which are self-annotated by the blog writers with happy or sad mood labels.

2.4.2 Automatic Emotion Classification

Aman and Szpakowicz (2007) studied fine-grained automatic classification of sentences on the basis of emotion categories. They focused on recognizing the emotional sentences in text, regardless of their emotion category. They extracted the sentences from the corpus for which there was consensus among the judges on their emotion category (i.e. the annotation data described in 2.3.3). They then assigned all emotion category sentences to a class entitled 'EM', while those with no emotion were assigned to class 'NE'. The resulting dataset had 1,466 sentences in the EM class, and 2,800 in the NE class. In their automatic classification experiments, Aman and Szpakowicz (2007) achieved an accuracy of 73.89%. They focused only on binary classification (i.e. whether a sentence contains an emotion or not).

2.4.3 Machine Learning for Text-based Emotion Prediction

Another research effort that focused on learning emotions from text is Alm et al. (2005). They investigated automatic classification of sentences in children's fairy tales, using the

nine basic emotions of Izard (1971). They used the manually annotated emotion dataset of stories, and the accuracy of test was $[60.05\% - 63.45\%]$. Read (2004) also has used a corpus of short stories, manually annotated with sentiment tags, in automatic emotion classification of sentences.

2.4.4 Textual Affect Sensing using Real-World Knowledge

In other related research, Liu et al. (2003a) utilized real-world knowledge about affect, drawn from a common-sense knowledge base. Their aim was to learn the semantics of text, in order to identify emotions at the sentence level. They started by extracting sentences from the knowledge-base that contain some affective information. The information was used to build affective models of text, which were then used to label each sentence with a figure representing one of Ekman's six basic emotions.

2.4.5 Mood Classification in Blog Corpus

Researchers have studied classification in blog corpora, such as Mishne (2005) and Jung et al. (2006). We explain these studies in Chapter 3, and compare them to our work.

2.5 Natural Language Generation Systems

A Natural Language Generation (NLG) system will achieve its goal by performing different tasks, such as selecting terminology and producing grammatically correct sentences. It must go through several stages in order to generate text which seems natural (i.e. similar to text that would be generated by a human being to express the given concepts).

According to (Reiter and Dale, 2000; Dalianis, 1996), the stages of an NLG system are:

- Content determination: Choosing what concepts to express.
- Lexicalization: Choosing words to express the concepts.

- Syntactic and morphological realization: producing the surface document or text by using syntactic and morphological rules.
- Sentence aggregation: Merging similar sentences into one sentence.
- Referring expression generation: Using pronouns to replace repeated noun phrases.
- Orthographic realization: Resolving matters such as formats, casing, and punctuation

Dalianis (1996) also introduced a two-stage NLG architecture in which, (1) the Discourse Planner chooses the appropriate content to express the communicative goal, and gathers information from a knowledge base to transform the content into text and, (2) the Surface Realizer generates sentences for the discourse specification based on its lexical and grammatical resources. The architecture of an NLG system is shown in Figure 2.2.

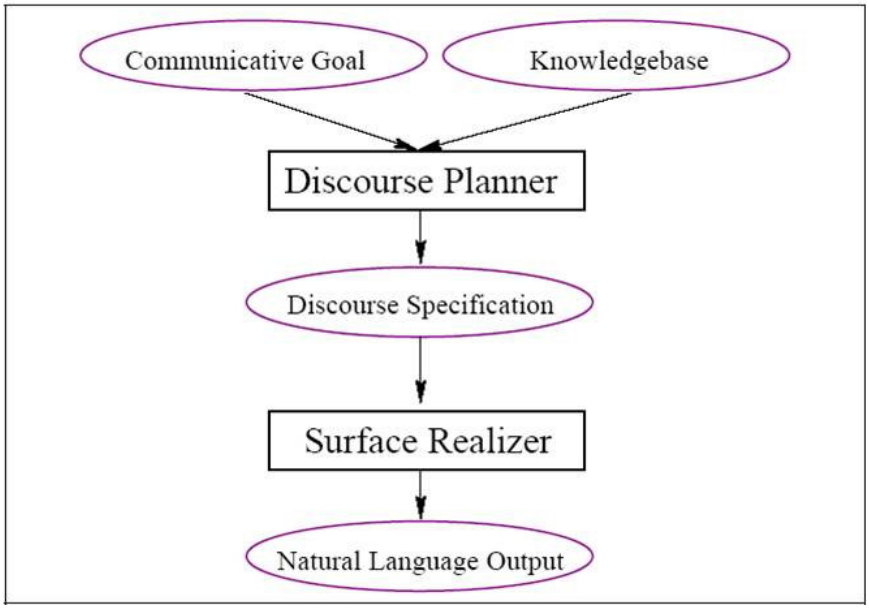

Figure 2.2: Architecture of NLG systems which introduced by Dalianis (1996).

2.5.1 NLG vs Human-Written Texts

Human variation may be an opportunity for NLG systems, because they can guarantee consistency. In weather forecasting, for example, users can receive texts written by several forecasters, which means that they may have problems reliably interpreting phrases such as "by evening", because of geographical area. In contrast, an NLG system could be programmed to use this phrase consistently. An NLG system could also be programmed to avoid idiosyncratic terms which users might not be familiar with (e.g, "bathtub"), and not to use terms in cases where people disagree about their applicability. Reiter and Sripada (2002) results show that it is not always easy for human writers to follow such consistency rules, particularly when they have limited time.

Interaction seems to be a key aspect in the process of human agreement on word usage (Garrod and Anderson, 1987). A small group of people who constantly communicate with each other over a long period will likely agree on the meaning of most words. However, these days human writers commonly write documents for people they have never met or otherwise interacted with. This reduces the effectiveness of the natural interaction mechanism for agreeing on word meanings.

In summary, dealing with lexical variation among human readers is a challenge for NLG systems, and will undoubtedly require a considerable amount of thought, research and data collection. But if NLG systems could eventually do a good job, they might produce texts superior to many human writers, which would greatly enhance the appeal of NLG technology.

2.5.2 Existing Natural Language Generation Tools

To our knowledge, there is currently no NLG system for the generation of emotional text in the literature. However, in this section we present some examples of existing NLG systems.

SKILLSUM: an NLG System for low-skilled readers

SKILLSUM (Williams and Reiter, 2006)) is a web-based application which integrates basic skills testing and generates feedback reports. SKILLSUM users are adults over 16 years of age with low basic reading skills. They test their literacy or numeracy by completing a short screening test of no more than 27 multiple-choice questions. They are then shown a personalized report, which is generated by the SKILLSUM's NLG system. Figure 2.3 shows the architecture of SKILLSUM's NLG module. The module is based on pipeline architecture, and uses the same three sequential processes as many NLG systems (i.e. document planning, micro-planning and realization), described in Reiter and Dale (2000). SKILLSUM's application domain is education. There are other NLG applications in this domain, including intelligent tutoring systems like that of (Moore et al., 2004), which is an interactive dialogue system. However, these systems address a student's immediate difficulties with a task, whereas SKILLSUM summarizes the student's overall skills

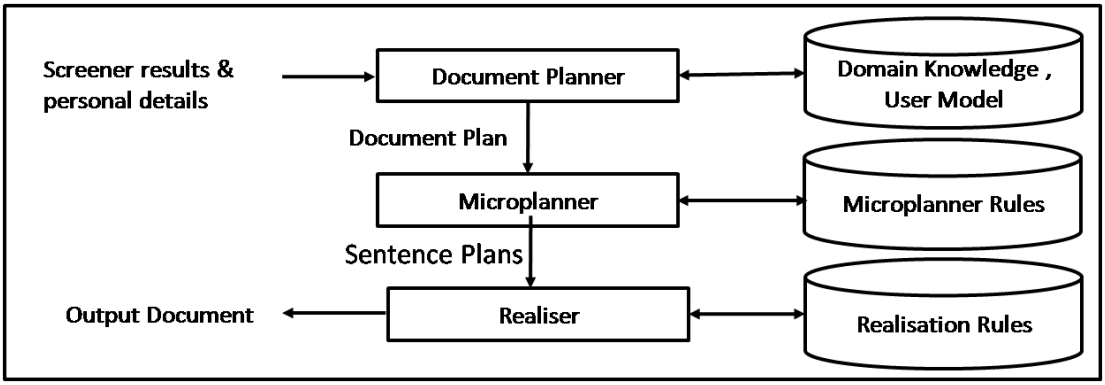

Figure 2.3: The Architecture of SKILLSUMs NLG Module (Williams and Reiter, 2006).

SimpleNLG

SimpleNLG (Gatt and Reiter, 2009) can be used to write a program which generates grammatically correct English sentences. It is a library (not an application) written in Java, that performs the basic tasks necessary for natural language generation; it assem-

bles parts of a sentence into grammatical form and outputs the result. For example, it capitalizes the first letter of the sentence, can add an auxiliary verb and make it agree with the subject, or simply add "ing" to the end of the verb if the progressive aspect of the verb is desired. Briefly, SimpleNLG's three tasks are:

Grammar: SimpleNLG puts sentences into grammatical form (enforces noun-verb agreement, creates well formed verb groups, etc).

Morphology: SimpleNLG handles inflection (modifies words to reflect information such as gender, tense, number or person).

Orthography: SimpleNLG inserts the appropriate whitespace between the words of the sentence, puts a period at the end of the sentence, and formats lists such as "apples, pears, and oranges".

Based on these advantages, we use SimpleNLG (Gatt and Reiter, 2009) for sentence realization. The user can specify a sentence by giving its content words and their grammatical roles, while allowing parts of the sentence to be fixed templates (See Section 5.4.2 for more details).

RealPro and Examplars

RealPro and Examplars, from the CoGenText RealPro company (Rambow, 1997), are text generation "engines" that perform syntactic realization. For example, they can transform abstract syntactic specifications of natural language sentences (or phrases) into their corresponding surface forms. They support multiple languages and multiple levels of linguistic representation, and the performance is suitable for real-world applications.

KPML: Komet-Penman MultiLingual

Another approach to linguistic realisation for Natural Language Generation research is motivated by systemic functional linguistics (SFL) (Halliday, 1985). SFL is concerned with the functions of language; a systemic functional grammar describes how functions can be mapped into or expressed by surface forms. This view is very appropriate in the context of NLG.

Bateman (1997) described Komet-Penman MultiLingual (KPML) as a linguistic realizer based on SFL, which has been used in several NLG projects. The KPML provides a robust platform for large-scale grammar engineering that is particularly oriented to multilingual grammar development and generation. KPML is also used in multilingual text generation research, and for teaching.

SURGE

Elhadad and Robin (1996) introduced SURGE as an alternative to the SFL approach. SURGE is based on a unification-based systemically-oriented grammar of English. It uses the Functional Unification Formalism (FUF) (Elhadad, 1994) as its underlying mechanism.

Other NLG tools

Other NLG tools include:

HALogen (Langkilde-Geary and Knight, 2002): is a Natural Language Generation system developed at the USC Information Sciences Institute. The package consists of a symbolic generator, a forest ranker and some sample inputs. The symbolic generator includes the Sensus Ontology dictionary (which is based on WordNet). The forest ranker includes a 250 million word n-gram language model (i.e. unigram, bigram and trigram), trained on WSJ newspaper text. The symbolic generator is written in LISP, and requires a CommonLisp interpreter.

TG/2: Busemann (2005) is a template-based generator, with a shallow verbalizer that can be customized for new domains and tasks. It combines context-free grammar with templates and canned text in a single formalism.

2.5.3 Applications of Natural Language Generation

Another goal of NLG systems is to enable computers to present information to people in an understandable way. Internally, computer systems use representations which they can

easily manipulate, such as airline schedule databases, accounting spreadsheets, expert system knowledge bases, grid-based simulations of physical systems, and others. In many cases, however, these information representations require considerable expertise to interpret, so there is a need for systems which can present such data in an understandable form to non-expert users. When the best presentation of the data is in English, or some other human language, NLG technology can be used to construct the presentation system. Reiter and Dale (2000) presented a survey of NLG applications, and we briefly describe some of them here. NLG techniques can be used to:

- generate textual weather forecasts from representations of graphical weather maps, as in (Goldberg et al., 1994);
- summarise statistical data extracted from a database or spreadsheet, as in (Iordanskaja et al., 1992);
- explain medical information in a patient-friendly way, as in (Buchanan et al., 1995; Cawsey et al., 1995);
- describe a chain of reasoning carried out by an expert system, as in (Swartout, 1983);
- produce answers to questions about an object described in a knowledge base, as in (Reiter and Dale, 2000).

NLG technology can also be used to build authoring aids that can help people create routine documents. Many people spend much of their time producing documents, often in situations where document production is not their main responsibility. Doctors, for example, can spend a significant part of their day writing referral letters, discharge summaries and other routine documents, or a computer programmer might spend as much time writing text (code documentation, program logic descriptions, code walk-through reviews, progress reports, and so on) as writing actual code. Tools which help such people to quickly produce quality documents could considerably enhance both productivity and morale. Examples of using NLG as an authoring aid include:

- helping customer service representatives write letters for customers, as in (Springer et al., 1991)
- helping engineers produce management summaries of design paths they have explored, as in McKeown et al. (1995);
- helping personnel officers produce job descriptions, as in (Caldwell and Korelsky, 1995) and helping technical authors produce instructions for using software, as in (Paris et al., 1995).

2.5.4 Text-to-Text Generation

Content Models and Coherence Models

Barzilay and Lee (2004); Chen et al. (2009) and Barzilay and Lapata (2008) proposed a framework for representing and measuring local coherence. Their approach presented a novel Bayesian topic model for learning discourse-level document structure. They proposed a global model, in which both topic selection and ordering are biased to be similar across a collection of related documents. Their algorithm can automatically transform text into a set of entity transition sequences, and record distributional, syntactic and referential information about discourse entities. They also re-conceptualized coherence assessment as a learning task, and showed that their entity-based representation is well-suited for ranking-based generation and text classification tasks. They achieved good performance on text ordering, summary coherence evaluation and readability assessment. Figure 2.4 illustrates the Text-to-Text Generation Model introduced in Barzilay and Lapata (2008).

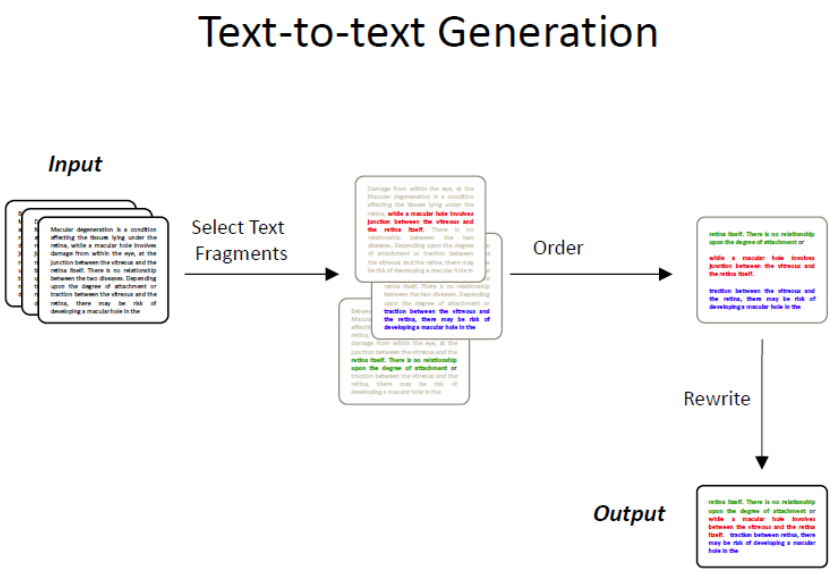

Figure 2.4: A Text-to-Text Generation Model (Barzilay and Lapata, 2008).

2.6 Summary

In this chapter, we explained previous research that is related to our work. We covered two areas. First we described sentiment, emotion, and mood analysis and classification, then we covered Natural Language Generation systems. In the first part, we explained the manual identification of emotion in text and the automatic detection of emotion. We showed which problems have been solved and which problems in emotion analysis still need effort to overcome. In the second part, we explored the most recent research in NLG system to find the best system that is suitable for our work. Our survey shows that very little research has been done in terms of generation of text with emotions. We found that SimpleNLG (Gatt and Reiter, 2009) is the one that suits our research, therefore we used SimpleNLG for surface realization tasks. We also discuss some related work in more detail in the later chapters, when it is relevant to specific problems that we consider in our research.

Chapter 3

Emotion and Mood Classification

3.1 Introduction

Weblog services are free Web sites that allow people to write and submit their opinions and thoughts regarding social, political and other events. On some of these sites, users can also indicate their emotional state at the time of writing, by choosing a mood label. One of the most popular weblog services which offers this feature is Livejournal[1]. Livejournal provides 132 mood identifiers in a drop box list and bloggers can use these to indicate their mood when they post a text or blog. Thus, they label their posts based on a set of possible moods or emotions.

In this chapter we address the task of classifying blog posts by mood. That is, given a blog post, we want to predict the most likely state of mind in which the post was written: whether the author was depressed, cheerful or bored, and so on. As in the majority of text classification tasks we take a machine learning approach, identifying a set of features to be used for the learning process.

There might be the case that the emotion of the writer is not the same as the emotion conveyed in the text. For example, happy people could write sad stories or report emotions of other people. It could be the case that a text belongs to several emotion classes. Nonetheless, in our corpus, each post is tagged by the user with one tag. We

[1] http://www.livejournalinc.com

assume, as it was done in previous work, that the tag is the current emotion of the text, with the risk to have several wrongly tagged posts in our corpus. However, since our corpus in very large (see Table 3.1 for some statistics of our corpus), we believe that this can be treated as noise in the data and it does not impact our evaluation.

Classifying texts by topic is a well-studied area (Sebastiani, 2002) as the words in the texts tend to be very good for characterizing topics. Classifying text by other stylistic properties is more difficult, since the documents can be about different topics in the same class, or about the same topic in different classes.

Classifications of texts by expressed sentiments or emotions are examples of non-topical classification. Automatic classification of blogs by mood is a similar but more challenging task, because of the colloquial language and because the texts tend to be short, which can cause feature vectors used for classification. The differences between subjectivity/sentiment/emotion classification, and classification of texts by mood, are the number of classes and the type of distinctions that need to be learned from the texts. The two classes targeted in subjectivity classification are objective and subjective texts. In sentiment analysis the classes are positive, negative and neutral. Several emotion classes are used in emotion analysis, while mood classification can include a very large number of classes.

A lot of research has been done in opinion and sentiment analysis by Pang and Lee (2008): we noted that work in Chapter 2. However, research on emotion and mood detection from texts is just beginning. Among the first preliminary investigations in emotion detection from texts, are the small scale experiments of Holzman and Pottenger (2003) and Rubin et al. (2004).

A few researchers have studied mood classification in blogs. Mishne (2005) collected a corpus of blog data annotated with mood labels, and then implemented a Support Vector Machine (SVM) classifier. The classification accuracy of this was only slightly above the baselines for both the training set and test set. Each dataset contained a number of positive examples for one mood, and the same number of negative examples, of other classes. Therefore the baseline of a random choice classifier was 50%.

Therefore, methods to increase the accuracy of the classification are required. Mishne (2005) used the top 40 moods as classes, whereas we have used all 132 classes and their hierarchical organization provided by Livejournal. Users are not aware of this hierarchical relationship between moods when they choose one. The moods are not shown in an hierarchy drop-down menu, but in a flat drop-down list.

In this chapter, we introduce a hierarchical approach to mood classification. We address the variety of possible mood labels, and our approach is flexible in the sense that we can change the set of moods (i.e., the classes in which we classify the blogs). We decided to employ a hierarchical approach, because we had such a large number of classes which were naturally grouped in a hierarchy. It makes sense for a classifier to learn the coarse-grained distinctions first, and learn the more fine-grained at subsequent levels in the hierarchy. We show that the results of the hierarchical classification are better than the results of a standard flat classification into 132 moods.

Our results also show that, although the blog corpus contains a large number of words in various domains, we can select features that lead to accurate classification of the expressed emotion and mood. We start with all the words as features (most of the words in the training data), then use feature selection techniques to reduce the feature space. We add sentiment orientation features and emoticons, and we show that they contribute significantly to improved classification.

The rest of the chapter is organized as follows. In Section 3.2 we describe the most recent related works. In Section 3.3 we examine the blog corpus that we consider for our research. In Section 3.4 we introduce our hierarchical classification approach. Section 3.5 follows with details regarding the features that we used for the classification process,divided into sets of related features. Our experiments and results are reported in Section 3.6, and we compare our work to existing work in affect analysis in Section 3.8. Section 3.9 concludes the chapter.

3.2 Related Work

In Chapter 2 we described the previous research into sentiment, mood, and emotion analysis in detail. In this section, we focus on research that has been done in mood classification divided into sets of features related to our research. In Section 3.8 we compare this research with our results.

3.2.1 Mood Classification in Blog Posts

Mishne (2005) presented preliminary work on classifying blog text according to the mood reported by the author during the writing. He used the LiveJournal data set, that contains a large collection of blog posts and online diary entries which have an indication of the writer's mood. He obtained modest but consistent improvements over a baseline, and their results indicate that increasing the amount of available training data will lead to an increase in accuracy. He also showed that though the classification accuracy is low, it is not significantly lower than human performance on the same task. Their main finding was that mood classification is a challenging task using current text analysis methods. We compare our results with Mishne (2005) in Section 3.8.

3.2.2 Hybrid Mood Classification Approach

Other research that investigated mood classification was done by Jung et al. (2006). They proposed a hybrid approach to identify mood in blog texts. In addition, they refined the mood corpus with a semi-automatic method to build and use it for evaluation in their system.

For their research they used ConceptNet (Liu and Singh, 2004) and the Affective Norms English Words (ANEW) (Bradley and Lang, 1999) list. To do classification they chose some unique features from blogs, and calculated statistical features such as term frequency, n-grams, and point-wise mutual information (PMI) for the SVM classification method. They also used paragraph-level segmentation based on mood flow analysis. They extracted their own blog data from LiveJournal.

Jung et al. (2006) used only four moods in their research:(*happy, angry, fear, sad*). We used their data set for these four moods, and incorporated two additional moods for our data set (*disgust, surprise*).

3.3 Data Set

We used the blog data set that Mishne (2005) collected for his research. The corpus contains 815,494 blog posts from Livejournal, the free weblog service used by millions of people to create weblogs. Figure 3.1 shows Web interface for the Livejournal blog service. The interface allows users to update their blogs, and includes input fields for the text of the post and the date, as well as an optional field for them to indicate their 'current mood'. To select their mood, users can choose from a list of 132 moods, or specify additional moods. We do not use the additional moods because very few posts are annotated with them . Statistics derived from this corpus are shown in Table 3.1.From the total posts, only 77% included an indication of mood; we did not consider the rest. On average there are 22 posts per blog. Table 3.2 shows the most frequent moods in the corpus, based on their frequency counts.

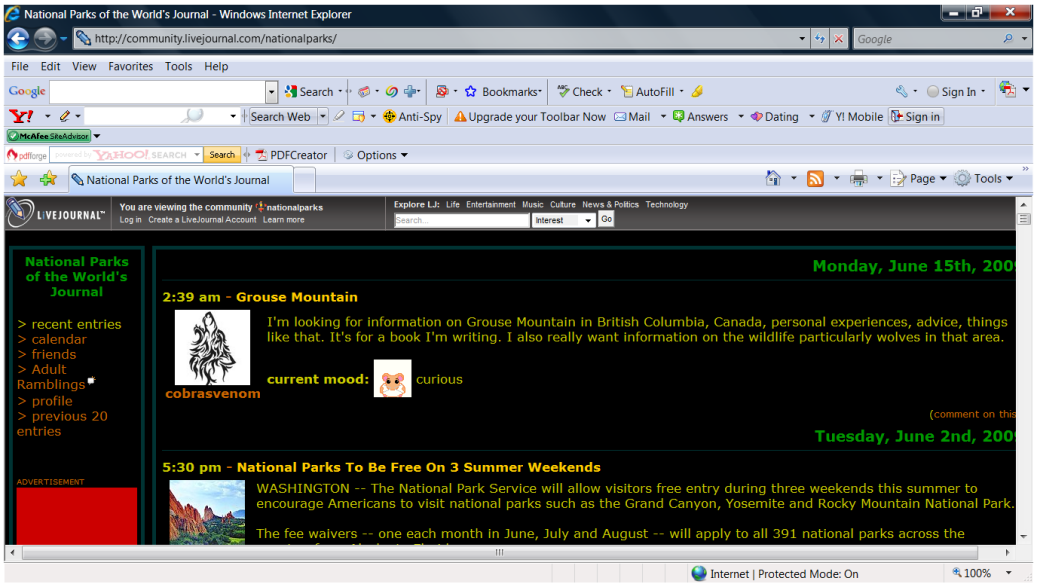

Figure 3.1: Example of blog postings in LiveJournal

Status	Counts
Moods that appeared once	46,558
Moods that appeared twice	4,279
Number of Standard Moods	132
Number of User-defined Moods	54,487
Total Words	69,149,217
Average-words/Post	200
Unique Words	596,638
Webpages per Mood	1000
Individual pages	122,624
Total Weblogs	37,009
Total Posts	815,494

Table 3.1: Statistics about words and posts in the data set.

In the Livejournal weblog service, moods are organized in a hierarchy, as shown in Figure 3.2. Therefore, we will use this hierarchy to select the classes for each classification level, in the method we propose in the next section.

It is notable that the mood "annotation" in this corpus is not always consistent. The blog users are different people, so obviously the points of resulting view from their moods are different. For example, a "frustrated" state for one person might be considered a different state by someone else; perhaps "anger" or "depression" or another category of emotion. We believe this is an advantage in our corpus. Unlike other corpora, we had access to user's and bloger's opinion about their state of mind at the moment of writing, rather than an external annotator's opinion. An additional important point about our corpus is that while it represents a large amount of different authors, it does not constitute a representative sample of adult writers. In fact, many of the blog maintainers are not adults. According to Livejournal, the median age of blog authors is 18, so half of the writers are actually teenagers.

Mood	Occurrences	Mood	Occurrences
amused	24857 (4.0%)	confused	8160 (1.3%)
tired	20299 (3.2%)	sick	7848 (1.3%)
happy	16471 (2.6%)	anxious	7052 (1.1%)
cheerful	12979 (2.1%)	exhausted	6943 (1.1%)
bored	12757 (2.0%)	crazy	6433 (1.0%)
accomplished	12200 (1.9%)	depressed	6386 (1.0%)
sleepy	11565 (1.81%)	curious	6330 (1.0%)
content	11180 (1.8%)	drained	6260 (1.0%)
excited	11099 (1.8%)	sad	6128 (1.0%)
contemplative	10724 (1.71%)	aggravated	5967 (1.0%)
blah	10127 (1.6%)	ecstatic	5965 (1.0%)
awake	10121 (1.6%)	blank	5441 (0.98%)
calm	10052 (1.6%)	hopeful	5059 (0.95%)
bouncy	10040 (1.6%)	thoughtful	4295 (0.91%)
chipper	9538 (1.5%)	frustrated	4132 (.90%)
annoyed	8277 (1.3%)	cranky	3945 (0.85%)
busy	7956 (1.3%)	loved	3883 (0.65%)
confused	7724 (1.20%)	angry	2832 (0.60%)
annoyed	7248 (1.15%)	working	2775 (0.55%)

Table 3.2: The most frequent moods in corpus.

3.4 Our Hierarchical Approach to Mood Classification

Most research in the area of data mining, text classification, and machine learning has focused on flat classification problems. Flat classification refers to standard binary or multi-class classification problems. On the other hand, many important real-world classification problems are naturally cast as hierarchical classification problems, where the classes to be predicted are organized into a hierarchical class or a Direct Acyclic Graph (Silla and Freitas, 2010).

Researchers have been developing new techniques to classify large volumes of documents and texts (e.g., biomedical data), particularly when the number of classes is large and the classes have a hierarchical structure (Dekel et al., 2004).

3.4.1 Hierarchical Classification

Hierarchical text categorization eliminates the assumption that categories are independent, and instead tries to incorporate the inter-relationship among the different categories. In a regular or flat text categorization approach, it is generally assumed that categories are independent and non-overlapping. This means that classification of an individual category is performed with the knowledge that all other categories are considered unrelated. In practice, however, categories tend to overlap and are not independent of each other (Silla and Freitas, 2010).

Hierarchical classification methods are different in a number of criteria(Freitas and Carvalho, 2007). The first is the type of hierarchical structure. This structure is based on the problem structure and it typically is either a tree or a DAG.

The second criterion is related to how deep the classification in the hierarchy is performed. That is, the hierarchical classification method can be implemented in a way that will always classify a leaf node refer to as mandatory leaf-node prediction.

The third criterion is related to how the hierarchical structure is explored. The current literature often refers to top-down (or local) classifiers, when the system employs a

set of local classifiers; big-bang (or global) classifiers, when a single classifier coping with the entire class hierarchy is used; or flat classifiers, which ignore the class relationships, typically predicting only the leaf nodes. However, a closer look at the existing hierarchical classification methods reveals the following issues related to learning, testing, and evaluation:

1. One of the main contributions of the hierarchical classification methods is to identify the "top-down" approach. This approach mainly concerns the testing phase of a classification algorithm, being therefore an approach to a large extent independent of the particular local approach used for training. By contrast, in the literature the term "top-down" approach is used to refer to both the testing and training phases, and without distinguishing the type of local classifier approach used in the training phase of the system (Silla and Freitas, 2010). The top-down approach is not a full hierarchical classification approach by itself, but rather a method for avoiding or correcting inconsistencies in class prediction at different levels, during the testing (rather than training) phase. Also, there are different ways of using local information to create local classifiers, and although most of them are referred to as top-down in the literature, they are very different during the training phase and slightly different in the test phase (Silla and Freitas, 2010).

2. According to Silla and Freitas (2010) most of the research so far has been concerned with tree-structured class hierarchies, probably due to the fact that trees are considerably simpler structures than DAGs (a major example is the Gene Ontology, which has become very important in biology). However, it seems that, there is a clear need for more research on hierarchical classification methods for DAG-structured class hierarchies.

3. Most hierarchical classification algorithms evaluate based on standard measures: accuracy/error rate and precision/recall. These measures are not suitable for hierarchical categorization since they do not differentiate among different kinds of misclassification errors. However, there are some points to consider. First, it is

difficult to extend to direct acyclic graph (DAG) hierarchies, where multiple paths between two categories can exist. Second, the values do not change with depth. Misclassification into a sibling category of a top level node and misclassification into a sibling of the deep node are considered the same type of error. An error at the deep level seems a lot less harmful than an error at the top level (Kiritchenko et al., 2004).

Wang et al. (1999) introduced two types of hierarchical text categorization methods: *global* and *local*. The global approach examines all the categories simultaneously in a global fashion and assigns labels with scores. On the other hand, the local method, uses local information and it is forced to make classification decisions at each internal node of a hierarchy, pushing most instances deep down (Kiritchenko et al., 2006). We explain these techniques below, and show how we applied hierarchical categorization techniques in the mood classification task.

Global Approach

Although the problem of hierarchical classification can use local approaches, learning a single global model for all classes has the advantage that the total size of the global classification model is typically considerably smaller, by comparison with the total size of all the local models learned by any of the local classifier approaches. In addition, dependencies between different classes with respect to class membership, e.g., any example belonging to a child class automatically belongs to parent class (Blockeel et al., 2002). This kind of approach is known as the big-bang approach, also called "global" learning. Big-bang (or global) classifiers are trained by considering the entire class hierarchy at once, and hence they lack the kind of modularity for local training of the classifier that is a core characteristic of the local classifier approach. In the global approach only one classifier discriminates all categories in a hierarchy. It differs from a flat categorization by considering the relations between the categories.

Local Approach

In the local approach, a classifier is built for each internal node in the hierarchy. It proceeds in a top-down fashion, starting with selecting the categories of the first level and then recursively choosing among the lower-level categories (the children of the higher-level categories). There are two possibilities: The first is to train one classifier for all the categories in the first level of the hierarchy (multi-class problem). Then one classifier is trained for each of the classes in the first level, to differentiate among all its sub-classes. The process continues if there are more levels. The second possibility is to build a single-class classifier for each node in each level of the hierarchy (a binary classifier that differentiates the class of the node from any other sibling classes). This approach has potential to do better learning during the training phase.

We used the second approach, with SVM classifiers, because it is more efficient. We had to choose between using n-ary classification at each node, or n-binary classifiers at each node (where n is the number of classes at each hierarchical level). We decided that only the n-binary classifiers method is feasible in our case: since we have a large number of classes the other would take too much time. Not only would the training times be too long, but the testing time would as well when running the classifiers on new data. Moreover, we used SVM classifiers, which are normally binary; thus we used binary classification. The results were combined by a Weka tool, in order to produce the final n-ary classifier.

We used the local approach, and we tried both ways mentioned above. For the first one, we used n-ary decision trees. For the second one, we used SVM classifiers, that are binary by nature, and then we combined them into n-ary classifiers using the Sequential Minimization Optimization (SMO) algorithm (Keerthi et al., 2001) from Weka. We briefly describe this method for combining classifiers in Section 3.4.2.

The second approach led to much better results. This is why we report only the SMO results in Section 3.6. In general, this approach has the disadvantage that it requires a large number of classifiers, because there are five hierarchy levels and there are many subclasses in each node of the hierarchy; but the SMO method in particular does not

have this disadvantage because it has an efficient way to combine the classifiers (without storing them as intermediary classifiers). Using the approach with other classifiers (e.g., decision trees), would make it very difficult to manually manage the many intermediary classifiers, and the results are not likely to be good, since, for this task, decision trees did not lead to good results.

3.4.2 Combining SVM Classifiers

Many researchers have become interested in SVM classifiers, because SVM has shown significant generalization performance on a wide range of problems (Vapnik, 1995; Burges, 1998). The SMO algorithm is a fast implementation of SVM introduced by Platt (1998): tests have shown that SMO is often much faster and has better scaling properties than other SVM implementations. However, it has a significant deficiency, caused by the manner in which it maintains and updates a single threshold value. Keerthi et al. (2001) proposed a solution to this problem. They suggested using two threshold parameters, and devised two modified versions of SMO that are more efficient than the original. Computational comparison on benchmark data sets showed that in most situations the modified versions perform significantly faster than the original SMO.

Another basic concern is that SVM is a binary classifier, and as such it cannot be used for problems with more than two classes. In the following, we examine the multi-class problem and the existing solutions.

The multi-class version of SMO uses a pair-wise coupling procedure, which involves estimating class probabilities for each pair of classes, then coupling the estimates together. It starts by forming all the pairs of two classes, and training classifiers for them. Then all the pair-wise decisions are combined to produce a multi-class classifier. The class that is finally chosen for a test instance is the class that has the highest number of pairs. The thresholds for the pairs of classes are optimized simultaneously, based on the training data. For more details, see Hastie and Tibshirani (1998).

In Weka, the implementation of Platt's SMO algorithm for training a support vector classifier uses the polynomial or the RBF (Radial Basis Function) kernel. The implemen-

tation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. Moreover, Weka's SMO is extended to the multi-class problem using the pairwise classification (Hastie and Tibshirani, 1998) explained above. To obtain the best probability estimates, we used it with the option that fits logistic regression models to the outputs of the support vector machine.

3.5 Feature Set

A brief explanation of the features we used in our machine learning experiments follows. We used most of the features from Mishne (2005), plus some additional sentiment orientation features, such as tagged words from the General Inquirer (Stone et al., 1966).

3.5.1 Frequency Counts

Bag-of-Words (BoW) is the most common feature representation used in automatic text classification. Most text classification systems consider a document as simply a "bag of words", and use the frequency of the words as features (Sebastiani, 2002); we also represented the words by their frequencies. Part-of-Speech (PoS) tags are another common feature of text classification systems. We used the frequency of words and PoS tag counts as features, and TreeTagger (Schmid, 1994) to extract the PoS tags.

3.5.2 Length-related Features

Since blog posts vary in length, we also considered length features, such as the length of the document, the number of sentences and the average number of words. We split the sentences by using standard punctuation marks as sentence delimiters.

3.5.3 Sentiment Orientation

We consider the task of estimating the sentiment orientations of a blog posting as a text classification problem. Each document of interest is represented as a "bag of words"

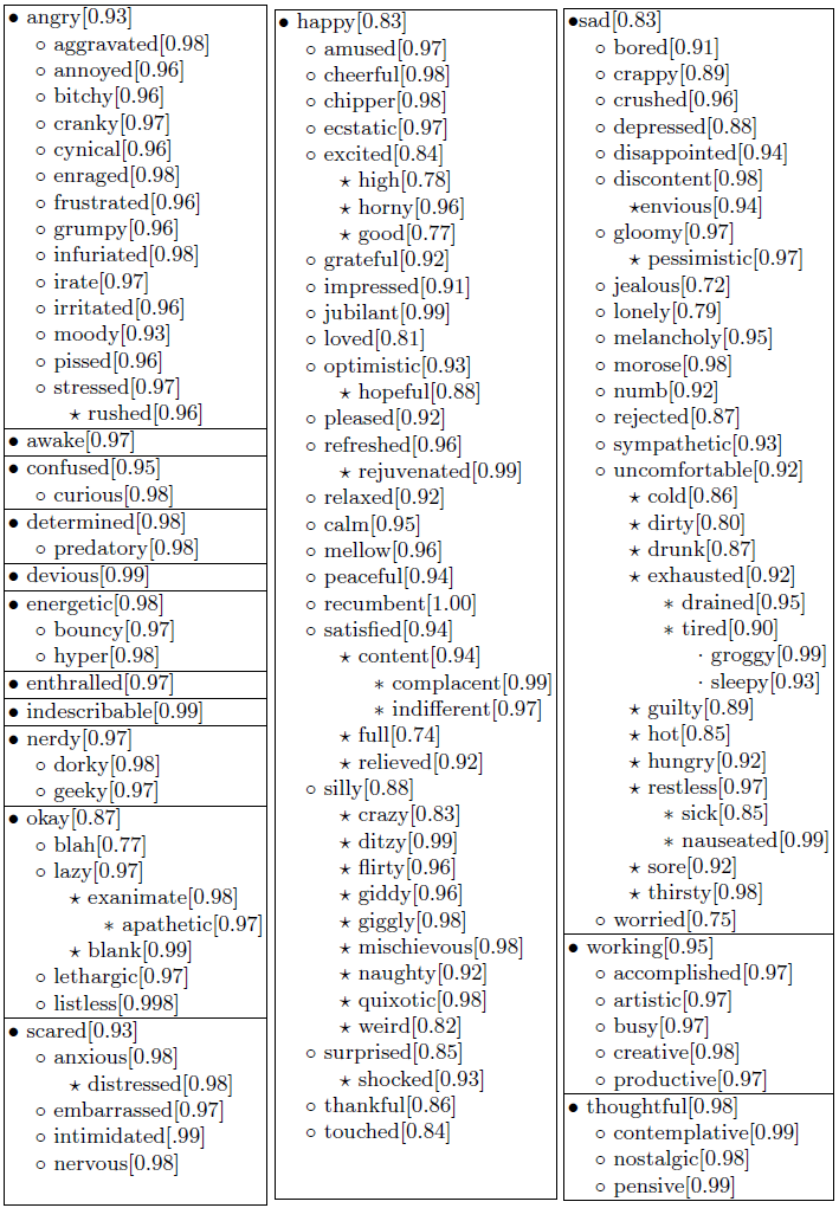

Figure 3.2: The hierarchy of the 132 moods; •:Level 1, ◦:Level 2, ⋆:Level 3, ∗:Level 4, ·:Level 5. The numbers in brackets indicate the best F-measure that we obtained for each node, as discussed later in Section 3.6.3

feature-vector x, where the entries of x are the frequencies with which the words in the vocabulary set V appear in the document. We need to know the orientation of each word to accurately estimate the orientation of the document. For each word, we assigned $+1$ for positive orientation, -1 for negative and zero for neutral. Therefore a score can be computed for each document, by summing up the word scores.

For emotion and mood classification, the sentiment orientation of some words can be a useful feature for predicting emotion and mood. For example, according to context some emotions or words, such as "annoyed" and "frustrated" are *negative* while others like "cheerful" and "loved" are *positive*. So, it is expected that blog posts with a positive aspect will, on average, have more positive orientation than negative blog posts.

In our feature set, we considered both the average word orientation and total orientation of a blog post as features. Determining the semantic orientation of a word is dependent on the method used for calculating it. We used several sources that are predictors of sentiment orientation. We calculated the total and the average orientation score for each document based on the words from the following resources:

- A list of 2,291 positive words and 1,915 negative words from the General Inquirer (GI) (Stone et al., 1966) (discussed in Section 2.1.6). These words were marked with affect classes, such as active, passive, strong, weak, pleasure, pain, feeling (other than pain/pleasure), arousal, virtue, vice, overstated and understated.

- A list of 21,885 verbs and nouns that were assigned a positive, negative, or neutral orientation score, we call this Kim-Hovy list (Kim and Hovy, 2004). Kim-Hovy list says negative for unparalleled because it is automatically acquired.

  In their research, they identified sentiments like the affective parts of opinions. They proposed a system that automatically finds those people who have opinions about a given topic, and the sentiment of each opinion. Their system contains a module for determining word sentiment, and another for combining sentiments within a sentence. They used various models of classifying and combining sentiment at word and sentence levels in their research, with promising results.

Word	Kim-Hovy	Turney-Littman
pricey	Positive	−4.99
repetitive	Positive	−1.63
teenage	Negative	−1.45
momentary	Negative	+0.01
fair	Positive	+0.02
earnest	Positive	+1.86
unparalleled	Negative	+3.67
fortunate	Positive	+5.72

Table 3.3: Semantic orientation values of words based on Kim-Hovy and Turney-Littman lists (Mishne, 2005)

- A list of 1,718 adjectives with their scores of polarity values, constructed by using the method developed by Turney and Littman (2003).

  We also used this source as sentiment orientation features. It contains a list of adjectives and their corresponding real-number polarity, with positive or negative values. The list was constructed according to Turney and Littman (2003), who measured the co-occurrence of a word with a small set of manually-classified keywords on the web.

For the above list of SO lexicons, we assigned +1 to positive words and −1 to negative words, and calculated the sum and the average score of each document and list of words in the lexicons.

Examples of words and their values in the Kim-Hovy and Turney-Littman lists are shown in Table 3.3. Here we need to clarify that the lists can be in conflict, for example one says positive one negative. But we use different score as feature for each list, and the Machine Learning algorithm decides which feature to trust.

3.5.4 Special Symbols

We used symbolic emotional icons (known as emoticons), which are printable characters intended to represent human emotions or attitudes. Emoticons are; often horizontal textual representations of facial expressions (e.g. .:) = smile and ;= wink).

Today, emoticons are very popular in many domains, including blogs and chat rooms. The use of emoticons originated in email messages, and quickly spread to other forms of online content. We used the nine most popular emoticons found in blog posts as features: tired: "(: |" , smile: ":)", kiss: ": $-*$", laughing: ": d", sad: ": (", love: ": x", worried: ": $-s$", surprise: ": $-o$", angry: "$x-$ (" in the blog posts as features.

3.6 Experiments and Results

Our experiments and results are described in this section. We explain the performance of the simple flat classification into 132 moods, and the performance of our hierarchy-based mood classification. We also explain the classification setting, then describe the experiments and results and investigate possible causes of errors.

3.6.1 Classification Setting

We used the Weka data mining software Hall et al. (2009) for our experiments. We chose Support Vector Machine (SVM) as the main classifier, because it has been shown to perform very well in text classification tasks, including previous studies on mood classification. Moreover, SVMs are able to deal with a large number of features and instances Joachims (1998). We experimented with a few other classifiers, such as Naïve Bayes and Decision Trees, but the results for our tasks were lower than those of SVM (the scores were 13 and 21 respectively). As explained in Section 3.4.2, we used the Sequential Minimization Optimization (SMO) algorithm from Weka to combine binary SVM classifiers into multi-class classifiers.

Our feature space is very large, so for efficiency, we reduced the feature space for

the *BoW* features by using feature selection methods. We used the *Chi-Square* feature selection method from Weka, in order to keep only the first 5,000 features from the 43,109 initial *BoW* features. In addition, we used the six sentiment orientation scores (for each of the three resources, the average score of the words from the text that are also in the resource, and their total score), and the emoticons.

3.6.2 Experiments

Our training data set for the experiments includes 234,129 instances, which vary among the 132 different moods. From all the posts, we randomly (66% from each class) selected the 144,129 as training data, as well as 90,000 (33% from each class) as test data. In this research, we split the data into training and test set in a similar way to Mishne. An alternative way would have been to select test data based on time-stamps. That is, the test data is more realistic if it was written at different time than the training data. Figure 3.3 shows the top 29 moods in our experimental data set (the training and test data that we selected). The distribution is slightly different than the distribution in the whole LiveJournal data described in Table 3.2.

Our first experiment consisted of a flat classification into the 132 moods, using the SMO classifier. The second experiment was performed to evaluate the hierarchical classification method. For this purpose, we first trained a classifier to classify into the 15 categories from the first level of hierarchy (i.e., *happy, sad, angry, okay, working, scared, awake, thoughtful, nerdy, indescribable, enthralled, determined, confuse, devious*, and *energetic*). In the next step, for each node from the first level of the hierarchy we extracted the related instances and their mood labels. For example, for the node *angry* we selected all the documents that have the label *angry, aggravated, annoyed, bitchy, cranky, cynical, enraged, frustrated, grumpy, infuriated, irate, irritated, moody, pissed*, and *stressed*. We then ran the classifier for each node in the second level. We repeated this procedure for each of the 15 categories from the first level of the hierarchy and we performed similar steps for the third, fourth and fifth level of the hierarchy.

In the next step, for each node from the first level of hierarchy we extracted the

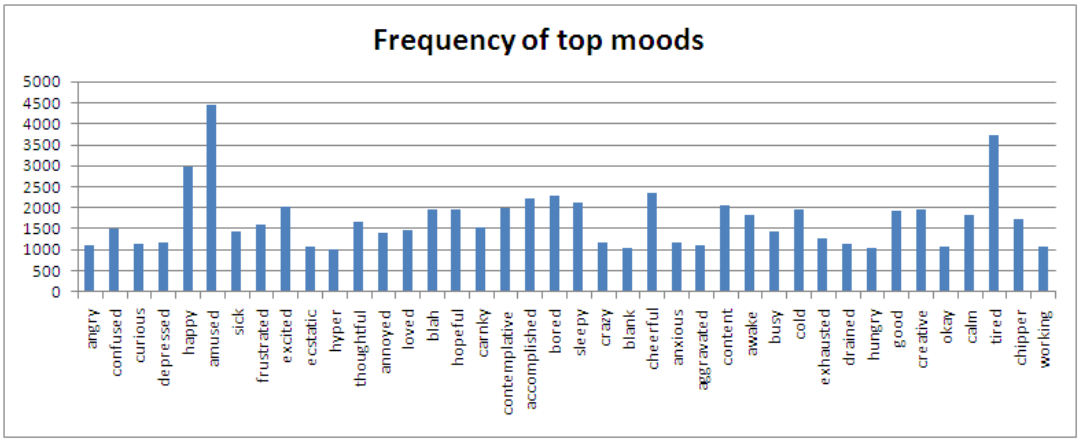

Figure 3.3: Frequency of the top moods in the experimental data set.

related instances and their mood labels. For instance, for the node *angry* we selected all the documents that have the label *angry, aggravated, annoyed, bitchy, cranky, cynical, enraged, frustrated, grumpy, infuriated, irate, irritated, moody, pissed*, and *stressed*. Finally, we ran the classifier for each node in the second level. We repeated this procedure for each of the 15 categories from the first level of the hierarchy. We continued similar steps for the third, fourth and fifth level of the hierarchy.

For both sets of experiments above we ran our classifiers using Bag of Words (BoW) features and used BoW plus Semantic Orientation (SO) features (including emoticons). We found that adding the SO features greatly improves the classification results (see the summary of results in Table 3.8).

3.6.3 Results and Discussion

The results of the first experiment, the flat classification into 132 moods, achieved an accuracy of 24.73% for $BoW+SO$ features and 18.29% for BoW features. This is an improvement compared to a baseline accuracy result of 7% when always choosing the most frequent mood (baseline is the lower expectation in each level for a naive classifier that always chooses the most frequent class).

In the second experiment, the hierarchical approach, the accuracy of the classifier that classifies into one of the 15 moods from the first level was 95.87% for $BoW+SO$ and

almost 40% for *BoW*, compared to a baseline of 15%.

For the next step we had 15 potential classifiers at the second level, one for each node at the first level. We ultimately had 11 classifiers, because four moods did not have any child branches in the hierarchy, so the classification for them was already completed. The average accuracy was 92.33% for *BoW+SO* features, 83.30% for *BoW* features only, and 32.70% for a baseline of the most frequent class. The difference between the hierarchical approach with the features and the baseline is 59.63%. Our baseline is a simple baseline, but it is necessary to compare the results with it.

There are several branches that have fewer children and show larger improvement, and several others with many children that show lower performance improvement. For example, the moods *happy*, *sad*, and *angry* have many child branches and the improvement is smaller, while the gain in performance is bigger for moods such as *nerdy*, which has two branches. Two branches mean three classes: in this case generic *nerdy* and two more specific types of *nerdy*: *geeky* and *dorky*. The results for Level 2 are shown in Table 3.4.

The results of the Level 3 classifier are shown in Table 3.5 and the results of Level4 in Table 3.6. Level5 has only one classifier, for *tired*, with an accuracy of 96.22% for *BoW+SO* features and 87.61% for *BoW*, with a baseline of 54.44%.

Our experiments and results clearly show that the hierarchical classification leads to strong performance and is well suited to the task. The summary of results for each level, shown in Table 3.8, supports this claim.

In Table 3.7, we added average precision, recall, and F-measure values for each level, for the best hierarchical classification setting (BoW+SO). The best F-measure for each leaf node is also indicated in square brackets in Figure 3.2, in order to show results for each of the 132 moods.

To directly compare the results of the flat categorization to the results of the hierarchical classifiers, we extracted the errors from all the levels. This provides a global accuracy of 79.92% for all 132 classes for *BoW+SO* features and 18.29% for the flat categorization. As illustrated in Table 3.8, this improvement in performance is achieved

Level 2	Baseline	BoW	BoW+SO
happy	8.64%	62.72%	86.97%
sad	10.38%	66.89%	86.88%
angry	11.67%	80.13%	91.90%
okay	24.55%	78.67%	82.25%
working	25.24%	87.74%	93.29%
scared	25.97%	89.21%	95.02%
thoughtful	35.99%	91.32%	94.84%
nerdy	41.40%	90.65%	97.68%
determined	65.52%	93.25%	95.83%
confused	56.32%	85.71%	94.33%
energetic	54.05%	90.05%	96.73%
Average	32.70%	83.30%	92.33%

Table 3.4: Accuracy for classification in Level 2 for both BoW and BoW+SO features.

when the sentiment orientation features are added. We believe the hierarchical method obtained better results because the classifiers are able to focus on a smaller number of classes, and learn the distinctions between them.

3.6.4 Error Analysis

The error rates are higher for classes with many sub-classes in the hierarchy, which is likely because the sub-classes are very similar to each other, so the algorithm cannot find enough features to distinguish between them. Another possible cause of errors is the nature of the hierarchy. We used it as it was provided, but there are a few areas where it does not seem logical; for example, the class "silly" being a sub-class of "happy". "Shocked" and "surprised" are also in the "happy" class, when in reality they could be negative (as in pleasant surprise versus unpleasant surprise). In addition, some of the 132 possible classes are not well-defined (e.g., recumbent and dorky), which could mean

Level3	Baseline	BoW	BoW+SO
uncomfortable	17.97%	71.03%	90.72%
surprised	56.18%	96.19%	97.68%
stressed	67.62%	94.58%	98.72%
silly	14.17%	74.53%	90.32%
satisfied	31.97%	80.15%	96.44%
refreshed	52.92%	95.55%	97.06%
optimistic	66.41%	90.35%	98.80%
lazy	31.37%	87.40%	96.88%
gloomy	66.21%	93.68%	99.88%
excited	35.87%	86.33%	91.75%
discontent	84.27%	92.08%	100%
anxious	58.88%	89.91%	96.85%
Average	48.65%	87.64%	95.84%

Table 3.5: Accuracy for classification at Level 3 for both BoW and BoW+SO features.

their labels have been arbitrarily chosen by the users who tagged their blogs.

To further investigate the difficulty of the task, we randomly selected a sample of blogs from those that were wrongly classified by our system. We wanted to see if human judges could classify them correctly (according to the tags chosen by the users who wrote the blogs) and, if not, determine the upper bound of the accuracy that can be achieved by humans on this sample. An automatic system cannot be expected to produce better results than this upper bound.

We cannot be certain that the data set was correctly annotated by the users who wrote the blogs. Some annotations might be wrong, which would lead to noise in the data set. We wanted to measure the agreement between human judges on this sample of data, to see how reliable the initial annotation is, and how difficult the task is. The judges were native speakers of English, graduate students in Computational Linguistics.

In general, the classification of opinions in texts is highly subjective, and this can cause

Level4	Baseline	BoW	BoW+SO
content	54.23%	89.73%	97.92%
restless	48.50%	90.27%	97.70%
exhausted	40.20%	89.17%	96.09%
exanimate	68.73%	96.46%	100%
Average	52.16%	91.40%	97.93%

Table 3.6: Accuracy for the hierarchical classification in Level 4 for both BoW and BoW+SO features.

Levels	Accuracy	Precision	Recall	F-measure
Level1	95.85%	0.95	0.95	0.95
Level2	92.33%	0.92	0.92	0.91
Level3	95.84%	0.99	0.99	0.99
Level4	97.93%	1.00	0.97	0.99
Level5	96.22%	0.99	0.99	0.99

Table 3.7: Overall average results in each level.

disagreement between the annotations of human judges. Differences in the skills and focus of the judges, and ambiguity in both the annotation guidelines and the annotation task itself, also contribute to disagreement between judges (Passonneau, 2006). We attempted to determine the degree of agreement between judges with respect to annotating blogs by mood; it is important to note that we did this using a sample of difficult blogs. If we had selected the sample from blogs correctly classified by the system, it is likely that human judges would be able to classify them correctly, and their annotations would be in agreement.

From our classification task for the first level of hierarchy, we randomly selected 58 instances which were wrongly *predicted* by our system (meaning that the 464 *actual* mood specified by the writer were not detected). In order to provide more detailed results for Level 1 classification, we added precision, recall and F-measure for each subclass of

Summary of the Results	
Baseline	7.00%
Flat Classification BoW	18.29%
Hierarchical Classification BoW	23.65%
Flat Classification BoW+SO	24.73%
Hierarchical Classification BoW+SO	79.92%

Table 3.8: The accuracy of the hierarchical classification when the classifiers from all the levels are applied successively (the errors from all the levels are multiplied), compared to the results of the flat classification, for both BoW and BoW+SO features.

this level (see Table 3.9), and a confusion matrix that shows the most frequent types of mistakes (see Figure 3.4).

We performed additional analysis on this random sample which our classifier had classified wrongly. We asked two human judges to assess it by annotating each text with one of the moods from the first level of hierarchy (i.e. indescribable, confused, angry, awake, devious, determined, energetic, enthralled, happy, nerdy, okay, sad, scared, thoughtful and working). They were also allowed to specify a second label, and to write comments.

We first measured the *correctness* of each judge. We trust the human judges in order to compute an approximate upper limit. As shown in Figure 3.5, the highest correctness score for classification by human judges was 36.79% which means the *upper boundary* for our system's correctness on this sample of data cannot be higher than 36.79%. These results demonstrate that the classification task in our blog corpus is very difficult. If we include the second choices of the judges as correct labelling, the highest correctness score goes up to 41.38%. In the second choice, the judges were more in agreement than for the first choice. These results are an estimate, since they are calculated on a sample of the data.

We then measured the degree to which the judges agreed when classifying a document into a corresponding mood. The agreement between judges A and B was 65.52%, and

Instances	Precision	Recall	F-Measure	Class
1	0.975	0.984	0.98	angry
2	0.957	0.991	0.974	awake
3	0.925	0.982	0.952	confused
4	0.98	0.984	0.982	determined
5	0.993	0.997	0.995	devious
6	0.985	1	0.992	energetic
7	0.989	0.989	0.989	enthralled
8	0.957	0.912	0.934	happy
9	0.993	1	0.997	indescribable
10	0.985	0.958	0.971	nerdy
11	0.96	0.802	0.874	okay
12	0.895	0.927	0.911	sad
13	0.924	0.953	0.939	scared
14	0.985	0.985	0.985	thoughtful
15	0.878	0.917	0.897	working
Weighted Avg.	0.959	0.958	0.958	0.992

Table 3.9: Detailed Results By Class for Level 1.

the kappa value that compensates for agreement by chance Cohen (1960) was 0.1889. The low agreement and kappa values prove that the task is difficult, even for human annotators. Also, some of the initial tags in the data set cannot be trusted. According to the comments of the human judges, the usual cause of a wrong tag is the existence of multiple moods in the same blog. The writer of the blog could choose only one mood, and it was most likely he/she would select the mood they were in when they started writing, then their mood changed during the writing, and this was reflected in the text. The judges also mentioned that some blogs described different moods of several persons that the blog author was writing about. Indeed, the evaluation is on a small sample of the data, since it would have taken too long for the human judges to look at all the data.

Confusion Matrix for Level 1.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	<-- classified as
744	2	5	0	0	0	0	0	0	0	1	0	0	2	2	a = angry
0	629	1	1	0	1	1	0	0	1	0	0	0	1	0	b = awake
0	5	491	0	0	0	0	0	0	0	0	1	2	0	1	c = confused
2	1	1	437	0	1	0	0	0	0	0	1	0	1	0	d = determined
0	0	0	0	301	0	0	0	1	0	0	0	0	0	0	e = devious
0	0	0	0	0	392	0	0	0	0	0	0	0	0	0	f = energetic
0	0	1	0	1	0	261	0	0	0	0	0	1	0	0	g = enthralled
4	7	6	5	0	1	1	917	1	1	5	21	13	2	21	h = happy
0	0	0	0	0	0	0	0	444	0	0	0	0	0	0	i = indescribable
1	0	5	0	0	1	0	0	0	203	0	1	0	0	1	j = nerdy
2	4	5	0	0	1	0	26	0	1	287	11	3	0	18	k = okay
8	5	2	1	0	1	0	3	0	0	2	357	2	2	2	l = sad
0	0	8	0	0	0	0	4	0	0	1	1	306	0	1	m = scared
0	2	2	0	1	0	1	0	1	0	0	0	1	529	0	n = thoughtful
2	2	4	2	0	0	0	8	0	0	3	6	3	0	330	o = working

Figure 3.4: Confusion Matrix for Correctly Classified Instances and Incorrectly Classified Instances at Level l.

Therefore the results are only estimations.

3.7 Hierarchical Classification for Six Emotions and Neutral Text

To avoid redoing the classification in six class of emotion (Ekman (1992) Emotion: happiness, sad, anger, fear, surprise, disgust), here we described the research that has been done in our NLP Lab by (Ghazi et al., 2010). In this they explore the task of automatic classification with six basic plus neutral (non-emotion text). In this hierarchical model, they consider how the presence of neutral instances affects the performance of distinguishing between emotions. Another facet of the evaluation concerns the relation

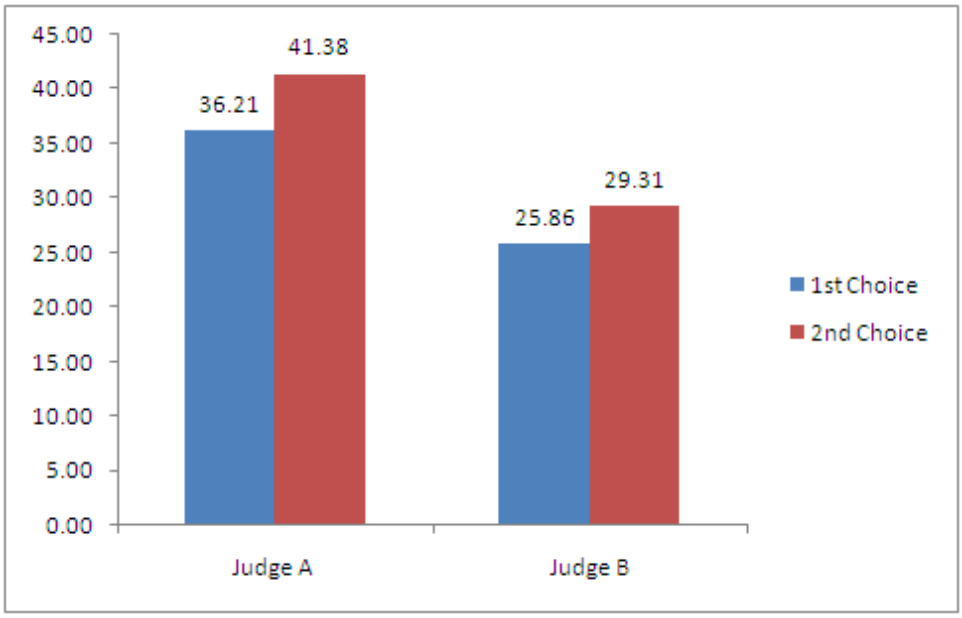

Figure 3.5: The Correctness results by two judges A and B.

between polarity and emotions. They apply a novel approach which arranges neutrality, polarity and emotions hierarchically. This method significantly outperforms the corresponding "flat" approach which does not take into account the hierarchical information. They also compare corpus-based and lexical-based feature sets and they choose the most appropriate set of features to be used in their hierarchical approach.

They convey the information in two forms of hierarchy. The first one is a two-level hierarchy which represents the relation of emotion and neutrality in text. The second form is a three-level hierarchy which addresses the relation between polarity and emotions in addition to the relation between emotion and neutrality. That is based on the assumption that, among the six chosen emotions, happiness belongs to the positive polarity class, while the other five emotions are regarded as having negative polarity (Ghazi et al., 2010).

Ghazi et al. (2010) perform sentence level classification, into six emotions plus one neutral class. In their research, they look at flat classification in seven classes, compared to hierarchical classification into neutral or emotion. In the two-level classifications,

the first level, emotional versus non-emotional classification, tries to determine whether an instance is neutral or emotional. The second step takes all instances which level 1 classified as emotional, and tries to classify them into one of the six emotion classes.

Their approach for two-level and three-level works as follows: If it is an emotion, they divide it into positive and negative. Then, if it is negative, they classify into "anger", "sadness", "disgust", "fear", "surprise". If it is positive, they consider as "happiness" class. The "surprise" class could have been split into positive and negative "surprise".

The authors used an annotated blog dataset for their experiments. In this dataset, each sentence is tagged by a dominant emotion in the sentence, or as non-emotional if it does not include any emotion. The dataset contains 173 weblog posts annotated by two judges. It is annotated based on Eckman's six emotions (Ekman, 1992) at the sentence level. The dataset contains 4090 annotated sentences, 68% of which were annotated as non-emotional. The highly unbalanced dataset with 68% of non-emotional sentences as the highest class and 3% of the fear and surprise classes prompted us to reduce the number of non-emotional sentences to 38% of all the sentences by removing 2000 of the non-emotional sentences, to reduce the unbalance. As a result, we chose for our experiments a corpus of blog sentences annotated with emotion labels, discussed in (Aman and Szpakowicz, 2007).

It is hard to show the results of all three applied approaches from here, but Figure 3.6 shows the combined results of all the results in, which displays the F-measure of each class for the three approaches.

3.8 Comparison to Related Work

Standard research on text classification is intended to detect the topic of documents. Less research has been done to detect specific features in the text (Sebastiani, 2002), such as the gender of the writer (Koppel et al., 2002b) or attribution (Koppel and Schler, 2004).

In this chapter, we focused on mood classification. The task is difficult because of the distinctive aspects of mood in blogs. For example, a blogger can start writing in a

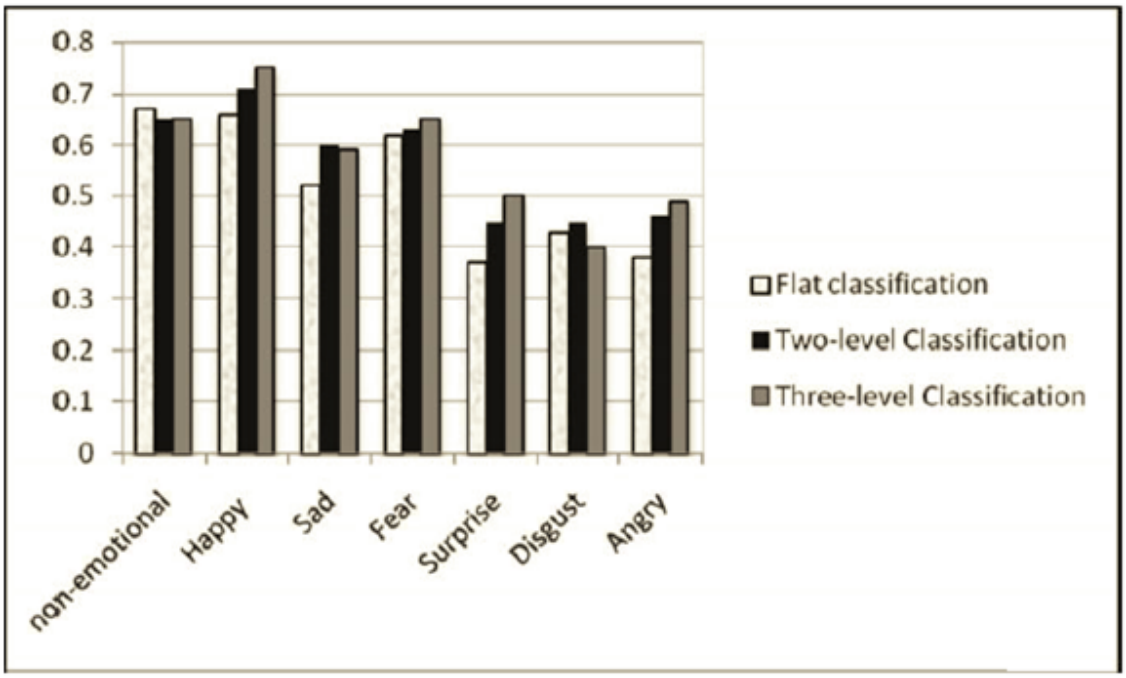

Figure 3.6: The comparison of F-measure result of the flat classification with the hierarchical classification approaches for seven classes (Ghazi et al., 2010).

certain mood, but the document can end with other content or mood. Some blogs are so intertwined that even human readers would have difficulty identifying the mood, and finding the relationship between the mood of the writer and the documents.

The work that is closest to our study is that of Mishne (2005), who used binary SVM classifiers for mood classification of blog data. As mentioned, we used the data set that he collected from LiveJournal. Mishne used only the 40 most-frequent moods, as shown in Table 3.2. He trained a classifier for each of the 40 moods, on different training sets, then showed the performance of each mood, based on various sizes of the corresponding training set. In order to compare our work with Mishne's results, we also classified into the 40 most-frequent moods. We obtained 84.89% accuracy, while the baseline given by the most frequent class is very low at 6.62%, and a random choice among 40 classes results in 2.5%. Mishne reported an average accuracy of 67%. He used a test set randomly chosen for each mood with a balanced distribution of classes, which means a 50% baseline for an approach that considers one class versus all the other classes.

Therefore, we are unable to directly compare our results to his, because we cannot use precisely the same test set, even though our test set is randomly chosen from the same data .

In summary, the essential differences between our work and Mishne's is that we used all the 132 moods, not only the 40 most-frequent ones, and we enhanced the feature set with more sentiment orientation features. In order to achieve good results for such a large number of classes, we chose to use a different approach (i.e., hierarchical classification).

Other research that investigated mood classification was done by Jung et al. (2006) who proposed a hybrid approach to identify mood in blog texts. They used ConceptNet (Liu and Singh, 2004) and the Affective Norms English Words (ANEW) (Bradley and Lang, 1999) list. To perform classification, they chose unique features from blogs and calculated statistical features such as term frequency, n-grams, and point-wise mutual information (PMI) for the SVM classification method. They also used a paragraph-level segmentation based on mood flow analysis, and extracted their own blog data from LiveJournal. They show good classification results only when restricting the task to four mood types: *happy, sad, angry,* and *scared*. In addition, they considered only documents with lengths of sentences between 4 and 40 tokens.

Liu et al. (2003a) introduced another approach that works on a larger corpus and is based on "Open Mind Commonsense" classification. Open Mind Commonsense was used as a real world corpus of 400,000 facts about the everyday world. They combined four linguistic models in order to recognize the affect. Their system analyzed the affective qualities of text, sentence by sentence. This has a practical value for people who want to evaluate the text they are writing.

Hierarchical categorization has been used for other text classification tasks. For example, Kiritchenko et al. (2006) applied this approach to biomedical texts, using a hierarchy from gene ontology, with good results. Wilson et al. (2005) applied a hierarchical approach to classification of customer reviews as positive, negative, or neutral. They used a very shallow hierarchy, to classify into neutral and polar texts, and to further classify the polar texts into positive and negative, with mixed results.

3.9 Summary

This chapter presented experiments on hierarchy-based mood classification of blog corpus. In addition, a hierarchical approach was considered to classify the data using the SVM algorithm. Our corpus was collected from Livejournal, an online service that allows users to post their personal thoughts. Our results showed that the hierarchical approach provides substantial performance improvement over flat classification. Classifying mood in blog text is a difficult task, due to the variety of users. However, our approach shows that if we classify blogs using the mood hierarchy, we can achieve very good performance. We also showed that using Sentiment Orientation (SO) features on top of Bag-of-Words (BoW) features greatly improves the classification results.

Chapter 4

Paraphrase Extraction for Emotion Terms

4.1 Introduction

Paraphrasing accrues from the various lexical and grammatical means of expressing meaning accessible in language. The importance of paraphrases has become more recognized recently, attracting researchers in field of computational linguistics (CL). Paraphrases ensure the main aspects of variability in language.

Paraphrases are different ways to express relevant information (at least in terms of meaning). Algorithms to extract and automatically identify paraphrases are interesting from both a linguistic and practical perspective. Many major functions of Natural Language Processing applications, such as multi-document summarization, need to avoid repetitive information from input documents, so in this case paraphrasing can be used in such applications. In natural language generation (NLG), paraphrasing is employed to create more varied and natural text. In our research, we extract paraphrases for emotions, with the intent of using them to automatically generate emotional texts for conversations between intelligent agents and characters in educational games. The paraphrasing is applied to generate text with more variety. To our knowledge, most current applications manually collect paraphrases for specific applications, or they use lexical

Table 4.1: Two sentence fragments from the emotion class *happy*, from the blog corpus.

his little boy was so happy to see him
the princess and she were very glad to visit him

resources such as WordNet (Felbaum, 1998) to identify paraphrases.

In this chapter, we introduce a novel method for extracting paraphrases for emotions from texts. We focus on the six basic emotions proposed by Ekman (1992): *happiness*, *sadness*, *anger*, *disgust*, *surprise*, and *fear*.

We describe the construction of the paraphrase extractor, and we also propose a k-window algorithm for selecting the contexts that are used in the paraphrase extraction method. We automatically learn patterns that can extract the emotion paraphrases from corpora, starting with a set of seed words. We make use of data sets, such as blogs and other annotated corpora, in which the emotions are marked. We also use a large collection of non-parallel corpora, which are described in Section 4.3. These corpora contain many instances of paraphrasing of using different phrases to express the same emotion.

An example of sentence fragments for one emotion class, *happiness*, is shown in Table 4.1. From them, the paraphrase pair that our method will extract is:

`"so happy to see"  "very glad to visit"`.

We provide an overview of related work on paraphrasing in Section 4.2. In Section 4.3 we describe the data sets that were used in this research, and we explain the details of our paraphrase extraction method in Section 4.4. We present results of our evaluation and discuss our results in Section 4.5 and finally in Section 4.6 we address some contributions and future related work.

4.2 Related Work

Three main approaches to collecting paraphrases have been proposed in the literature: manual collection, utilization of existing lexical resources, and corpus-based extraction

of expressions that occur in similar contexts (Barzilay and McKeown, 2001). Manually-collected paraphrases were used in natural language generation (Iordanskaja et al., 1991). Langkilde and Knight (1998) used lexical resources in statistical sentence generation, summarization, and question-answering. Barzilay and McKeown (2001) used a corpus-based method to identify paraphrases from a corpus of multiple English translations of the same source text. Our method is similar to this, but it extracts paraphrases only for a particular emotion, and requires a regular corpus, not a parallel corpus of multiple translations.

4.2.1 Applications of Paraphrases Extraction and Generation

In their survey, Madnani and Dorr (2010) categorized applications of paraphrase extraction and generation into following categories.

1. **Query and Pattern Extension:** One of the most common applications of paraphrasing is the automatic generation of query variants for submission to information retrieval systems or of patterns for submission to information extraction systems.

2. **Human Reference Data for Evaluation:** To measure the performance of the machine translation and summarization systems. Machine translation and document summarization are two applications that use comparison against human-authored references; as one of their evaluation methods.

3. **Machine Translation:** Paraphrasing has also been applied to directly improve the translation process. Automatically induced paraphrases can be used to improve a statistical phrase-based machine translation system. Such a system works by dividing the given sentence into phrases and translating each phrase individually, by looking up its translation in a table.

4.2.2 Paraphrasing with Corpora

Madnani and Dorr (2010) also explored in detail the corpus-based paraphrase generation and extraction approaches that have emerged and have become popular in the last decade or so. These corpus-based methods have the potential of covering a much wider range of paraphrasing phenomena and the advantage of the widespread availability of corpora.

Madnani and Dorr (2010) organized the corpus-based paraphrases generation and extraction by the type of corpora used to generate the paraphrases. They believe this form of organization is the most instructive, because most of the algorithmic decisions made for paraphrase generation will depend heavily on the type of corpus used. These categories are the following.

1. **Distributional Similarity:** It is relatively easy to realize the concept of distributional similarity; words or phrases that share the same distribution. The same set of words in the same context in a corpus tend to have similar meanings.

2. **Paraphrasing Using a Single Monolingual Corpus:** They are paraphrase generation methods that operate on a single monolingual corpus. The idea is almost the same as the Distributional Similarity.

3. **Paraphrasing Using Monolingual Comparable Corpora:** It is also possible to generate paraphrase pairs from a parallel corpus where each component of the corpus is in the same language. These corpora are comparable instead of being truly parallel: Parallelism between sentences is replaced by just partial semantic and topical overlap at the level of documents. Therefore, for these corpora, the task of finding paraphrases is harder.

4. **Paraphrasing Using Monolingual Parallel Corpora:** Such corpora are usually not very easily available (comparable instead of being truly parallel). Parallelism between sentences is replaced by just partial semantic and topical overlap at the level of documents, e.g, describing events with the same topics.

5. **Bilingual Parallel Corpora:** Using bilingual parallel corpora for paraphrasing has the inherent advantage that sentences in the other language are semantically equivalent to sentences in the intended paraphrasing language.

Glickman and Dagan (2004) investigated the extraction of lexical paraphrases from a single corpus. They used a syntactic parser to detect structures related to the same event, and from two such structures they extracted the paraphrases for verbs. Their method works only for verbs, while our method allows the extraction of any lexical paraphrases if sample seeds are available.

Some research has been done in paraphrase extraction for natural language processing and generation for different applications. Das and Smith (2009) presented an approach to determining whether two sentences have a paraphrase relationship. They applied a generative model to produce a paraphrase of a given sentence, then used probabilistic inference to asses whether two sentences share the paraphrase relationship. In another research, Wang et al. (2009) studied the problem of extracting technical paraphrases from a parallel software corpus. Their aim was to report duplicate bugs, and to do this they used sentence selection, and global context-based and co-occurrence-based scoring. Studies have also been done in paraphrase generation in NLG (Zhao et al., 2009), (Chevelu et al., 2009).

Bootstrapping methods have been applied to various natural language applications, including word sense disambiguation (Yarowsky, 1995), lexicon construction for information extraction (Riloff and Jones, 1999), and named entity classification (Collins and Singer, 1999). In our research, we use a bootstrapping approach to learn paraphrases for emotions.

4.3 Data

The text data from which we extract paraphrases are composed of four concatenated datasets. They contain sentences annotated with the six basic emotions. The number of sentences in each data set is presented in Table 4.2.

Table 4.2: The number of emotion-annotated sentences in each dataset.

Dataset	Happiness	Sadness	Anger	Disgust	Surprise	Fear
LiveJournal	7705	1698	4758	1191	1191	3996
TextAffect	334	214	175	28	131	166
Fairy tales	445	264	216	217	113	165
Annotated blog dataset	536	173	115	115	172	179

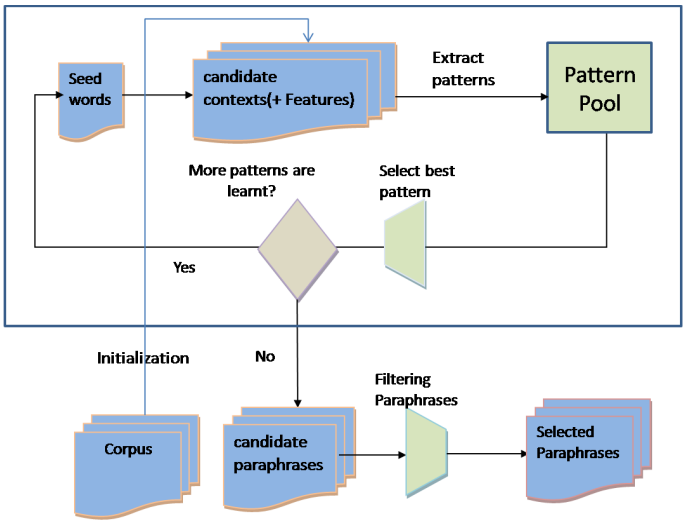

Figure 4.1: High-level view of the paraphrase extraction method (adapted from Banea et al. (2008a).

4.4 Our Method for Paraphrase Extraction

For each of the six emotions, we ran our method on the set of sentences marked with the corresponding emotion from the concatenated corpus. We started with a set of seed words form WordNet Affect (Strapparava and Valitutti, 2004), for each emotion of interest. The number of seed words was: happiness 395, surprise 68, fear 140, disgust 50, anger 250, and sadness 200. Table 4.3 shows some of seeds for each category of emotion.

Since sentences in our data sets differ and they are not aligned as parallel sentences as in (Barzilay and McKeown, 2001), our algorithm constructs pairs of similar sentences,

Happiness
glad, warmheartedness, exalt, comforting, joviality, gladness, romantic
enjoy, amorous, joyful, like, cheer, adoring, fascinating, happy, impress, merry
satisfaction, joy, great, cheerful, charmed, pleased, inspire, good, fulfill
Sadness
poor, sorry, woeful, miserable, glooming, bad, grim, tearful, glum, dismay, mourning
dismal, joyless, sadness, blue, rueful, hamed, regret, hapless, regretful
misery, godforsaken, oppression, harass, dark, sadly, attrition
Anger
belligerence, envious, aggravate, resentful, abominate, murderously, greedy, bother
angry, disdain, hatred, annoy, mad, jealousy, huffiness, sore, anger, harass, envy
enraged, hostile, hateful, irritating, outrage, devil, irritate
Disgust
sicken, foul, disgust, nausea, revolt, nauseous, hideous, horror, detestable, wicked
repel, offensive, repulse, yucky, repulsive, queasy, obscene, noisome
Surprise
wondrous, amaze, gravel, marvel, fantastic, wonderful, marvelous, wonder, trounce
wonderment, astonish, admiration, terrific, dumbfounded, surprising
Fear
fearful, apprehensively, anxiously, presage, horrified, hysterical, timidity, dire
horrible, timid, fright, hesitance, affright, trepid, horrific, unassertive, panic
apprehensiveness, hideous, scary, cruel, scared, terror, awful, fear,
dread, crawl, anxious, distrust, diffidence

Table 4.3: Some of the seeds from WordNet Affect for each category of emotion.

based on the local context. We assumed that if the contexts surrounding two seeds look similar, then these contexts are likely to be helpful for extracting new paraphrases. Once we find which contexts are useful for identifying paraphrases, we are able to extract more paraphrase patterns from our data set. Therefore, we simply define a context as a window of $\pm k$ words which surrounding an emotion seeds.

Verb-object relations and noun-modifier relations are examples of such contexts. They have been used from non-parallel corpora in word similarity tasks (Pereira et al., 1993; Hatzivassiloglou and McKeown, 1997). However, in our task, we can have other relations for paraphrasing, because we know which sentences convey the same information and belong to the same category of emotions.

Figure 4.1 illustrates the high-level architecture of our paraphrase extraction method. The input to this is a text corpus for an emotion category and a manually defined list of seed words. Before starting bootstrapping, we ran the k-window algorithm on every sentence in the corpus, in order to construct *candidate contexts*. In Section 4.4.6 we explain how the bootstrapping algorithm processes and selects the paraphrases based on strong surrounding contexts. As shown in Figure 4.1, our method has several stages: extracting candidate contexts, using them to extract patterns, selecting the best patterns, extracting potential paraphrases, and filtering these to get the final paraphrases.

4.4.1 Preprocessing

During preprocessing, HTML and XML tags are eliminated from the blog data and other data sets, then the text is tokenized and annotated with part-of-speech tags. We use the Stanford part-of-speech tagger and chunker (Toutanova et al., 2003) to identify noun and verb phrases in the sentences. Then we use a sliding window based on the k-window approach, to identify candidate contexts that contain the target seeds.

4.4.2 The *k*-window Algorithm

We use the k-window algorithm introduced by Bostad (2003) to identify all the tokens surrounding a specific term in a window with a size of $\pm k$ words. We extract candidate patterns for each seed, from the sentences. Starting with one seed, we extract all surrounding contexts within a window of k words before and k words after each occurrence of the seed, until all occurrence of the seed have been processed. Studies showed that five tokens ($k = 5$) achieved better results in k-window approach Khan (2007). Since the value of k is set to 5 for these experiments the longest candidate contexts will have the form $w_1, w_2, w_3, w_4, w_5, seed, w_6, w_7, w_8, w_9, w_{10}$, where w_i $(i = 1, ..., 10)$ are word tokens. We explain the features we extract from each candidate context to determine similar contexts in the same subsection.

4.4.3 Feature Extraction

Previous research on word sense disambiguation on contextual analysis has identified several local and topical features that are good indicators of word properties. These include surrounding words and their part-of-speech tags, collocations, and keywords in contexts (Mihalcea, 2004). Other features have also been proposed: bigrams, named entities, syntactic features, and semantic relations with other words in the context.

We transfer the candidate phrases extracted by the sliding k-window algorithm into a vector space of features. We consider features that include both lexical and syntactic descriptions of the paraphrases for all pairs of two candidates. The lexical features include the sequence of tokens for each phrase in the paraphrase pair. The syntactic features consist of a sequence of part-of-speech (PoS) tags where equivalent words and words with the same root and the same PoS are marked. For example, the value of the syntactic feature for the pair "`so glad to see`" and "`very happy to visit`" is "$RB_1\ JJ_1\ TO\ VB_1$" and "$RB_2\ JJ_2\ TO\ VB_2$", where indices indicate word equalities. However, based on the above evidence and our previous research, we also investigate other features that could help to achieve our goal. Table 4.5 lists the features that we

used for paraphrase extraction, which also include some term frequency features. For example, in Table 4.4 we show extracted features from a relevant context. Our results prove that these features work and they are useful at least for our algorithm.

Table 4.4: An example of extracted features.

Candidate context: He was further annoyed by the jay bird
'PRP VBD RB VBN IN DT NN NN',65,8,'VBD RB',?,was, ?,?,?,He/PRP,was/VBD,further/RB,annoyed,by/IN,the/DT, jay/NN,bird/NN,?,?,jay,?,'IN DT NN',2,2,0,1

4.4.4 Analyzing the Context Surrounding the Seeds

Considering the differences between sentences from different corpora, our method builds on the similarity of the local contexts within k-windows, rather than on global contexts. For example, consider the two sentences in Table 4.6. Analyzing the contexts surrounding the "seed" placeholder (in italics and bold in both sentences), it is expected that the two contexts would have the same meaning, because they have the similar pre-modifiers ("so" and "very"), and a post-modifier related to the same preposition, "to". In fact, the first seed is "glad", and the second seed is "happy". We hypothesize that if the contexts surrounding two phrases look similar enough, these two phrases are likely to be paraphrases. Once we know which contexts are good paraphrase predictors, we can extract paraphrase patterns from our corpus.

Can we conclude that identical modifiers of a subject imply verb similarity? To address this question, we need to identify contexts that are good predictors for paraphrasing in a corpus. To find relevant contexts, we can analyze all the contexts surrounding corresponding words in the pairs of aligned sentences, and use these to learn new paraphrases. This provides the basis for a bootstrapping mechanism. Starting with equivalent seed words in aligned sentences, we can incrementally learn the relevant contexts, then use these to learn new paraphrases. Equivalent seed words play two roles in this process;

Table 4.5: The features that we used for paraphrase extraction.

Features	Description
F1	Sequence of part-of-speech
F2	Length of sequence in bytes
F3	Number of tokens
F4	Sequence of PoS between the seed and the first verb before the seed
F5	Sequence of PoS between the seed and the first noun before the seed
F6	First verb before the seed
F7	First noun before the seed
F8	Token before the seed
F9	Seed
F10	Token after the seed
F11	First verb after the seed
F12	First noun after the seed
F13	Sequence of PoS between the seed and the first verb after the seed
F14	Sequence of PoS between the seed and the first noun after the seed
F15	Number of verbs in the candidate context
F16	Number of nouns in the candidate context
F17	Number of adjective in the candidate context
F18	Number of adverbs in the candidate context

they are used to learn context rules, and they are used in the application of these rules, because the rules contain information about the equality of the words in context. This method of co-training has been previously applied to a variety of natural language tasks, including word sense disambiguation (Yarowsky, 1995), lexicon construction for information extraction (Riloff and Jones, 1999), and named entity classification (Collins and Singer, 1999). In our case, the co-training process creates a binary classifier, which predicts whether a given pair of phrases creates a paraphrase.

Table 4.6: Two sentence fragments (candidate contexts) from the emotion class *happy*, from the blog corpus.

his little boy was so ***"seed"*** to see him
princess and she were very ***"seed"*** to visit him

4.4.5 Predicting Pairs of Paraphrases from Contextual Features

We define the contextual features as combinations of the left and right syntactic contexts surrounding actual known paraphrases. There are a number of context representations that are possible candidates, including lexical n-grams, POS n-grams and parse tree fragments. Part-of-Speech tags provide the required level of abstraction, and can be accurately computed for our texts. Suppose we have two candidate k-windows $w1$ and $w2$. Then we define $pair_1 = (left_1$, "seed", $right_1)$ and $pair_2 = (left_2$, "seed", $right_2)$. As mentioned in Section 4.4.2, the left or right context is a sequence of part-of-speech tags of k words, occurring on the left/right of the paraphrase. We mark tags of equivalent words in each pair as syntactic paraphrase features. For example, when $k = 5$, the contextual feature for the paraphrase pair ("glad", "happy") from the sentences in Table 4.4 is: "RB_1 JJ_1 TO VB_1" and "RB_2 JJ_2 TO VB_2". In the next section, we describe how we learned to extract paraphrases, using these contextual features.

4.4.6 Bootstrapping Algorithm for Paraphrase Extraction

Our bootstrapping algorithm is summarized in Figure 4.2. It begins with a set of seeds, which are considered initial paraphrases. The set of extraction patterns is initially empty. The algorithm generates candidate contexts from the aligned similar contexts. The candidate patterns are scored by the number of paraphrases they can extract. Those with the highest scores are added to the set of extraction patterns. Using the extended set of extraction patterns, more paraphrase pairs are extracted and added to the set of paraphrases. Using the enlarged set of paraphrases, more extraction patterns are

extracted. The process repeats until no new patterns or paraphrases are learned.

The results and evaluation show that our method performs well and produces acceptable outputs. Our bootstrapping algorithm uses two loops, one for each seed and one for each paraphrase pattern; therefore, the complexity of our bootstrapping algorithm is $O(n^2)$.

Our method can accumulate a large lexicon of emotion phrases, by bootstrapping from the manually initialized list of seed words. In each iteration, the paraphrase set is expanded with related phrases found in the corpus, which are filtered using a strong measure of surrounding context similarity. These strong measure is based on the number of pair of paraphrase that extracted by each pair. On the other hand, each pair that extracts more paraphrase is stronger. The bootstrapping process starts by selecting a subset of the extraction patterns that are intended to extract the paraphrases. We call this set the "pattern pool". The phrases extracted by these patterns become candidate paraphrases. They are filtered based on how many patterns select them, in order to produce the final paraphrases from the set of candidate paraphrases.

Algorithm 1: Bootstrapping Algorithm.

```
Input: seeds set, corpus, and k-window candidates
Output: Set of paraphrases
For each seed of emotion in seeds set
  Loop until no more paraphrases or no more contexts are learned.
    1- Locate the seeds in each sentence
    2- Find similar contexts surrounding a pair of two seeds
    3- Analyze all contexts surrounding the two seeds to extract
       the strongest patterns
    4- Use the new patterns to learn more paraphrases
```

Figure 4.2: Our bootstrapping algorithm for extracting paraphrases.

Initialization

Words which appear in both sentences of an aligned pair are used to create the initial seed patterns. Using seed words (or equivalent words) for the emotion class, we create a set of positive paraphrasing examples, such as $word_1 = glad$, and $word_2 = happy$. The words that are not equivalent in aligned pairs are used to produce negative examples as well, so these words are not paraphrases of each other. To find negative examples, we match the equivalent words against all the different words in the pairs, and we stipulate that equivalent words can match only each other, and no other word in the aligned pairs.

Extracting Patterns

Using these initial seeds, we record contexts around positive and negative paraphrase examples. From all the extracted contexts, we need to identify those ones which are strong predictors of their category. We extracted the features from each candidate context, as described above. Then we learn extraction patterns, in which some words are substituted by their part-of-speech. We use the seeds to build initial patterns. Two candidate contexts that contain the same seed create one positive example. By using each initial seed, we can extract all contexts surrounding these positive examples. Then we select the stronger ones. We used the method of Collins and Singer (1999) to compute the strength of each example. If we consider x as a context, the strength of x as a positive example is defined as:

$$Strength(x) = count(x+)/count(x) \tag{4.1}$$

where $count(x+)$ is the number of times context x surrounded a seed in a positive example and $count(x)$ is frequency of the context x. This allows us to score the potential pattern.

Context length is an important parameter for learning new patterns. Based on our experiments, we found a context length of three words leads to the best results. Moreover, we noticed that for some patterns, less lengthy contexts perform better. For these reasons, when we record contexts around positive examples, in most cases we keep all the contexts with lengths less than or equal to three.

Since we have different corpora, it is expected that the similarity between contexts varies between corpora. Therefore, our contextual patterns are learned for each corpus separately.

Extracting the Paraphrases

After we extracted the context patterns in the previous step, we applied them to the corpus to detect a new set of pairs of paraphrases. The patterns that were determined by searching for pairs of paraphrases that match the *left* and the *right* parts of a *seed* are strings of up to k tokens. Then, for each extracted pair, new paraphrases are recorded and filtered in the same manner as the contextual patterns. This iterative process is terminated when no new paraphrases have been detected.

4.5 Results and Evaluation

Our algorithm generates a set of extraction patterns, and a set of pairs of paraphrases: some of the extracted paraphrases are shown in Table 4.7. The paraphrases that are considered correct are shown under *Correct paraphrases*. As explained in the next section, two human judges agreed that these paraphrases are acceptable. The results considered incorrect by the two judges are shown under *Incorrect paraphrases*.

Our algorithm learned 196 extraction patterns, and produced 5,926 pairs of paraphrases. Table 4.8 shows the number of extraction patterns and the number of paraphrase pairs produced by the algorithm for each class of emotions. We used two techniques to evaluate our algorithm: First, human judges determined if the paraphrases extracted from the sample are correct, and we calculated the degree of agreement between the judges (See Section 4.5.1). Second, we assessed the recall and precision of our method (See Section 4.5.2. We describe these evaluations in the following subsections.

Disgust
Correct paraphrases:
being a wicked::getting of evil; been rather sick::feeling rather nauseated; feels somewhat queasy::felt kind of sick; damn being sick::am getting sick
Incorrect paraphrases:
disgusting and vile::appealing and nauseated; get so sick::some truly disgusting
Fear
Correct paraphrases:
was freaking scared::was quite frightened; just very afraid::just so scared; tears of fright::full of terror; freaking scary::intense fear;
Incorrect paraphrases:
serious panic attack::easily scared; not necessarily fear::despite your fear
Anger
Correct paraphrases: upset and angry::angry and pissed; am royally pissed::feeling pretty angry; made me mad::see me angry; do to torment::just to spite
Incorrect paraphrases:
very pretty annoying::very very angry; bitter and spite::tired and angry
Happiness
Correct paraphrases:
the love of::the joy of; in great mood::in good condition; the joy of::the glad of; good feeling::good mood
Incorrect paraphrases:
as much eagerness::as many gladness; feeling smart::feel happy
Sadness
Correct paraphrases:
too depressing::so sad; quite miserable::quite sorrowful; strangely unhappy::so misery; been really down::feel really sad
Incorrect paraphrases:
out of pity::out of misery; akward and depressing::terrible and gloomy
Surprise
Correct paraphrases:
amazement at::surprised by; always wonder::always surprised; still astounded::still amazed; unexpected surprise::got shocked
Incorrect paraphrases:
passion and tremendous::serious and amazing; tremendous stress::huge shock

Table 4.7: Examples of paraphrases extracted by our algorithm (correctly and incorrectly).

4.5.1 Evaluating Correctness with Human Judges

We used the same method as Barzilay and McKeown (2001) to evaluate the correctness of the extracted paraphrase pairs. We randomly selected in total 600 paraphrase pairs from the lexical paraphrases produced by our algorithm, 100 paraphrase pairs for each class of emotion. Two human judges evaluated the correctness of each paraphrase, to determine whether or not the two expressions are good paraphrases.

We provided guidelines for the judges, in which we defined paraphrase as "approximate conceptual equivalence", the same definition as that used by Barzilay and McKeown (2001). Each judge could choose "Yes" or "No" for each pair of paraphrases being tested. We did not include example sentences which contained these paraphrases.

Class of Emotion	# Paraphrases Pairs	# Extraction Patterns
Disgust	1125	12
Fear	1004	31
Anger	670	47
Happiness	1095	68
Sadness	1308	25
Surprise	724	13
Total	5926	196

Table 4.8: The number of lexical and extraction patterns produced by the algorithm.

A similar Machine Translation evaluation task for word-to-word translation was done by Melamed (2001).

Figure 4.3 presents the results of the evaluation (i.e. the correctness for each class of emotion according to judges A and B). By *correct* we mean those that judges agree (both judges) and *incorrect*, those that judges do not agree (both judges). The judges were graduate students in computational linguistics, and native speakers of English.

We also measured the agreement between the two judges and the Kappa coefficient (Siegel and Castellan, 1988). If there was complete agreement between the judges $Kappa = 1$, and if there is no agreement $Kappa = 0$. The $Kappa$ values and agreement values for our judges are presented in Figure 4.4.

The inter-judge agreement for all the paraphrases of the six classes of emotions is 81.72% (490 out of the 600 paraphrase pairs in our sample). The judges agreed that some pairs are good paraphrases or some pairs are not good paraphrases, which is why the values in Figure 4.4 are higher than the correctness values in Figure 4.4. The $Kappa$ coefficient compensates for chance agreement. The $Kappa$ value over all the paraphrase pairs is 74.41%, which is significant agreement.

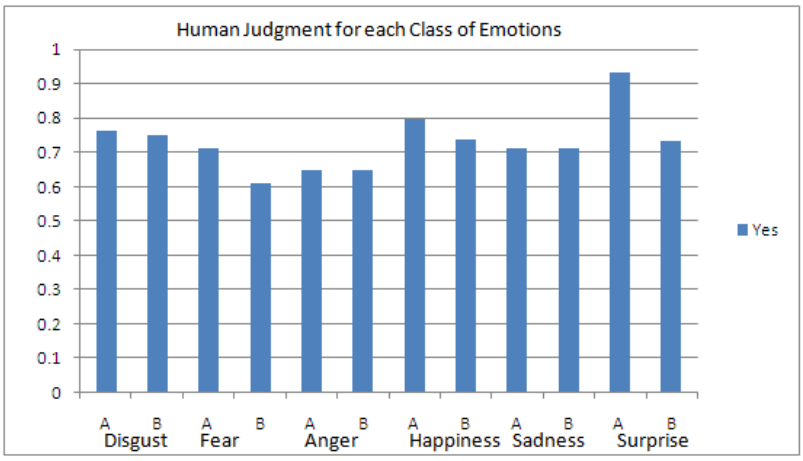

Figure 4.3: The correctness results according the judge A and judge B, for each class of emotion.

4.5.2 Estimating Recall

Evaluating the *Recall* of our algorithm is difficult, because the algorithm does not include all English words; it can only detect paraphrasing relationships between words in our corpus. Moreover, a direct comparison using an electronic thesaurus such as WordNet is not feasible, because WordNet contains synonym sets between words, and very few multi-word expressions. Thus we decided to estimate recall manually, by having a human judge extract paraphrases from a sample of text. We randomly selected 60 texts (10 for each emotion class) and asked a judge to extract paraphrases from them. For each emotion class, the judge extracted expressions that reflected the emotion, and then combined pairs that were conceptually equivalent. Since this process was very time-consuming and tedious, it was not practical to ask a second judge to do the same task.

With respect to information retrieval, *Precision* and *Recall* are defined in terms of a set of retrieved documents and a set of relevant documents. In the following sections, we describe how we computed the *Precision* and *Recall* for our algorithm, compared to the manually extracted paraphrases.

For precision, we compared the number of paraphrases that were extracted by the algorithm from the same texts, with the number also extracted by the human judge (see

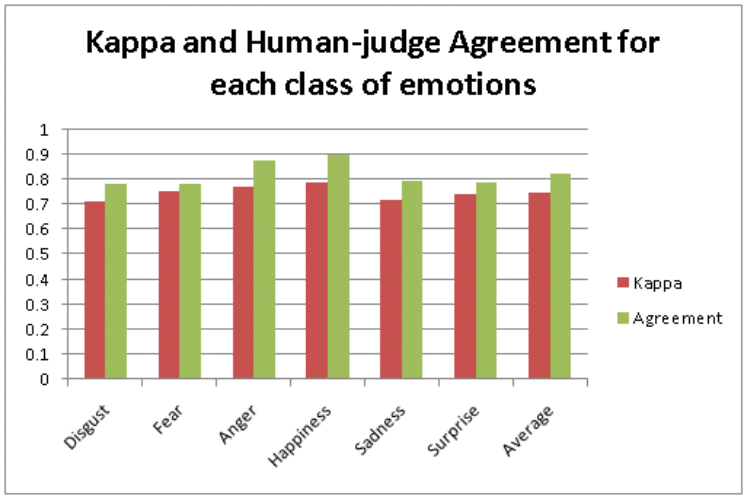

Figure 4.4: The *Kappa* coefficients and the agreement between the two human judges.

Equation 4.2). On average, from 89 paraphrases extracted by the algorithm, 74 were identified as paraphrases by the human judge (84.23%). See Table 4.9 for the values for all the classes

$$P = \frac{\#\ Correctly\ Retrieved\ Paraphrases\ by\ the\ Algorithm}{All\ Paraphrases\ Retrieved\ by\ the\ Algorithm} \tag{4.2}$$

To compute the recall, we counted how many of the paraphrases extracted by the human judge were correctly extracted by the algorithm (See Equation 4.3).

$$R = \frac{\#\ Correctly\ Retrieved\ Paraphrases\ by\ the\ Algorithm}{All\ Paraphrases\ Retrieved\ by\ the\ Human\ Judge} \tag{4.3}$$

As mentioned, evaluating *Recall* is difficult for a human judge, so we provide an explanation of how a judge extracts the paraphrases from texts: We gave the judge several randomly selected texts for each emotion class, from which to extract paraphrases. In the following example, the random text we gave the judge is from the Happiness class:

‘‘Then the maiden looked at them and recognized her brothers, was glad and crept forth from beneath the bed. When the cloth was laid, the Lord sat down with the man and his wife, and he enjoyed their coarse food, for there were

Table 4.9: Precision and Recall for a sample of texts, for each category of emotion, and their average.

Category of Emotions	Precision	Recall
Disgust	82.33%	92.91%
Fear	82.64%	88.20%
Anger	93.67%	80.57%
Happiness	82.00%	90.89%
Sadness	82.00%	89.88%
Surprise	79.78%	89.50%
Average	84.23%	88.66%

happy faces at the table. But the moment he kissed her she opened her eyes and awoke, and smiled upon him; and they went out together; and soon the king and queen also awoke, and all the court, and gazed on each other with great wonder. Then the children went home together, and were heartily delighted, and if they have not died, they are living still. Hans was delighted as he sat on the horse, drew himself up, squared his elbows, turned out his toes, cracked his whip, and rode merrily off, one minute whistling a merry tune, and another singing, ‘‘How lighthearted I feel,’’ said the father, ‘‘so pleased and cheerful.’’ The giant was pleased with the good cheer, and ate and drank to his heart"s content. So Master Thumb stayed at home with his father and mother, in peace; for though he had been so great a traveller, and had done and seen so many fine things, and was fond enough of telling the whole story, he always agreed that, after all, there’s no place like HOME!’’

For extracting the paraphrases, the judge performed the following procedure. First, emotion expressions were extracted from the whole text, then pairs of two expressions were formed in such a way that they can be considered paraphrases. The expressions

extracted from the above text are: “was glad”, “enjoyed”, “happy”, “smiled”, “gazed”, “great wonder”, “heartily delighted”, “delighted”, “merrily”, “whistling a merry tune”, “singing”, “lighthearted”, “pleased and cheerful”, “pleased with the good cheer”, “to his heart’s content”, “in peace” and “fond enough”.

The judge decided that the following seven pairs were paraphrases: “whistling a merry tune :: singing”, “pleased and cheerful :: pleased with the good cheer”, “was glad :: delighted”, “lighthearted :: happy”, “glad :: happy”, “lighthearted :: to his heart’s content”, “happy :: delighted”.

From the seven paraphrases extracted by the judge, our system identified five as paraphrases. The two not detected by the system were: “whistling a merry tune :: singing” and “lighthearted :: to his heart’s content”. The system also detected two paraphrases which were not detected by the judge: “glad :: like” and “happy :: like”. These are considered wrong by the human judge.

4.5.3 Error Analysis

We looked at some of the extracted paraphrases, particularly the samples which the human judges considered correct. A few examples are listed in Table 4.7.

There are several comments to be made. Some paraphrases are not grammatically correct, but they have the right semantic content from the point of view of the expressed emotions. In such cases, the human judges considered the paraphrases correct. Due to their informal nature, the blog texts contain a lot of ungrammatical sentences, causing our paraphrase extraction algorithm to extract some awkward paraphrases (e.g., damn being sick::am getting sick).

We chose to keep these paraphrases in our result database. Later on, when we use the paraphrases in natural language generation applications, we will employ an additional module, such as language model built on a very large corpus, to ensure that we do not generate awkward sentences.

Some paraphrases contain offensive language, such as the last word in the pair: *upset and angry::angry and pissed*. This is again due to the informal aspect of the blog texts.

These paraphrases can be filtered out, when they are used in an application that prohibits such wording in the generated language. We also can use the filtering of the offensive expressions using a system from Razavi et al. (2010).

Among the paraphrases considered incorrect by the human judges, some have the expected parallel structure but the meanings of the words are too different. For example, the conjunctions of adjectives *bitter and spite::tired and angry* were not considered correct paraphrases because the second also implies "tiredness", while the first does not.

Other errors could be caused by expressions with the exact same seed, but surrounded by contexts that are not parallel. For example,*not necessarily fear::despite your fear*. In general, these contexts will not have high strength, but some of them could be strong enough to be used by the algorithm.

4.5.4 Discussion and Comparison to Related Work

To the best of our knowledge, no similar research has been done involving extracting paraphrases for emotion terms from corpora. However, Barzilay and McKeown (2001) did comparable work for corpus-based identification of general paraphrases from multiple English translations of the same source text. We examined the pros and cons of our method versus their method. The advantages of our method are:

- The corpus does not need to be parallel. Our algorithm uses the entire corpus together to construct its bootstrapping method, while in (Barzilay and McKeown, 2001) the parallel corpus is needed in order to detect positive contexts. Since we construct the candidate contexts based on the k-window approach, there is no need for sentences to be aligned. In (Barzilay and McKeown, 2001) sentence alignment is essential, in order to recognize equivalent words and positive contexts. The Barzilay and McKeown (2001) algorithm must find positive contexts before it can look for appropriate patterns to extract paraphrases. Thus, if equivalent words do not occur in the aligned sentences, the algorithm fails to find positive contexts. Our algorithm starts with given seeds, which allows us to detect positive context

with the k-window method.

- The experiments and evaluation indicate that our bootstrapping algorithm and paraphrasing method achieve well and produces reasonable outcomes.

Glickman and Dagan (2004) investigated the extraction of lexical paraphrases for verbs from a single corpus. Their method identifies isolated paraphrase instances in a single corpus, without relying on any a priori structure and information. Their algorithm used a syntactic parser to identify the syntactic structure of the corpus sentences, and to identify verb instances. They treated the corpus uniformly as a set of distinct sentences, regardless of the document or paragraph the sentences belonged to. For each verb instance, they extracted the syntactic components that were directly related to the verb, in the parse tree. They conducted their experiments on the first 15 million word/token subset of the Reuters Corpus. They also used human judges for evaluation. Their correctness was evaluated at 61.40%, the agreement between the judges was 0.63, and the *Kappa* value was 0.61.

Their approach has some limitations and disadvantages, such as it can only extract paraphrases for verbs. The algorithm depends on verifying whether two verb instances are likely to be paraphrases that describe the same event. Although it uses a single corpus (as opposed to parallel), to allow the extraction of verb paraphrases the syntactic structure of the sentences must be about the same events. Our approach is broader, since it works for all types of lexical paraphrases.

A limitation of our method is the need for initial seed words. However, obtaining these is not a problem, as they can be found in online dictionaries, WordNet Affect and other lexical recourses.

4.6 Summary

In this Chapter, we introduced a method for corpus-based extraction of paraphrases for emotion terms. We developed a procedure that uses a bootstrapping technique based

on contextual and lexical features, that can successfully extract paraphrases using a non-parallel corpus. We showed that a bootstrapping algorithm based on contextual surrounding context features of paraphrases achieves good performance results on our data set. The evaluation suggests that our approach, based on algorithms that are adapted to extract many of the paraphrases that are typical for a multiple corpus, is clearly achievable when measured against similar methods. Moreover, a preliminary comparison suggests that the paraphrase extraction algorithm used in our bootstrapping method is more reliable than those based on global alignment, instance-based or vector similarity approaches.

In future work, we will extend this technique to extract paraphrases from more corpora and for additional types of emotions. In terms of evaluation, we will use the extracted paraphrases as features in machine learning classifiers that sort candidate sentences into classes of emotions. If the results of the classification are good, the extracted paraphrases are of high quality. Future research is planned to extend the approach to handle more complex paraphrase structures, and to increase performance by relying on additional sources of evidence.

Chapter 5

Natural Language Generation and Authoring Tool

5.1 Introduction

Natural Language Generation (NLG) is considered a sub-field of artificial intelligence and computational linguistics. The focus of NLG is to build computer systems able to generate text from non-linguistic information, in natural language understandable to users. The task of NLG systems is to merge knowledge about language and the application domain, and automatically generate different types of texts, including reports, documents, guides and help messages (Reiter and Dale, 2000).

Since our goal is to generate emotional text, in this chapter we describe the modules that must be in place to implement an NLG system, and possible algorithms and supporting representations for each module. We also explain our Authoring Environment Tools for the Automatic Generation of Narrative Content (Caropreso et al., 2009a). We developed an automatic tool that provides access to SimpleNLG (Gatt and Reiter, 2009), in order to generate sentences with variable parts or templates. We implemented this NLG Template Authoring Environment tool based on templates required for generating content for a digital-based interactive simulation game. Our goal was to provide a tool to help low-skilled programmers generate text based on variables provided in template-

based format. We also wanted to provide game content designers with an accessible tool they could use to create and manipulate the NLG templates, and thus generate sentences that would support the narrative progression of the game.

As mentioned in Section 2.5, two approaches to NLG are most widely used: "template-based" and "deep-linguistic" (Gagn and Briggs, 1997). The template-based NLG systems provide scaffolding in the form of templates with a predefined structure, and perhaps some of the final text. The deep-linguistic approach attempts to build sentences from a logical representation, and is designed to be flexible and capable of expressing any sentence, given a valid logical input form. However, adoption of these systems has been constrained by the sophistication of the grammar system required, and the steep learning curve for writing logical forms. An example of this type of system is KPML (Bateman, 1997).

5.1.1 Template-based NLG Systems

There are many ways to represent the messages that form the leaf nodes of a text plan, including logical formulae and templates (Reiter and Dale, 2000). Template systems represent sentences as boilerplate text, and parameters that need to be inserted into the boilerplate (Geldof and de Velde, 1997).

Some NLG systems do not perform syntactic realisation; the determination process directly specifies messages as text templates. Some of the advantages of the template approach are as follows:

- They are useful when the output texts require only limited syntactic variability. For example, if all the sentences to be generated are in simple present tense, there is no need for a complex realisation mechanism that can generate sentences in other tenses.

- Another advantage of the template approach is that it can assist domain experts. Templates are usually easier to understand than mechanisms that manipulate more complex syntactic structures.

5.2 Requirements and Specification for NLG System

When developing any software system, including a natural language generation system, the first step is to conduct a requirements analysis, and then create the initial system specification. However, for NLG systems, iterative development prototyping methodologies work better than waterfall-type systems (Reiter and Dale, 2000). McKeown et al. (1995) investigated the use of corpora in building NLG systems as well.

Since we use corpora as input for our analysis of texts with emotions, in the following we review corpus-based approaches to NLG systems that were introduced by Reiter and Dale (2000).

5.2.1 Initial Corpus of Output Texts

The first step in conducting a corpus-based requirements analysis is to create an initial corpus of human-authored texts and inputs. In the simplest case, the corpus can be created using archived examples of human-authored texts. We use the blog corpus, as well as the three other data sets we described in the previous chapters.

5.2.2 Creating a Target Text Corpus

In some cases, NLG systems modify the content of the initial corpus. This happens for various reasons, such as data not being available in the right format. Therefore, the target text corpus is the result of all the changes, based on set of texts. It should characterize the output that can be generated by the NLG system.

5.3 The Architecture and Components of an NLG System

After meeting the requirements above, the NLG system is ready to be designed. In this section, we describe different architectures and representations of the tasks that NLG

systems perform.

5.3.1 Natural Language Generation Tasks

The important function of any Natural Language Generation system is to map input data to output text. Reiter and Dale (2000) split this task into a number of different steps. In the following, we briefly explain some of these tasks.

Content determination

Content Determination establishes what information must be communicated in the text. This process builds a set of messages from the inputs or data sources to the system. The messages can then be used by the subsequent modules.

Discourse planning

Discourse Planning is the process of organizing and adding structure to the set of messages to be processed. It renders the output generated text more clear and easy to read, and it ensures that each sentence has a beginning, a middle and an end.

Sentence Aggregation

Sentence Aggregation is used to combine and build the group messages into sentences.

Referring expression generation

Referring Expression Generation is the task of selecting words or phrases to identify domain entities. It also chooses words to identify a particular domain entity.

Linguistic realisation

Linguistic Realisation applies the rules of grammar, to generate a sentence which is syntactically and morphologically correct.

Lexicalization

An important task in NLG is to choose the correct words. Lexicalization determines which words should be used to describe particular domain concepts or entities.

5.3.2 NLG Architectures

There are different ways to create a Natural Language Generation system to perform the above tasks. The easiest way is to build the module for each task separately. In Section 2.5 we introduced another architecture, which was proposed by Dalianis (1996), and is illustrated in Figure 2.2.

However, Reiter and Dale (2000) introduced the most common architecture for NLG systems from an application point of view. Figure 5.1 illustrates the three stage pipeline of this architecture.

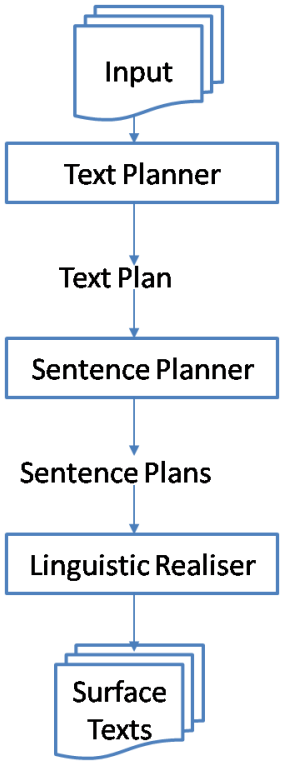

Figure 5.1: Architecture of NLG systems introduced by Reiter and Dale (2000).

Text Planning

This stage combines the content determination and discourse planning tasks described above. In many applications it is difficult to separate these two activities.

Sentence Planning

This stage merges sentence aggregation, lexicalization and referring expression generation. Some researchers, such as Matthiessen (1991), do not agree with this grouping, believing that lexicalisation should be combined with linguistic realisation. However, most NLG systems combine these three tasks into one stage, and we use this approach in our research.

Linguistic Realisation

As described above in Section 5.3.1, this task involves syntactic, morphological, and orthographic processing.

5.4 Our Authoring Environment Tool

5.4.1 Introduction

We created an environment that simplifies the use of SimpleNLG to generate sentences with variable parts or templates. We developed this NLG Template Authoring Environment in order to provide the templates required to generate content for a digital-based, interactive simulation game. The goal of this project was to give game content designers an accessible tool they could use to create and manipulate the NLG templates, and thereby generate sentences that would support the narrative progression of the game. The NLG Template Authoring Environment presents a model sentence, and allows the user to identify the variable (i.e. dynamically generated) sections, which also serves to implicitly 'lock-down' the static elements of the sentence. The content author can subsequently highlight the dependencies between the variable elements. The system then

produces a list of all possible sentences that could be created from the given model with the specified variables and dependencies. By consulting this output, the user can refine the template model, adjusting it to the requirements.

The rapid adoption of information retrieval tools has been supported by the availability of an iterative process of identifying the information need of the user via the progressive refinement of supplied keywords, in response to the display of search results. Our hope is that a similar process would hasten the adoption of NLG tools by the wider information processing community as well. We are developing a similar process that can support the NLG system users to iteratively improve the templates that they create. The key value provided by this process is that the template is created in a natural manner. The symbolic expressions are created behind the scenes based on the user choices. Thus, non-experts can improve their template in a "try-and-see" iterative fashion without needing to build a complete theoretical representation of the template before seeing any results. Although experts may still prefer to approach the NLG template generation task using more powerful symbolic representation, this system is very valuable for the software developers, narrative content writers and industrial engineers who would not be sufficiently motivated to develop NLG template writing skills in order to employ automatic text generation during the creative process. Thus, this NLG Template Authoring Environment has been designed to help individuals who would not normally be able to use this technology, due to the specialized skills required to do so (Christensen and Raynor, 2003).

The design and performance evaluation of the NLG Template Authoring Environment was guided by the requirements of a digital-based, interactive simulation game. A set of sentence templates about different aspects of categories was selected from the templates that were manually designed for that game. These were then recreated using our system, which had to be iteratively adapted until it was possible to implement all the aspects of the templates. The system usability was then tested by comparing the performance achieved and the time required by the game content writers to obtain the necessary sentences for two simple negotiation games, both manually, and by using the

NLG Template Authoring Environment.

5.4.2 SimpleNLG

SimpleNLG is an NLG system that allows the user to specify a sentence by providing the content words and their grammatical roles (e.g., subject or verb). The specification can be conducted at different levels of detail. For example, "the black cat" could be specified as a noun phrase with no further details, or a noun phrase where "cat" is the head of the phrase, "black" is a modifier and "the" is the determiner. SimpleNLG is implemented as a Java library, and thus requires some Java programming knowledge. The following shows the code needed to generate the sentence "My dog chases George." In this example, a new phrase specification is created, the subject, verb, and complement are set, and, finally, the document is realized and the sentence is displayed.

```
Example 1:
SPhraseSpec p = new SPhraseSpec();
p.setSubject(''my dog'');
p.setVerb(''chase'');
  p.addComplement(''George'');
Realiser r = new Realiser();
System.out.println(r.realiseDocument(p));
```

SimpleNLG automates several tasks, including orthography, morphology and grammatical realization. For the latter, it employs grammar rules to convert abstract representations of sentences into actual text. The tasks performed to generate the sentence in Example 1 above were the first letter of the sentence was capitalized, the verb was set in agreement with the subject, all the words were put together in grammatical form, white spaces were inserted in the appropriate places between words, and a period was added at the end of the sentence. SimpleNLG also permits the user to specify several features for the main verb, including tense (present, past or future), whether it is subjective (verbs indicate willingness, emotion, doubt and denial), whether it is in progressive,

passive or perfect form, whether it is in interrogative form, whether it is negated, and which, if any, modal to use (i.e., could, must). Though some of these features affect the verb only, others affect the structure of the entire sentence, such as when it needs to be expressed in the passive voice. The programming nature of SimpleNLG allows the user to define flexible templates by using programming variables in the sentence specification. As previously discussed, templates are frames that define sentences with fixed and variable components, and the variable components could have different values. When templates are used without an NLG system they are known as "canned text", and have the disadvantage of being inflexible because only predefined variables can change. When templates are defined using SimpleNLG, however, they retain all the functionality of the NLG system (e.g., being able to modify the verb features or the output format, and making use of grammatical knowledge), while allowing for the variable values to change.

5.4.3 NLG Template for Authoring Environment Tools

We developed a Natural Language Generation Template Authoring Environment to provide game designers with the capability to produce the sentence templates they could use to generate sentences for their games. In the context of creating sentence templates for games design, this system bridges the gap between the game designers' content knowledge, and the expertise required to make use of NLG systems. The environment allows the user to create an example sentence, define which parts would be variable and the possible values, and specify dependencies between the variables. It then produces the sentences that could be generated from the given template, by calculating all the possible combinations of variable values within the specified dependencies. The user can then refine the template, by changing either the given example or the specified variables and dependencies, in order to tailor the generated sentences to the game requirements.

Design

Figure 5.2 presents a graphical design for the NLG Template Authoring Environment, using the simple example of a sentence in which three variables and a dependency are

specified. As shown, the system allows the user to input an example sentence with an identified main verb, a subject, and a complement (see the text in the respective boxes). In addition, information about the verb (i.e. tense, form, modals) can be specified; the defaults are present tense, non-progressive form and active voice. The user has the choice of changing these options or adding others. For example, they could add the past and future tenses to the default selected present tense. These verb options are presented in a separate options window.

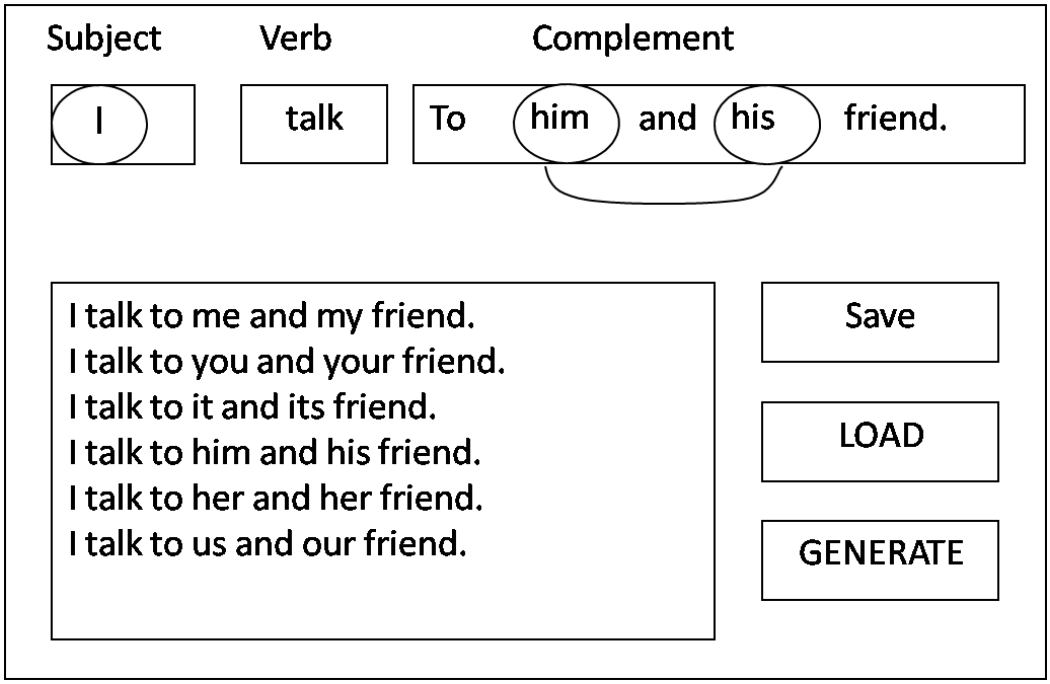

Figure 5.2: NLG Template Authoring Environment.

The system allows the user to identify variables in the subject and the complement of the sentence (see the ovals around some of the words in the text boxes). The user is required to indicate the type of each specified variable (e.g., personal pronoun, possessive pronoun, employee type), and which values of that type are allowed (e.g., all personal pronouns, only "she" and "he"). The user can also indicate the dependencies between variables (see the linking of "him" and "his" in the example). All the information provided to create a template (i.e., the example sentence and its variables and dependencies) can be saved and recovered later, using the provided "Save" and "Load"

utilities. The "Generate" utility creates new sentences according to the template indicated in the example sentence, and presents them to the user. These sentences are the result of combining the values of the variables and the verb options (when more than one was specified) in all possible ways, while respecting the dependencies between variables.

Implementation

The NLG Template Authoring Environment was implemented in Java. The SimpleNLG library was used to automatically generate correct sentences, and to provide the user with the capability to explore different attributes of the verb. The example sentence is parsed, then stored in a structure that keeps the static and variable components separate (with information about where the variable components connect with the static ones). The variables are represented by objects which store the necessary information (e.g., variable type, default value, current value, gender and number of the current value) that is used when generating all the viable combinations. The variable type references a text file containing all the possible values and their respective syntactic information (person, number and gender), which will be tested for agreement with the verb, and for dependency between the variables' purposes. Each variable object can update its current value with the next permitted (unfiltered) value of its type. For example, a variable of type 'personal pronoun' with current value "I" will update to "you" when required by calling the instruction 'getNextValue', unless "you" has been filtered from the variables' values, in which case the next permitted value will be used. This is the method that generates all possible combinations of the variables' values, by accessing the list of variables and instructing each of them to update their value. This method also checks the list of dependencies, in order to filter out the pairs that do not satisfy them.

Once a combination of values for all the variables is generated and considered to be a valid choice (after filtering according to the dependencies), the static and variable parts of the sentence are reunited, and then the SimpleNLG package is applied in order to realize the sentences. At this stage, all required modifications to the verb are performed, and several possibilities could be displayed, depending on the choice of verb options. For

example, if the user has indicated present, past and future as the verb tense options, three sentences, one for each tense, will be displayed for the current combination of variable values.

SimpleNLG requires the infinitive of a verb, so a separate module is internally called before passing a verb on to it. This module uses WordNet (Felbaum, 1998) to convert the main verb in the sentence example to its infinitive form. The mechanism that generates all possible combinations of the variable values is currently implemented recursively. In the future, this will be replaced by a more effective method which uses dynamic programming. The current system does not perform syntactic analysis of the template model provided by the user. The different parts of the example sentence are given to SimpleNLG for generation. It is assumed that these are composed of English words of the expected syntactic type. The future design will perform a syntactic analysis and validation of the user's input.

Interface

A user-friendly intuitive graphical interface has also been implemented in Java, using the "Swing" library (a partial screenshot is shown in Figure 5.3. When using this interface, the user first enters an example sentence and clicks on 'Analyze', then identifies a section as variable by assigning a type or semantic class to the word in that section. As discussed previously, the values of a semantic class are stored in a text file, which allows the user to create new semantic classes as required. Restrictions to the values that a variable can have are also indicated by the graphical interface. Dependencies can be assigned only between already declared variables. The main verb and all its options are indicated in the lower section of the interface. In Figure 5.3 , the example sentence is "I walk my dog." "I" is a variable of type Personal Pronoun, "walk" is the main verb, "my" is a variable of type Possessive Pronoun, and "dog" is a variable of type Animals. There is a dependency between "I" and "my" (which allows the user to make their values agree in person, number and gender when generating all possible combinations).

In Figure 5.3, we also see that the user has selected the values "present and past"

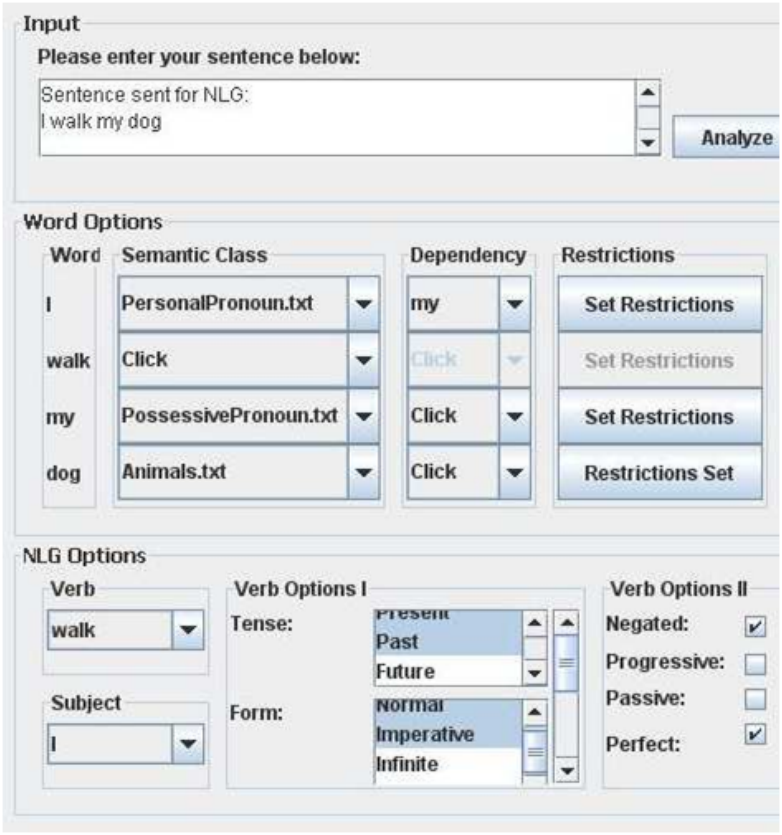

Figure 5.3: Graphical Interface.

for the verb tense, and “indicative” and ”imperative” for the verb form. Therefore, four sentences will be generated for each combination of the variables’ values (i.e. one sentence for each combination of the four tense and form selections). All the sentences will have the verb negated and will use the perfect tense, as indicated by the verb options in the lower-right column. The interface also outputs an XML file containing the template information, which is used for communication purposes between the graphical interface and the generation system. The generated templates in XML format are available for review and possible editing by linguistic experts. The modified templates can also be provided directly to the generation system, allowing computational linguistic experts to manually create templates for the sentence generation system. Another system application has recently been developed, and is also available through the interface. It allows the content writers to automatically generate a file containing all the sentences that will

Templates
1. The ACTORS (ME/US) could help DEPARTMENTS.
2. The ACTORS IS/ARE now available to help.
3. I/WE struggled because of MY/OUR lack of knowledge.
4. I/WE AM/ARE pleased to report that I/WE completed the task TASKS.
5. I/WE WAS/WERE not the greatest choice for keeping things moving along quickly.

Table 5.1: Testing examples.

be used as feedback for a particular game, formatted specifically to meet the game requirements. It acquires the sentences from all the templates created and approved by the user.

5.4.4 Testing the System's Capabilities

In order to verify that the NLG Template Authoring Environment was functioning correctly, we selected a set of sentence templates from the game ISO 14K. The templates were chosen manually, taking into account the need to include different aspects; for example, the number and type of the variables and dependencies. The testing of these examples addresses many more templates of the same type. The five selected sentence templates that formed our test set are displayed in Table 5.1 , and are identified from this point on by their reference number or order in the table. In the template examples, we show the variable parts of the templates in capitals. ACTORS, DEPARTMENTS and TASKS refer to one of several possible nouns previously defined for each of the classes with these names. The terms in capitals letters separated by a forward slash (/) display all the accepted values for that variable (e.g., I/WE represents a variable of type personal pronoun which could take only the selected values "I" or "we", as the rest have been filtered out).

5.4.5 Testing the System's Usability

In order to test the Template Generation Authoring System's usability, we have trained three users. We gave them an introduction to templates and the system's general goal. We explained the meaning of semantic classes, variables, and dependencies. We described the different generation options that could be passed to SimpleNLG and what changes would they produce on the resulting sentences. We finally showed them the interface and created some example templates together. This whole training took around an hour, after which they were able to successfully create and iteratively refine their own templates. After allowing them to experiment with the system, they were asked to complete the evaluation questionnaire found in Appendix A.3. According to their answers, they found the system easy to use and they only needed a day to familiarize themselves with the different interface options and to feel comfortable using it.

For the purpose of further testing the usability of our system, we have designed two simple negotiation games. The first game consists of negotiating the sale of a house from either the buyer or the seller perspective. The second game consists of negotiating the salary and benefits of a job offer, from either the applicant or the employer perspective. The three users were asked to produce the texts for the two games, manually (scenario 1) and using our system (scenario 2). The two scenarios used different perspective, so that the texts to be created were not exactly the same, but comparable in difficulty level. Then the users were asked to fill in the questionnaire from Appendix A.4.

The content writers were assigned by the system's trainer during the generation of the sentences for the first game. They used the system unassisted for the second game. The time invested and the erors made by the users when generating content for the negotiation game are shown in Appendix A.5.

5.5 Summary and Conclusion

In this chapter, we reviewed and identified the needs and components of a Natural Language Generation system for our research. We briefly explained:

- the pros and cons of building natural language generation systems;
- some of the techniques that can be used to determine the requirements to be met by such systems;
- the tasks that NLG systems need to accomplish; and,
- the details of one particular architectural model that accomplishes these tasks.
- our authoring tool for generation texts using templates.
- the evaluation of our authoring tool.

Acknowledgment

The authoring tool is a joint work with Maria Fernanda Caropreso. I thanks for her contribution toward this chapter.

Chapter 6

Generating Text to Express Emotion

6.1 Introduction

In this chapter, for generation of text with emotions, 1) we can use the Authoring Tool from Chapter 5 (we will explain later in this chapter how the tool is extended for emotion generation); 2) we introduce a pattern-based model; the user can choose a pattern, then based on the pattern variable fields, the tool will generate emotion sentences.

6.2 Our Models to Generate Emotion Sentences

The vast amount of text that is becoming available online offers new possibilities for achieving corpus-based approaches to NLG systems. Most corpus-based systems rely on a text corpus that has been manually tagged in some way. The Brown Corpus (Francis and Kucera, 1982) and the Penn Treebank corpus (Marcus et al., 1993) for example, are widely used because they have been manually annotated with part-of-speech (POS) tagging. These corpora can be used by different natural language processing systems for different domains. However, some corpus-based systems rely on a text corpus that has been manually tagged in a domain-specific or task-specific manner. For example, corpus-based approaches to information extraction generally rely on special domain-specific text annotations. Consequently, the manual tagging effort is considerably less cost effective,

because the annotated corpus is useful for only one type of system and one domain. Corpus-based approaches to information extraction have demonstrated significant time savings over conventional hand-coding methods (Riloff, 1996), but the time required to annotate a training corpus is non-trivial.

To further reduce this knowledge-engineering bottleneck, we have developed a "pattern-based model", a system that generates extraction patterns using untagged text. Our model requires only a pre-classified corpus of relevant texts; nothing inside the texts needs to be tagged in any way. Our goal with this model is to extract emotion patterns from the corpus, then find the best patterns to describe semantic representations for each emotion category. For example, our model uses a semantic representation of the input sentence to be conveyed in a different realization sentence.

During our research, for paraphrase extraction in Chapter 4, we recorded the parts of speech (PoS) tags, and the words surrounding the emotion seeds. We extracted pronouns and open-class lexical terms (nouns, verbs, adjectives and adverbs) before and after each seed, for each class of emotion. These POSs have advantages in our model: first, to find paraphrases and context similarity for paraphrase extraction; and second, to find the behavior and the formation of POS and words that surround each emotion, which is our goal in this chapter. This also helps determine the construction of emotion sentences. Figure 6.1 illustrates the interaction between input sentences taken from a corpus, the pattern extraction and analysis, planning the sentence by using the patterns (or templates), and the output text produced by the surface realization engine.

Although based on a small set of examples, the combination of sentence, noun, verb, adjective and adverb patterns with the ability to change individual values, could allow the application to generate a range of different sentences.

6.2.1 Data Set

Our aim is to determine what types of words people use when they express their emotions and, in general, what kind of POS they use in their daily writing, particularly with respect to pronouns, nouns, verbs, adjectives and adverbs. Based on these findings, we aim to

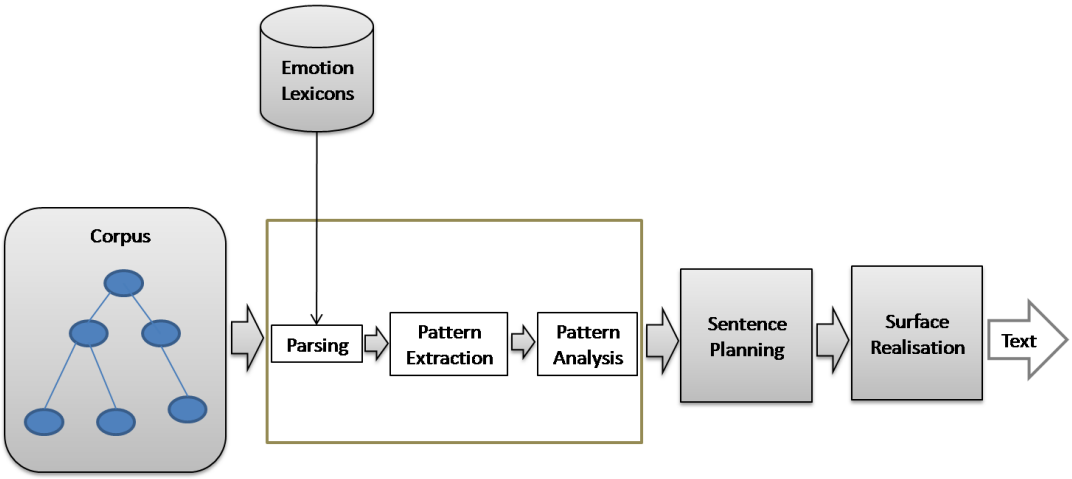

Figure 6.1: Our Pattern-based Architecture for Sentence Generation.

build a pattern-based model which can generate emotion sentences.

The starting point of our work was to provide a corpus of emotion sentences for the six emotion categories we used in our research (i.e., joy, anger, sadness, fear, surprise and disgust). To do this, we used the same data set that we collected for paraphrase extraction (explained in Chapter 4). Each sentence in the target corpus might contain an emotion word, which can be used to produce a pattern of the original sentence.

6.2.2 Part-of-Speech Tagging and Tokenization

In order to construct our pattern-based model, we preprocessed the corpus using part of speech tagging and word tokenization.

In our experiments, parts of speech tagging is done using the POS tagger developed by Klein and Manning (2003). First we reviewed Table 4.5, and from the features in this table we used pronoun, noun, verb, adverb and adjective before and after emotion seeds, along with their tokens, for each emotion category. These features are shown in Table 6.1.

This step produced a syntactic analysis for each sentence, and identified the above parts of speech tags. For each POS we introduced some heuristic rules to extract original patterns. These features help to extract patterns.

6.2.3 Pattern Extraction

We extracted the features shown in Table 6.1, which allowed us to determine the frequency of these features in the corpus.

We transferred the tokenized sentences into a vector space of features. We considered features that include both lexical and syntactic descriptions of all the sentences. The lexical features include the sequence of tokens for each phrase in the paraphrase pair.

Table 6.2 displays the statistics for four lexical classes of POS. It shows that pronouns, nouns, verbs, adverbs and adjectives are most frequent features in our data set. It also shows that, on average, 65% (out of 100% before seeds) of these features appear before emotion seeds, and 54% (out of 100% after seeds) of the time they appear after emotion seeds. In the table, L (Left) and R (Right) are parts of speech or tokens before emotion seeds, and parts of speech or tokens after emotion seeds, respectively. Previous research on word sense disambiguation in contextual analysis has identified several local and topical features that are good indicators of word properties. These include surrounding words and their parts of speech tags, collocation and keywords in contexts (Mihalcea, 2004). Other features have also been proposed, including bigrams, named entities, syntactic features and semantic relations with other words in the context.

We analyzed the formation and construction of pronouns, nouns, verbs, adjectives and adverbs for each of the emotion categories. Based on this, we extracted the initial patterns that represent emotion categories and are good candidates to generate better patterns, and which will also help generate emotion sentences.

Pronoun Formation in Emotion Sentences

As shown in Figures 6.2 and 6.3, the most frequent parts of speech before and after the emotion seeds are pronouns. The statistics show that “I”, “me” and “my” are the pronouns used most often. Thus, it can be concluded that people usually use emotion sentences to refer to themselves. Table 6.4 presents some of these pronouns, and it shows that pronouns are used after the emotion seeds 34% of the time.

Features	Description
F1	Sequence of PoS and Tokens of the Sentence
F2	First Verb before the Seed
F3	First Noun before the Seed
F4	Pronoun before the Seed
F5	Seed
F6	Pronoun after the Seed
F7	First Noun after the Seed
F8	First Verb after the Seed
F9	First Adjective after the Seed
F10	First Adverb after the Seed

Table 6.1: The features that we used to extract patterns.

Noun Formation in Emotion Sentences

Another part of speech that often appeared before and after emotion seeds is the noun. Some of the nouns in emotion sentences are seen in Table 6.5. Table 6.3 shows that nouns appear after emotion seeds 41% of the time, and before emotion seeds 12%.

Verb Formation in Emotion Sentences

Another part of speech that mostly appeared before and after emotion seeds is the verb. Some of verbs in emotion sentences are shown in Table 6.6. Our results show that verbs appear before emotion seeds over twice as often (27%) as than after emotion seeds (10%).

Adjective Formation in Emotion Sentences

Adjectives also appeared before and after emotion seeds, as shown in Table 6.7. Our results show that adjectives have the lowest frequency of appearance before and after emotion seeds. However, they come before or after with almost the same frequency: 7%

Emotion	Pronoun	Verb	Noun	Adjective	Adverb	Total
anger(L)	2061	1178	561	279	1250	5329(69%)
anger(R)	1607	486	1757	190	256	2689(35%)
joy(L)	2544	2275	1080	672	2487	9058(68%)
joy(R)	2415	716	2961	368	877	7337(54%)
fear(L)	2334	2062	952	613	2293	5961(50%)
fear(R)	2177	648	2575	350	832	6582(54%)
disgust(L)	2581	2444	1118	699	2570	9412(68%)
disgust(L)	2441	734	3036	378	921	7510(54%)
sad(L)	2858	3244	1353	905	3204	11564(68%)
sad(R)	2889	881	3642	479	843	8734(51%)
surprise(L)	2111	1316	602	312	1384	5725(51%)
surprise(R)	1688	508	1911	222	290	4619(56%
Total(L)	14489	12519	5666	3480	13188	47049(65%)
Total(R)	13217	3973	15882	1987	4019	39078(54%)

Table 6.2: The Frequency of the Pronouns, Verbs, Nouns, Adjectives, and Adverbs for each class of emotion in our data set.

and 5%, respectively.

Adverb Formation in Emotion Sentences

The last parts of speech we examined was the adverb. Table 6.8 shows some of the adverbs in our data set. The result for adverbs shows that they appear before and after emotion seeds more than adjectives; 28% before emotion seeds and 10% after.

Initial Patterns

Based on the results shown in the previous sections, we were able to determine *initial pattern* for our system. We considered parts of speech before and after emotion seeds, and we included the emotion seeds in an initial pattern. For example; "N ES V" (N:

Emotion	Pronoun	Verb	Noun	Adjective	Adverb
anger(L)	27%	15%	7%	4%	16%
anger(R)	37%	11%	41%	4%	6%
joy(L)	28%	25%	12%	7%	27%
joy(R)	33%	10%	40%	5%	12%
fear(L)	39%	35%	16%	10%	38%
fear(R)	33%	10%	39%	5%	13%
disgust(L)	27%	26%	12%	7%	27%
disgust(R)	33%	10%	40%	5%	12%
sadness(L)	25%	28%	12%	8%	28%
sadness(R)	33%	10%	42%	5%	10%
surprise(L)	37%	23%	11%	5%	24%
surprise(R)	37%	11%	41%	5%	6%
Total(L)	31%	27%	12%	7%	28%
Total(R)	34%	10%	41%	5%	10%

Table 6.3: The Percentage of POS (Pronoun, Verb, Noun, Adjective, Adverb) for each class of emotion in our data set.

Noun, ES: Emotion Seed, V: Verb) is an *initial pattern*. Based on above definition, we can extract all initial patterns surrounding the emotion seeds.

6.2.4 Pattern Analysis

In this section, we analyze the patterns and explain how we can construct "final" patterns from extended patterns by extending the initial patterns. In the following, we define the notations of extended patterns and final patterns.

anger	joy	fear	disgust	sadness	surprise
I/it	I/it	I/it	I/it	I/I	I/it
me/my	my/I	my/I	my/I	my/it	my/my
my/you	me/me	me/me	me/me	me/my	me/you
it/I	it/my	it/my	it/my	it/me	it/I
you/me	you/you	you/you	you/you	you/you	you/me
they/them	her/them	they/them	her/them	her/them	they/them
them/her	they/her	her/her	they/her	they/her	them/her
your/him	his/him	them/him	his/him	your/him	her/him
her/myself	them/myself	your/myself	them/myself	his/ myself	your/myself
his/he	their/he	its/he	their/he	its/he	his/he

Table 6.4: The most frequent pronouns for each emotion category in our data set (the left and right of "/" means that these are the most frequent pronouns that appeared before and after seeds in each emotion category).

Extended Patterns

Since we intended to generate emotion sentences, and any sentence must have three main components (i.e., subject, verb, and object), it is difficult to construct emotion sentences from the initial patterns. Therefore, we extended the initial patterns to create larger patterns that were suitable candidates for sentence generation; we called these "extended patterns". For example, from the initial pattern `|V ES N|`, we can construct *extended* patterns such as `‘‘N V ES N’’`, `‘‘PR V ES N’’`, `‘‘N V ES JJ’’`, and many others.

However, it became clear that the *extended* patterns may not be suitable candidates for sentence generation, and so we selected *final* patterns from the extended patterns. The definition of a *final* patterns follows.

anger	joy	fear	disgust	sadness	surprise
hate/people	rose/people	hate/attack	beverages/movie	pain/people	bomb/traffic
insult/hate	cookie/life	fire/school	addiction/dictator	stage/song	mistake/fact
something/right	holiday/news	stories/management	hamburger/smoke	moment/hate	argument/rape
busses/everything	heart/school	dentist/dream	heart/movies	mission/nightmare	vote/school
bit/school	movie/money	driver/person	rope/time	nervousness/week	devil/account
panic/mood	people/girls	work/guys	period/holiday	heart/school	midterm/mood
world/day	music/food	goddamn/nightmare	people/headache	people/story	king/drinking
traffic/cause	love/thing	guy/doctors	storm/school	upset/mood	movie/report
fools/thing	parents/friend	words/life	market/class	death/idea	blood/cause
rage/person	bed/Valentine	panic/snow	home/morning	jealousy/exam	religion/kiss
everyone/life	Christmas/honey	attack/letter	games/animation	dream/party	issue/feeling

Table 6.5: The most frequent nouns for each emotion category in our data set.

Final Patterns

We call P a final pattern if it can generate a grammatically-correct sentence. To find final patterns, we started with the initial pattern that we produced in the previous section. For clarification, we explain our method with an example. Let the initial pattern p be ‘‘V ES N’’. We take p and match it to a candidate sentence from the corpus to find the extended pattern. From the pattern p we can construct the pattern P_1 ‘‘N V ES N’’. An example of a correct sentence from pattern P_1 is, “Stuff looks like heaven”. From p we can also construct pattern P_2 ‘‘PR V ES N’’, and an example of this pattern is, “It was a great idea”. Finally, in this example, from the initial pattern p we can construct the extended patterns P_1 and P_2, which are *final* patterns. As shown by the examples, the *final* patterns can generate different types of emotion sentences. Figure 6.2 illustrates the transformation of the construction of some *initial* patterns into *extended* patterns, with examples. We followed the above method and retrieved many *final* patterns from the *extended* patterns.

Based on the above examples, we now introduce our method for constructing patterns, as follows:

1. Initial Pattern: Determine some Initial Patterns.

2. Extended Pattern: Construct the extended pattern based on the initial patterns

anger	joy	fear	disgust	sadness	surprise
be/be	be/be	be/get	be/do	be/feel	grew/amaze
get/get	sound/have	afraid/be	get/see	feel/see	am/shut
hate/have	feel/make	happen/have	look/have	get/live	look/put
make/feel	/win	sound/feel	have/hate	have/get	get/feel
look/go	look/feel	get/go	stay/feel	beat/bored	know/wear
say/do	say/tell	make/do	describe/make	freak/loath	mean/scream
annoy/hate	seem/grew	scare/deal	drive/talk	look/try	hear/loose
cry/come	read/eat	ask/hate	hear/look	break/piss	help/left
anger/work	get/smile	feel/improve	born/wonder	say/harm	change/play
feel/live	take/see	hate/sleep	smell/soar	seem/turn	hate/live

Table 6.6: The most frequent verbs for each emotion category in our data set.

from step 1.

3. Final Pattern: Construct the final patterns based on the extended patterns from step 2.

4. Determine whether the Final Patterns from step 3 are suitable patterns to generate correct English sentences (the user can determine whether the final patterns are meaningful patterns).

5. Store all Final Patterns from step 4 that are able to generate sentences.

6.2.5 Sentence Planner

Sentence planning is one of the main tasks of any NLG system. It determines how the information is divided among individual sentences, and which parts (e.g., pronouns, discourse markers, verbs, etc.) should be added to make the text coherent and smooth (Reiter and Dale, 2000).

In sentence planning, we define some rules for constructing patterns from the previous section to generate emotion sentences. For example, pattern "PR V great NN" could

anger	joy	fear	disgust	sadness	surprise
little/stupid	little/slow	little/scared	little/scared	little/little	little/stupid
bad/bad	much/terrible	much/stupid	much/stupid	much/sad	bad/bad
mad/mad	scared/irrational	scared/bad	scared/bad	sad/scared	mad/enough
so/angry	bad/low	bad/enough	bad/enough	scared/stupid	so/mad
sick/enough	great/good	big/mad	great/mad	great/bad	stupid/angry
evil/little	good/first	more/angry	good/angry	bad/mad	evil/little
big/dumb	big/wicked	mad/little	big/little	so/angry	great/dumb
lawful/blue	so/lawful	great/scary	so/scary	good/last	big/blue
enough/evil	mad/true	biggest/last	mad/last	big/enough	so/evil
stupid/lawful	enough/enough	horrible/blue	stupid/nervous	lonely/scary	lawful/first

Table 6.7: The most frequent adjectives for each emotion category in our data set.

generate the sentence "He was great idea", which is not a correct sentence. So, our rules add restrictions in order to choose a correct pronoun for subject of the sentence. We also try for agreement between the pronouns and verbs, and the coherence of the sentence. We focus here on sentence planning (micro-planning) as a distinct phase of the generation process.

Pattern Selection Task

- **Transforming a pattern to a sentence format.** This will manage the chosen pattern to determine the correct format for the sentence structure, in terms of subject, verb and object formation. In this case, based on the pattern format, the user can select a subject and a verb, and the rest of the pattern will be considered as object by the system. Another option is for the system to identify the subject, verb and object based on a selected pattern, and for the user to identify the variable parts in the system. For example, take the pattern `''Pronoun Verb Emotion-Seed Noun''`. The Pronoun will be considered as the Subject, the Verb will be the verb and the Emotion-Seed and Noun will be considered as the Object. This will help the sentence realization module to generate an

anger	joy	fear	disgust	sadness	surprise
really/so	so/so	so/when	so/when	so/so	so/so
just/enough	really/just	really/how	really/how	really/not	really/then
very/then	completely/regardless	just/so	very/so	very/now	just/enough
too/just	actually/really	very/not	just/not	just/just	very/just
still/now	especially/forever	too/now	too/now	too/enough	too/now
more/even	truly/then	still/then	how/just	more/then	still/even
also/well	especially/seriously	how/just	still/then	still/right	more/too
absolutely/too	pleasantly/especially	more/why	more/too	pretty/again	also/well
even/very	incredibly/mainly	also/right	also/why	also/too	most/sometimes
most/lately	absolutely/mostly	most/enough	most/right	most/really	absolutely/really

Table 6.8: The most frequent adverbs for each emotion category in our data set.

output sentence.

- **Determining correct emotion words.** Despite choosing proper words for POS in patterns, this stage is also needed to select proper emotion words from the emotion lexicon for each sentence.

- **Lexical choice:** This task involves selecting from a set of semantically equivalent but syntactically different alternatives and entities.

6.2.6 Template-based Approach

The goal of generating sentences in NLG systems is not only to produce realization of the input semantics; the output sentences must also convey the same meaning, expressed in various ways, through different lexical and syntactic approaches. These different realizations are known as paraphrases. If an NLG system can come up with an appropriate realization, it must be able to generate similar sentences that realize the input semantics.

Our model takes a sentence as an input to be conveyed in different realization sentences. The sentence can contain variables, as explain in Chapter 5. The output of this

Initial Patterns
Noun ES Verb
Verb ES Noun
Pronoun ES Verb
Verb ES Pronoun
Adjective ES Verb
Adverb ES Verb
Verb ES Adjective
Verb ES Adverb

Table 6.9: Some Initial Patterns (N: Noun, V: Verb, PR: Pronoun, JJ: Adjective, RB: Adverb, ES: Emotion-Seed).

process is a set of grammatical sentences with meanings that match the original semantics. Our algorithm for sentence generation from this structure is: 1) extract a template for the input sentence and determine a lexical resource that matches each word in the template; 2) realize each piece for each argument in the template; and 3) combine the realizations from steps 1 and 2 in a grammatical format. For example, in the sentence, "They enjoy their lovely family Christmas holidays", our system can detect emotion terms in the sentence by constructing the template, and replace them with their alternative synonyms or paraphrases. Our results prove that our algorithm performs correctly and it is able to generate sentences to express emotion.

Handling the Paraphrases

To handle paraphrases in our model we consider different types of paraphrases, such as: **Synonymy:** Synonyms are the simplest paraphrases. We use them as different words that have the similar or the same meaning (i.e., (a) "They *enjoy* their *lovely* family Christmas holidays" (b) "They *like* their *beautiful* family Christmas holidays").

Replacement with Paraphrases: Another approach is to use realizations with different paraphrases, rather than just synonyms (i.e., "so happy to see" and "very glad

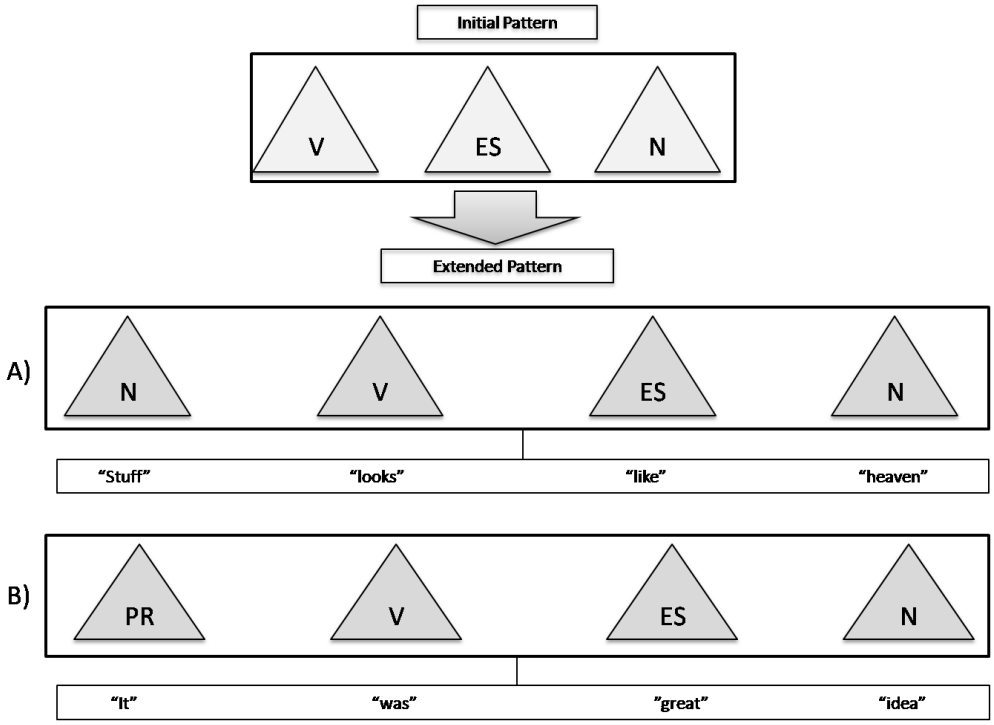

Figure 6.2: Constructing Extended Pattern by Initial Pattern with Examples (ES: Emotion Seed)

to visit").

6.2.7 Surface Realization

The realizer (the final NLG module) generates actual text based on decisions made by the document planner and the sentence planner (microplanner). A realizer generates individual sentences, typically from a 'deep syntactic' structure (Reiter, 2007). The realizer needs to ensure that the rules of English are followed. Some of these rules are:

- Punctuation rules: For example, the sentence: "Amy looks great, nice, and beautiful" must end with "." not with ",."
- Morphology: the plural of *box* is *boxes*, not *boxs*.
- Agreement: For example: "I am happy" instead of "I are happy".

Final Patterns	example
N V ES N	Amy is happy girl
PR V ES N	She is nice person
PR V ES RB	they lived happily together
PR V RB ES	I am quite upset
PR V ES JJ	I am feeling smart
PR V RB ES	I am pretty good

Table 6.10: Some final patterns (N: Noun, V: Verb, PR: Pronoun, JJ: Adjective, RB: Adverb).

- Reflexivity: For example: "Amy made herself happy" instead of "Amy made Amy happy".

We used SimpleNLG (Gatt and Reiter, 2009) and our Authoring Tool NLG System (Caropreso et al., 2009a) for sentence realization and to sentence generation. Using the pattern definitions from the previous sections, we designed a simple surface realization component for our model.

Our system is based on a set of patterns, the combination of sentences, and patterns (i.e., NP, VP, ES (Emotion-Seed), JJ (Adjective), and RB (Adverb)) with the ability to change individual pattern values. This is a flexibility aspect of our NLG system. We also believe that it could be adequate for simple text-generating applications. The system might need to generate a sentence that does not resemble any other in the corpus or the input text. However, this can be overcome by simply typing it into the text using natural language, or by specifying the new sentence from scratch, though in this case more input knowledge would be required.

We designed a simple surface realization component for our model, using the pattern definitions from the previous section. Our surface realization module can currently accept a template as input (to be taken as a sample structure with inherited default values for the output sentence) and, optionally, parameters representing the alternative semantics of its subject, verb and object constituents. Alternatively, it is possible to specify a sentence

from scratch without using an existing template as a basis, in a standard pattern format such as "Noun Verb Emotion-Seed Noun" or other pattern formats. We believe the latter option (pattern-based format) in the system helps to specify simpler sentence structures more conveniently, instead of having to look up an example, or find templates in the corpus.

In both the template-based and pattern-based approaches the user selects a target template/pattern, then the system provides a set of values to fill in the template/pattern variable fields. In the template-based approach, these input values overwrite the default values provided by the template; that is, those values that were inherited from the corpus data or other lexical sources.

6.2.8 Examples and Results

Providing Variables for Fields

Depending on the type of template or pattern, our system can support five types of variables: pronouns, nouns, verbs, adjectives and adverbs. Here we briefly discuss these variables, and their resources for our system.

The variables for the pronoun fields can be any type of pronouns, based on the pattern or template requirement. Some of these pronouns are provided in Table 6.4.

The supported variable fields for nouns and noun phrases are: determiner type, gender, number, person, pre and post-modifiers, the noun-phrase head, proposition, or relative clause. In this work nouns can be chosen by the user, or from the nouns provided in Section 6.2.3 (examples of these are shown in Table 6.5).

For verbs or verb phrases, the variable fields are verb or verb phrase type, and can be finite, infinitive, mode, verb tense or adverbial modifiers. The gender and number for verbs are not specified directly, but they can be inherited from the subject by the sentence realizer to avoid a conflicting input specification. Also, we can use a sample of list of verbs, like those provided in Table 6.6.

For adverb variable fields, our system can accept different types of adverbs (in terms

of word), such as adverbs of manner (e.g., carefully), adverbs of time (e.g., today, next week), adverbs of frequency (e.g., usually, occasionally), adverbs of degree (e.g., a lot, so much), and adverbs of comment (e.g., fortunately). Those in the list of adverbs provided in Table 6.8 can also be used (Neviarouskaya et al., 2007). For adverb and adjective variables, we used the list that provided by Neviarouskaya et al. (2007).

In terms of adjective variable fields, our system can accept different types of adjectives (in terms of word) (Neviarouskaya et al., 2007), such as: personal titles (e.g., Mr., Ms, Dr., etc.), possessive adjectives (e.g., my, your, his, our, their, its), demonstrative adjectives (e.g., this, that, these, those), indefinite adjectives (e.g., no, any, many, few, several), and numbers. The list of adjectives provided in Table 6.7 can also be used.

Examples with the Template-based Approach

In the following, we describe how a template approach works in our system (see Table 6.11). The system may choose the requirement for the template to generate the desired output text, which is shown in (1). To produce (2), the user is able to adjust some values of the variables in the template fields. Also, more available linguistic knowledge allows us to have more changes in the original template structure, and generate (3). The advantage of this method is that it can accommodate canned text in the template. Another advantage of our work is that it can generate different emotion sentences for different emotion categories. For example, if we change the emotion-seed *happy* to *sad* or *angry*, the output can be *sadness* or *anger* emotion text. Thus, in our method we can easily shift the valence of text or sentence by exchanging the emotion-seed in the current template to generate new text with different emotion expression (i.e., "she is happy because she passed her exam." to "she is sad because she failed her exam.")

Depending on changes in the values requested by the system or user, various agreement rules may be needed to reestablish grammaticality and fluency in the output sentence, in case if it is not fluent or grammatically correct. In our work, we achieve this by using SimpleNLG, our Authoring Tools NLG System . For example, as illustrated in Table 6.12, if we select the sentence template as (4), and singular subject "Amy" changed

Input	**Expected output**
(1) template	[Amy]$_{subject}$
	[looks like]$_{verb}$
	[a happy]$_{emotions-seed}$
	[person]$_{object}$
(2) template ,	[Amy]$_{subject}$
emotion-seed=nice,	[is]$_{verb}$
verb=is,	[a nice]$_{emotion-seed}$
object=person	[person]$_{object}$
(3) template,	[Amy]$_{subject}$
emotion-seed=great,	[has]$_{verb}$
verb=has,	[great]$_{emotion-seed}$
object=personality	[personality]$_{object}$

Table 6.11: Examples of (semantic) input, "Amy looks like a happy person." and expected (surface realization text) output.

to represent a plural "They" as shown in (5), then the agreement verb is needed to adjust the verb and the plural is needed for "person" to be modified to the plural "people" as shown in (6). Finally, in this case the basic agreement between $< Subject, Verb >$ or $< Subject, Object >$ is established.

The user can generate simple or complex sentences, as needed. Here is one more example of a sentence that can be generated using the authoring tool: "I am *so glad* that I bought this elegant house, after being on the house buying market for more than six months." Here is another sentence that was automatically generated by the tool from the initial sentence, using the paraphrase pair "so glad::very pleased" from our knowledge-base of paraphrases: "I am *very pleased* that I bought this elegant house, after being on the house buying market for more than six months." The tool detected that one expression from the paraphrase pair was present in the initial sentence, and generated a new sentence using the other expression in the pair. More examples generated with the

(4) [Amy]$_{subject}$	[looks]$_{verb}$
	[happy]$_{emotion-seed}$
	[person]$_{object}$
(5) [They]$_{subject}$	[looks]$_{verb}$
	[happy]$_{emotion-seed}$
	[person]$_{object}$
(6) [They]$_{subject}$	[are]$_{verb}$
	[happy]$_{emotion-seed}$
	[people]$_{object}$

Table 6.12: An original template sentence (4) reused with a new subject (5) and agreement (6). The outputs are: "Amy looks like a happy person" and "They are happy people".

template-based approach are shown in Table 6.13.

Examples with the Pattern-based Approach

To generate sentences with our patterns, the user can 1) choose any pattern, 2) identify the desired emotion category (happiness, anger, fear, sadness, surprise or disgust) and 3) select the variables for pattern fields (in this case, the system can choose automatically as well). The system then 4) transforms the semantic representation that is determined by the user to guide sentence formation, and generates different types of emotion sentences.

We describe how a pattern-based example works in our system below (Figure 6.3). As shown, the user selects a desired pattern; here the selected pattern is "Noun Verb Emotion-Seed Noun", and the desired emotion is "happiness". Then the user can select "Amy" for the first Noun in the pattern, and one of the verbs "look", "be" or "have" as the verb for the variable field Verb in the pattern. As an Emotion-Seed, suppose the user selects "happy", "nice" and "great". For the last field of the pattern, we consider "person" and "personality". To generate the final output, the system will transform the combination of patterns to a sentence format which is "Subject+Verb+Emotion-

Original: I am so **incredibly angry** right now! Generated: I am so **unbelievably mad** right now!
Original: somebody just called me a teeny, and I am **extremely pissed off**! Generated: somebody just called me a teeny, and I am **so angry**!
Original: In the meantime I am **damn being sick** with a cold but I am still alive. Generated: In the meantime I am **getting sick** with a cold but I am still alive.
Original: I am getting **really anxious** and **scared** and I am **afraid** of the pain and needles. Generated: I am getting **so worried** and **feared** and I am **fearful** of the pain and needles.
Original: The children went home together, and were **heartily delighted**. Generated: The children went home together, and were **sincerely happy**.
Original: I **feel pushed aside**, yet my heart resides with you. Generated: I **feel bad**, yet my heart resides with you.
Original: It's **truly amazing** the people we can find in this world. Generated: It's **really wonderful** the people we can find in this world.

Table 6.13: Some examples generated with the template-based approach by paraphrasing emotion expressions. The original sentenced are real sentences from blogs.

Seed+Object".

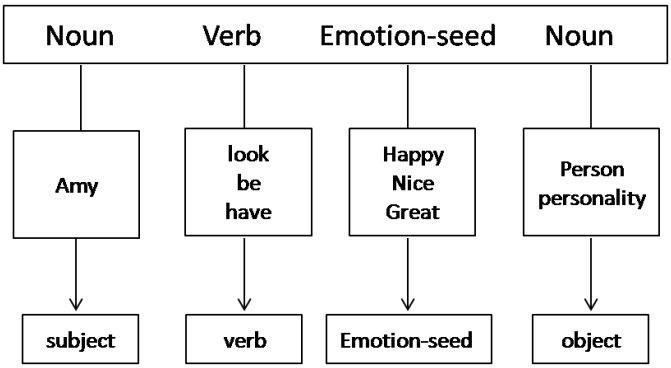

Figure 6.3: An example for generating sentence with a pattern for the emotion *happiness*.

As noted, some fields in the pattern need to be in agreement. For example, there must be agreement between $< Subject, Verb >$ and sometimes $< Subject, Object >$. Also, the pattern must have proper emotion seeds to be suitable for the sentence structure, and for fluency in the final output sentence. These constrains are again enforced by the sentence planner, agreement rules and the surface realization module.

With this pattern, the system can generate different types of emotion sentences. For example, we can use our Authoring Tool to generate different types of sentences with various subjects (singular, plural), verbs (with different tenses: present, past, future) and objects. Also, by changing the emotion expression to a different emotion category (e.g., *anger*), the system is able to generate various types of emotion sentences for the *anger* emotion with the same pattern. The final surface realization for this pattern is presented in Figure 6.14. We note here that, with one pattern, the system can generate various emotion sentences for different emotion classes, but the type of emotion sentence to be generated is the user's choice.

As we have shown, our system can generate various types of emotion sentences for each emotion class. Therefore, it is able to generate many texts for each pattern (the patterns are presented and described in Section 6.2.4).

However, as any research, our work has some limitations, as well. For example, for the results in Table 6.14, some sentences might not be perfect English sentences.

Amy looks like a happy person.
Amy is a nice person.
Amy has nice personality.
Stuff looks like heaven.
It was great idea.
They had a good time.

Table 6.14: An example of surface realization with the pattern: "Noun Verb Emotion-Seed Noun" for the emotion happiness.

For example, "Amy looks happy girl" must be "Amy looks a happy girl", or, for the sentence "She is nice person", is better to have "She is a nice person". As we can see, these sentences will be more fluent, if the determiner "a" is added. We added an extra module in our system, so that the user is able to add the determiners and other function words in the generated sentence.

6.2.9 Interface for the Pattern-based Approach

To utilitize our pattern-based model, we develop a graphical interface in Java using the Swing library. A screenshot with an example of this interface is shown in Figure 6.4. When using this interface, the user chooses a desired emotion from the emotion categories panel. Next, the user indicates a pattern from the pattern section. We allow function words such as "a, an, the, of, like", beside each pattern variable, that the user can choose, if it is needed.

This interface allows the user to create the favorite pattern as needed. In the screenshot shown in Figure 6.4, the example pattern is "Pronoun Verb Emotion-Seed Noun", and the user can choose a function word ("the") before the emotion seed, for obtaining a correct sentence. One example of generated output sentence is; "He hates the ugly dog". Based on this output, the system is able to generate longer sentence by adding more modifiers, to obtain the sentence: "He hates the ugly big red dogs." More examples generated with the pattern-based approach using the GUI are shown in Table 6.15.

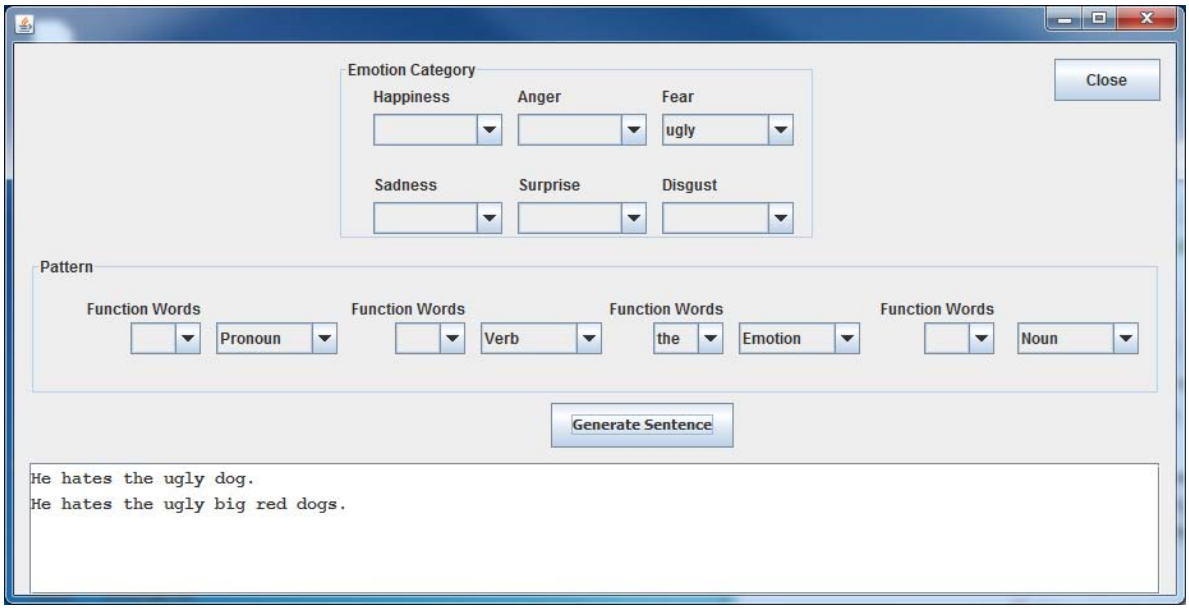

Figure 6.4: Graphical interface for the pattern-based approach.

6.3 Summary

This chapter presented a method of sentence generation to express emotions. In this chapter we introduced our pattern-based model for generating emotion sentences. The model is divided into seven sections: Data Set, Parsing and Tokenization, Pattern Extraction, Pattern Analysis, Sentence Planner, Surface Realization, and Examples and Results.

Our model started with initial patterns, then constructed extended patterns. From the extended patterns we chose final patterns that are suitable for generating emotion sentences.

In addition, we introduced Surface Realization into our model. The Realizer, which is the last module of our NLG model, generates actual sentences according to the Sentence Planner module. Our realizer can generate sentences from our pattern-based or template-based approaches. Our template based approach can be employed by users to generate sentences that can be as complex as needed, since most of the sentences if provided by a user. Variable parts can be employed as needed, and paraphrases of emotion terms can be generated automatically.

Patterns	Examples
N V ES N	I saw a mad man.
	I saw a mad man on my way.
PR V ES N	The flowers have a good smell.
	The flowers have a good smell and beautiful bloom.
PR V ES RB	She is afraid of dogs.
	She is afraid of dogs at night.
N V RB ES	Students are very excited.
	Students are very excited to see their marks.
PR V ES JJ	She has a good feeling.
	She has a good feeling about her exam.

Table 6.15: Some examples generated with the pattern-based approach by using the interface.

We presented some examples and results for our models. We showed that the models can generate various types of emotion sentences, either from semantic representation of input, or by choosing the pattern and the desired emotion class. The results indicate that our system generates simple English emotion sentences, and some that require minor modifications, in the sense that the user has to provide functions words. Since our work is a starting point for this line of research, our generation system is not fully automatic; the user needs to invest some effort in order to generate the desired sentences.

For future work, we plan to extend our model to cover more patterns for English sentences, and to include more emotion categories. We can also extend the system to use aggregation in order to generate more complex emotion sentences.

Chapter 7

Conclusion and Future Work

The purpose of the research detailed in this thesis was to investigate a computational approach to the recognition and generation of emotion in text. In order to accomplish this, we implemented an emotion and mood analysis and generation capability. The analysis was based on hierarchical emotion and mood classification, using machine learning techniques in different levels of the hierarchy. In addition to text classification, we also introduced and implemented a state-of-the-art method to extract paraphrases for emotion terms. This method used a bootstrapping algorithm, as well as textual and syntactic similarities, in order to extract paraphrases. We built rich lexical features and paraphrases that can be used in natural language generation. With regard to generation, we took part in the design, development and implementation of an Authoring Tool for Natural Language Generation that can generate narrative content text for any template based input. Finally, we introduce and developed a pattern-based model that is capable of generating emotion text for the six basic Ekman emotions. In the following, we present the outcomes and conclusions of this work.

In Chapter 3, we presented experiments for a hierarchy-based mood classification of blog texts. The hierarchical approached classified the data using the SVM algorithm. Our corpus was collected from Livejournal, an online service that allows users to post their personal thoughts.

In Chapter 4, we introduce a technique for corpus-based extraction of paraphrases

for emotion terms. We designed a method that uses a bootstrapping algorithm based on contextual and lexical features that can successfully extract paraphrases using a non-parallel corpus. We showed that the algorithm, which is based on the surrounding contextual features of paraphrases, achieves good performance on our data set. The extracted paraphrases are a very useful resource, which we used in our emotion generation system.

In Chapter 5, we designed and implemented an Authoring Tool for an NLG system. My contributions were in design, part of the implementation (using SimpleNLG), testing, and evaluating the Authoring Tool. This tool was applied to interactive simulation games that are used for training (Caropreso et al., 2009c,a). Simulation games usually require a huge amount of coherent narrative content, and one of the best solutions for this problem is to use NLG techniques. To do this, we implemented an NLG environment for generating and modifying NLG templates that requires no programming skills, and is able to function with minimal linguistic knowledge. We used this NLG Authoring Tool in our model to generate text to express emotions.

We examined our pattern-based model for generation, and showed how our model starts with initial patterns, then constructs extended patterns from which we choose "final" patterns that are suitable for generating emotion sentences. We also introduced a sentence planning module, which provides rules and constraints for our model.

Also, we introduce a surface realization module for our model. The realizer, which is the last module of our NLG model, generates an actual sentence according to the sentence planner (microplanner) module. Our realizer is able to generate sentences from our pattern-based or template-based approaches.

7.1 Summary of Contributions of this Thesis

A summary of the contributions of this thesis that were presented in Section 1.3 follows. The research for the thesis produced several contributions to emotion analysis and generation. Noteworthy is the fact that our methods are domain-independent and robust.

The contributions are listed below:

- We proposed a hierarchical classification method for mood / emotion in texts, that uses semantic orientation features.

- We proposed a new method for classifying text by mood and emotion, based on a hierarchical approach. We described experiments involving hierarchy-based mood classification of a blog corpus (Chapter 4), and we introduce a hierarchical approach to classifying data using the SVM algorithm. The data set was collected from Livejournal, an online service that allows users to post their personal thoughts. Our results indicated that the hierarchical approach provides substantial performance improvement over flat classification. Classifying mood in blog text is difficult, due to the diversity of the users. However, our approach showed that if we classify blogs using the mood hierarchy we can achieve very good results. We also showed that using sentiment orientation features on top of BoW features improves the classification performance.

Based on the hierarchical classification, we produced a domain of valuable features that can be used in emotion analysis and generation in natural language processing. As shown in many results in Chapter 3, these features are capable because our approach achieved high accuracy and performed well in classification.

Designed an algorithm to extract paraphrases for emotions

We based our method for paraphrase extraction on the assumption that phrases which appear in similar contexts are paraphrases. For our purposes, contexts are groups of the words that surround a phrase. The bootstrapping algorithm learns which contexts are good predictors of paraphrases, by analyzing the contexts surrounding identical words in aligned contexts. These contexts are used to extract new paraphrases, which are then applied to learn more contexts. Our algorithm produces phrasal and single word lexical paraphrases, as well as syntactic paraphrases. The results show that the algorithm extracts paraphrases with high accuracy.

Automatic generation of narrative content for digital games

We developed an NLG system that is used for training (Caropreso et al., 2009c,a). This type of system usually requires a huge amount of coherent narrative content, and using NLG techniques is an efficient and effective solution to the narrative content creation problem. However, the use of NLG systems requires sophisticated linguistic knowledge, and sometimes programming knowledge as well. For this reason, NLG systems are typically not accessible to the game designers who create the narrative content of the games. We designed and implemented a visual environment for creating and modifying NLG templates that requires no programming knowledge, and very little linguistic knowledge, to operate. It allows the specification of templates with any number of variables and dependencies between them, and it automatically generates all the sentences that follow the created template. Our NLG system used SimpleNLG (Gatt and Reiter, 2009) to provide the linguistic background knowledge. The system performance was tested in the context of an interactive simulation game. We identified the need for an NLG Template Authoring Environment that allows game content designers with no linguistic or programming background to experiment with, and eventually design, language templates.

Pattern-based model to generate text with emotion

We developed a pattern-based model for generating emotion sentences. We analyzed the model by extracting syntactical patterns from data sets, then extending the patterns to be good candidates for sentence generation. This was done by the tokenization and extraction of patterns, and by analyzing the patterns. We then developed a Sentence Planning module, which provides rules and constraints for our model.

In addition, we developed surface realization to display output text. The realizer generates sentences according to the sentence planner (microplanner) module, and is also used to generate sentences from our pattern-based model.

7.2 Directions for Future Research

Emotion analysis and generation is a relatively new field, and it has attracted a lot of interest in recent years. As with most doctoral studies, the research presented here has raised more questions than it has answered. In this section, we outline some avenues for further work that can build on the research conducted for this thesis.

We discuss short and long-term future directions in three categories: directions for emotion analysis and classification, future plans for paraphrase extraction, and plans for future work on our generation model.

Emotion Analysis and Classification

- For the classification of text by emotions and moods, we plan to experiment with more feature sets, types of features, and feature selection methods.
- We also plan to test our hierarchical algorithm on different corpora and data sets, which will determine the limitations of the approach.
- In the longer term, we would like to apply the hierarchical approach in semi-supervised learning models. We want to learn from the top-down and bottom-up of one classification level to the next, which will tell us how a particular level of the hierarchy can influence a higher or lower level. This could be used for both local and global hierarchical approaches.

Paraphrase Extraction

- For future work in paraphrase extraction, we could extend the techniques to extract paraphrases from more corpora, and for more types of emotions.
- In terms of evaluation, we could use the extracted paraphrases as features in machine learning classifiers that sort candidate sentences into classes of emotions. If the results of the classification are good, the extracted paraphrases are of high

quality. To explore more, we can use a supervised learning method to identify the performance of our paraphrase extraction algorithm.

Generation

For future work in authoring tools and text generation to express emotions, we plan to extend our model and system in the following directions:

- Develop our Authoring Tool NLG System to generate more complex sentences, rather than only simple and compound sentences.
- Cover various patterns for English sentences, in addition to those already present in our system.
- Extend our pattern-based model to cover more emotion categories than the six basic emotions (Ekman, 1992).
- Generate more complex sentences with the pattern-based model using aggregation.
- Generate more texts that change from a current emotion into a different emotion. For example, an *angry* sentence becoming a *happy* sentence. Although our pattern-based model is capable of shifting valence, we would like to explore other methods based on more comprehensive natural language generation techniques.

Bibliography

Alm, C., Roth, D., and Sproat, R. (2005). Emotions from text: machine learning for text-based emotion prediction. In *Human Language Technology Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 579–586.

Aman, S. and Szpakowicz, S. (2007). Identifying expressions of emotion in text. In *Proceedings of Text Speech and Dialog (TSD)*, pages 196–205.

Andreevskaia, A. and Bergler, S. (2008). When specialists and generalists work together: overcoming domain dependence in sentiment tagging. In *Proceedings of ACL-2008: HLT*, pages 290–298.

Banea, C., Mihalcea, R., and Wiebe, J. (2008a). A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In (ELRA), E. L. R. A., editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Banea, C., Mihalcea, R., Wiebe, J., and Hassan, S. (2008b). Multilingual subjectivity analysis using machine translation. In *Proceedings of EMNLP-2008*.

Barzilay, R. and Lapata, M. (2008). Modeling local coherence: An entity-based approach. In *Computational Linguistics*.

Barzilay, R. and Lee, L. (2004). Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceeding of NAACL-HLT*.

Barzilay, R. and McKeown, K. (2001). Extracting paraphrases from a parallel corpus. In *Proceeding of ACL/EACL, 2001, Toulouse*.

Bateman, J. A. (1997). Enabling technology for multilingual natural language generation: the kpml development environment. *Journal of Natural Language Engineering*, 3(1):15–55.

Beineke, P., Hastie, T., and Vaithyanathan, S. (2004). Improving review classification via human-provided information. In *Proceedings of the ACL*, pages 263–270.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *Proceedings of ACL-2007*.

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., and Struyf, J. (2002). Hierarchical multi-classification. In *Proceedings of the first SIGKDD workshop on multirelational data mining (MRDM-2002), Edmonton, Alberta, Canada*, page 2135. Houghton Mifflin, Boston.

Bostad, T. (2003). *Sentence Based Automatic Sentiment Classification*. PhD thesis, University of Cambridge, Computer Speech Text and Internet Technologies (CSTIT), Computer Laboratory.

Bradley, M. and Lang, P. (1999). Affective Norms for English Words (ANEW). *University of Florida*.

Broad, C. (1995). Emotion and sentiment. *Journal of Aesthetics and Art Criticism*, 13:203–214.

Buchanan, B., Moore, J., Forsythe, D., Carenini, G., Banks, G., and Ohlsson, S. (1995). An intelligent interactive system for delivering individualized information to patients. *Artificial Intelligence in Medicine*, (7):117–154.

Busemann, S. (2005). *Ten Years After: An Update on TG/2*. Proceedings of the 10th ENLG, pp:32–39, Aberdeen.

Caldwell, D. and Korelsky, T. (1995). Bilingual generation of job descriptions from quasi-conceptual forms. In *Proceedings of the Fourth Conference on Applied Natural-Language Processing. ACL*, pages 1–6.

Campbell, R. S. and Pennebaker, J. W. (2003). The secret life of pronouns: Flexibility in writing style and physical health. In *Psychological Science, 14, 60-65.*

Cardie, C., Wiebe, J., Wilson, T., and Litman, D. (2003). Combining low-level and summary representations of opinions for multi-perspective question answering. In *In AAAI Spring Symposium on New Directions in Question Answering*, pages 20–27.

Caropreso, M. F., Inkpen, D., Khan, S., and Keshtkar, F. (2009a). Automatic generation of narrative content for digital games. In *IEEE NLP-KE 2009 (International Conference in Natural Language Processing and Knowledge Engineering), Dalian , China.*

Caropreso, M. F., Inkpen, D., Khan, S., and Keshtkar, F. (2009b). Novice-friendly natural language generation template authoring environment. In *22nd Canadian Artificial Intelligence Conference, Kelowna, BC, Canada.*

Caropreso, M. F., Inkpen, D., Khan, S., and Keshtkar, F. (2009c). Visual development process for automatic generation of digital games narrative content. In *47th ACL-IJCNLP Workshop, Singapore. WS4: Language Generation and Summarization.*

Cawsey, A., Binsted, K., and Jones, R. (1995). Personalized explanations for patient education. In *Proceedings of the 5th European Workshop on Natural Language Generation*, pages 59–74.

Chen, H., Branavan, S., Barzilay, R., and Karger, D. R. (2009). Latent topic models for document structure induction. In *Proceeding of NAACL-HLT.*

Chevelu, J., Lavergne, T., Lepage, Y., and Moudenc, T. (2009). Introduction of a new paraphrase generation tool based on monte-carlo sampling. In *Proceedings of ACL-IJCNLP 2009, Singapore*, pages 249–25.

Choi, Y., Cardie, C., Riloff, E., and Patwardhan, S. (2005). Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT-EMNLP-05, the Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing*.

Christensen, C. M. and Raynor, M. E. (2003). *The Innovator's Solution: Creating and Sustaining Successful Growth*. Harvard Business School Press.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37 – 46.

Collins, M. and Singer, Y. (1999). Unsupervised models for named entity classification. In *proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Dalianis, H. (1996). *Concise Natural Language Generation from Formal Specifications*. PhD thesis, of Computer and Systems Sciences,Royal Institute of Technology/Stockholm University, Stockholm University, Sweden.

Das, D. and Smith, N. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of ACL-IJCNLP 2009, Singapore*, pages 468–476.

Das, S. and Chen, M. (2001). Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*.

Dekel, O., Keshet, J., and Singer, Y. (2004). Large margin hierarchical classification. In *Proceedings of the 21st International Conference on Machine Learning*.

Devitt, A. and Ahmad, K. (2007). Sentiment polarity identification in financial news: a cohesion-based approach. In *Proceedings of ACL-2007*.

ĎMello, S., Craig, S., Gholson, B., Franklin, S., Picard, R., and Graesser, A. (2005). Integrating affect sensors in an intelligent tutoring system. In *Proceedings of the International Conference on Intelligent User Interfaces.*

Durant, K. T. and Smith., M. D. (2006). Mining sentiment classification from political web logs. In *Proceedings of KDD Workshop on Web Mining and Web Usage Analysis (WebKDD) and the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).*

Ekman, P. (1992). An argument for basic emotions. *Cognition and Emotion*, 6:169–200.

Elhadad, M. (1994). *Using Argumentation to Control Lexicon Choice.* PhD thesis, Columbia University, New York, USA.

Elhadad, M. and Robin, J. (1996). An overview of surge: a reusable comprehensive syntactic realisation component. In *Proceedings of the 8th International Workshop on Natural Language Generation*, pages 1–4.

Esuli, A. (2006). Sentiment classification bibliography. In *www.ira.uka.de/bibliography/Misc/Sentiment.html.*

Felbaum, C. (1998). *WordNet, an Electronic Lexical Database for English.* Cambridge: MIT Press.

Finn, A. and Kushmerick, N. (2006). Learning to classify documents according to genre. *Journal of the American Society for Information Science and Technology (JASIST)*, 7(5). Special issue on computational analysis of style.

Finn, A., Kushmerick, N., and Smyth, B. (2002). Genre classification and domain transfer for information filtering. In *Proceedings European Colloquium on Information Retrieval Research, Glasgow*, pages 353–362.

Francis, W. N. and Kucera, H. (1982). Frequency analysis of english usage. In *Lexicon and grammar.* Houghton Mifflin, Boston.

Freitas, A. and Carvalho, A. (2007). *Research and trends in data mining technologies and applications, Idea Group.* Chapter A: tutorial on hierarchical classification with applications in bioinformatics.

Frijda, N., Mesquita, B., Sonnemans, J., and Goozen, S. V. (1992). The duration of affective phenomena or emotions, sentiments and passions. *Review of studies on emotion, Chichester: Wiley*, 1.

Gagn, R. M. and Briggs, L. J. (1997). *Principles of instructional design.* Harcourt Brace-Jovanovich, Fort Worth, TX, 4th ed. edition.

Garrod, S. and Anderson, A. (1987). Saying what you mean in dialog: A study in conceptual and semantic co-ordination. *Cognition*, 27:181–218.

Gatt, A. and Reiter, E. (2009). SimpleNLG: A realisation engine for practical applications. In *ENLG.*

Geldof, S. and de Velde, W. V. (1997). An architecture for template based (hyper)text generation. In *Proceedings of the 6th European Workshop on Natural Language Generation,*, pages 28–37.

Ghazi, D., Inkpen, D., and Szpakowicz, S. (2010). Hierarchical approach to emotion recognition and classification in texts. In Farzindar, A. and Keselj, V., editors, *Advances in Artificial Intelligence*, volume 6085 of *Lecture Notes in Computer Science*, pages 40–50. Springer Berlin / Heidelberg.

Glickman, O. and Dagan, I. (2004). Acquiring lexical paraphrases from a single corpus. In *Recent Advances in Natural Language Processing III, Nicolov, Nicolas, Kalina Bontcheva, Galia Angelova and Ruslan Mitkov (eds.) (selected papers from RANLP-2003)*, pages 81–90.

Goldberg, E., Driedgar, N., and Kittredge, R. (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9:45–53.

Green, D., Goldman, S., and Salovey, P. (1993). Measurement error masks bipolarity in affect ratings. *Journal of Personality and Social Psychology*, 64:1029–1041.

Grefenstette, G., Qu, Y., Evans, D. A., and Shanahan, J. G. (2004). Validating the coverage of lexical resources for affect analysis and automatically classifying new words along semantic axes. In *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text.*

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software. An Update; SIGKDD Explorations, Volume 11, Issue 1.

Halliday, M. (1985). *An Introduction to Functional Grammar.* London. Edward Arnold.

Hastie, T. and Tibshirani, R. (1998). Classification by pairwise coupling. *The Annals of Statistics*, 26:451 – 471.

Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of 35th Association for Computational Linguistics (ACL) and 8th European Association for Computational Linguistics (EACL).*

Hatzivassiloglou, V. and Wiebe, J. (2000). Effects of adjective orientation and gradability on sentence subjectivity. In *COLING.*

Hearst, M. (1992). *Direction-based text interpretation as an information access refinement.* Lawrence Erlbaum Associates.

Hiroshi, K., Tetsuya, N., and Hideo, W. (2004). Deeper sentiment analysis using machine translation technology. In *Proceedings of COLING-04.*

Holzman, L. and Pottenger, W. (2003). Classification of emotions in internet chat: An application of machine learning using speech phonemes. *Technical Report LU-CSE-03-002, Lehigh University.*

Hu, M. and Liu, B. (2004). Mining opinion features in customer reviews. In *Proceedings of AAAI*, pages 755–760.

Huettner, A. and Subasic, P. (2000). Fuzzy typing for document management. In *ACL 2000 Companion Volume, Tutorial Abstracts and Demonstration Notes*, pages 26–27.

Hurst, M. and Nigam, K. (2004). Retrieving topical sentiments from online document collections. In *Document Recognition and Retrieval XI*, pages 27–34.

Inkpen, D., Keshtkar, F., and Ghazi, D. (2009). *Analysis and Generation of Emotion in Texts*, pages 3–13. Presa Universitara Clujeana.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B., and Polguere, A. (1992). Generation of extended bilingual statistical reports. In *Proceedings of the Fifteenth International Conference on Computational Linguistics, COLING-92*, volume 3, pages 1019–1023.

Iordanskaja, L., Kittredge, R., and Polguere, A. (1991). *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic.

Izard, C. E. (1971). *The Face of Emotion*. Appleton-Century-Crofts, New York, NY.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *ECML, UK*, pages 137–142.

Jung, Y., Park, H., and Myaeng, S. H. (2006). A hybrid mood classification approach for blog text. *LNCS*, pages 137–142.

Kaliouby, E. (2005). *Mind-Reading Machines: Automated Inference of Complex Mental States*. PhD thesis, University of Cambridge, Computer Laboratory.

Karlgren, J. and Cutting, D. (1994). Recognizing text genres with simple metrics using discriminant analysis. In *Proceedings of COLING*, pages 1071–1075.

Keerthi, S., Shevade, S., Bhattacharyya, C., and Murthy, K. (2001). Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637 – 649.

Kennedy, A. and Inkpen, D. (2006). Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, pages 110–125.

Keshtkar, F. and Inkpen, D. (2009). Using sentiment orientation features for mood classification in blogs. In *Natural Language Processing and Knowledge Engineering, 2009. NLP-KE 2009. International Conference on*, pages 1–6.

Keshtkar, F. and Inkpen, D. (2010a). A bootstrapping method for extracting paraphrases of emotion expressions from texts. *Computational Intelligence Journal Special Issue on "Computational Approaches to Analysis of Emotion in Text", (Submitted, Aug. 2010).*

Keshtkar, F. and Inkpen, D. (2010b). A corpus-based method for extracting paraphrases of emotion terms. In *11th NAACL-HLT-2010(North American Chapter of the Association for Computational Linguistics-Human Language Technologies),WS2: Computational Approaches to Analysis and Generation of Emotion in Text*, pages 35–44, 2010, Los Angeles, CA.

Keshtkar, F. and Inkpen, D. (2011a). A computational model to drive sentence generation from paraphrases to express emotion. In *Corpus Linguistics 2011: Discourse and Corpus Linguistics conference*, Birmingham, England.

Keshtkar, F. and Inkpen, D. (2011b). An hierarchical approach to mood classification in blogs. *Journal of Natural Language Engineering (JNLE)*, pages 1–21.

Keshtkar, F. and Inkpen, D. (2011c). A Pattern-based Model for Generating Text to Express Emotion. In *Affective Computing and Intelligent Interaction (ACII 2011)*, Memphis, TN.

Kessler, B., Nunberg, G., and Schutze, H. (1997). Automatic detection of text genre. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 32–38.

Khan, S. (2007). *Negation and Antonymy in Sentiment Classification.* PhD thesis, Trinity Hall College, University of Cambridge, Cambridge, UK.

Kim, S.-M. and Hovy, E. (2004). Determining the sentiment of opinions. *Proceedings of the 20th COLING.*

Kim, S.-M. and Hovy, E. (2005a). Automatic detection of opinion bearing words and sentences. In *Companion Volume to the Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05).*

Kim, S.-M. and Hovy, E. (2005b). Identifying opinion holders for question answering in opinion texts. In *Proceedings of AAAI-05 Workshop on Question Answering in Restricted Domains.*

Kiritchenko, S., Matwin, S., Nock, R., and Famili, F. (2004). Functional annotation of genes using hierarchical text categorization. In *Proceedings of the 5th Annual Meeting of the NRCs Genomics and Health Research Initiatives Program (GHI) Montral*, page 192201.

Kiritchenko, S., Matwin, S., Nock, R., and Famili, F. (2006). Learning in the presence of class hierarchies. In *Proceedings of Learning-Snowbird 2006, Snowbird UT.*

Klein, D. and Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems, Cambridge, MA: MIT Press*, 15:3–10.

Kobayashi, N., Inui, K., Matsumoto, Y., Tateishi, K., and Fukushima, T. (2004). Collecting evaluative expressions for opinion extraction. In *Proceedings of IJCNLP-04, the 1st International Joint Conference on Natural Language Processing.*

Koppel, M., Argamona, S., and Shimoni, A. R. (2002a). Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412.

Koppel, M., Argamona, S., and Shimoni, A. R. (2002b). Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17(4):401–412.

Koppel, M. and Schler, J. (2004). Authorship verification as a one-class classification problem. *ICML, Twenty-first international conference on Machine learning*.

Koppel, M. and Shtrimberg, I. (2004). Good news or bad news? let the market decide. In *IProceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text*.

Kushal, D., Lawrence, S., and Pennock, D. M. (2003). Opinion extraction and semantic classification of product reviews. In *Proceedings of World Wide Web Conference (WWW)*.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *COLING-ACL*.

Langkilde-Geary, I. and Knight, K. (2002). Halogen statistical sentence generator. In *Proceedings of the ACL-02 Demonstrations Session*, pages 102–103, Philadelphia. Association for Computational Linguistics.

Lee, Y.-B. and Myaeng, S. H. (2002). Text genre classification with genre-revealing and subject-revealing features. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*.

Lepore, S. J. and Smyth, J. M. (2002). The writing cure: How expressive writing promotes health and emotional well-being. In *Washington, DC: American Psychological Association*.

Li, J. and Sun, M. (2007). Experimental study on sentiment classification of chinese review using machine learning techniques. In *Proceedings of IEEE-NLPKE-2007*.

Lin, W.-H., Wilson, T., Wiebe, J., and Hauptmann, A. (2006). Which side are you on? identifying perspectives at the document and sentence levels. In *CoNLL*.

Liu, H., Hockenberry, M., and Selker, T. (2003a). A model of textual affect sensing using real-world knowledge. *IUI, Proceedings of the 8th international conference on Intelligent user interfaces*, pages 125–132.

Liu, H., Lieberman, H., and Selker, T. (2003b). A model of textual affect sensing using real-world knowledge. In *Proceedings of Intelligent User Interfaces (IUI)*, pages 125–132.

Liu, H. and Singh, P. (2004). Conceptnet, a practical commonsense reasoning tool-kit. *BT Technology Journal*, pages 211–226.

Madnani, N. and Dorr, B. J. (2010). Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36.

Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993). Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Matthiessen, C. (1991). Lexico(grammatical) choice in text generation. *C. Paris, W. Swartout, and W. Mann. Natural Language Generation in Artificial Intelligence and Computational Linguistics. Kluwer Academic Press*, pages 249–292.

McDonald, R., Hannan, K., Neylon, T., Wells, M., and Reynar, J. (2007). Structured models for fine-to-coarse sentiment analysis. In *Proceedings of ACL-2007*.

McKeown, K., Kukich, K., and Shaw, J. (1995). Practical issues in automatic document generation. In *Proceedings of the Fourth Conference on Applied Natural-Language Processing*, pages 7–14.

Melamed, I. D. (2001). *Empirical Methods for Exploiting Parallel Texts*. MIT Press.

Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In *Natural Language Learning (CoNLL 2004)*, Boston.

Mihalcea, R. and Liu, H. (2006). A corpus-based approach to finding happiness. In *AAAI Spring Symposium on Computational Approaches to Weblogs, Stanford, CA*.

Mihalcea, R. and Strapparava, C. (2006). Learning to laugh (automatically): Computational models for humor recognition. *Journal of Computational Intelligence*.

Mishne, G. (2005). Experiments with mood classification in blog posts. *ACM SIGIR*.

Mohammad, S. M. and Turney, P. D. (2010). Emotions evoked by commonwords and phrases: Using mechanical turk to create an emotion lexicon. In *The 11th NAACL HLT 2010, Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 36–44.

Moore, J. D., Porayska-Pomsta, K., Varges, S., and Zinn, C. (2004). Generating tutorial feedback with affec. In *Proc. of the 17th International Florida Artificial Intelligence Research Society Conference. AAAI Press*, pages 923–928.

Mosteller, F. and L.Wallace, D. (1984). Applied bayesian and classical inference: The case of the federalist papers. *Springer-Verlag*.

Mullen, T. and Collier, N. (2004). Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP-04*.

Neviarouskaya, A., Prendinger, H., and Ishizuka, M. (2007). Textual affect sensing for sociable and expressive online communication. In *ACII*, pages 218–229.

Nicolov, N., Salvetti, F., Liberman, M., and Martin, J. H. (2006). The aaai spring symposium on computational approaches to weblogs. aaai press. In *AAAI Press*.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

Pang, B. and Lee, L. (2008). Opinion minding and sentiment analysis. *Foundation and Trend in Information Retrieval*, 2.

Paris, C., Linden, K., Fischer, M., Hartley, A., Pemberton, L., Power, R., and Scott, D. (1995). A support tool for writing multilingual instructions. In *Proceedings of Fourteenth International Joint Conference on Artificial Intelligence*, pages 1398–1404.

Passonneau, R. (2006). Measuring agreement on set-valued items (masi) for semantic and pragmatic annotation. In *Proceeding of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*.

Pennebaker, J. W., Chung, C. K., Ireland, M., Gonzales, A., and Booth, R. J. (2007). Linguistic inquiry and word count (liwc). In *Mahwah: Lawrence Erlbaum Associates*.

Pereira, F., Tishby, N., and Lee., L. (1993). Distributional clustering of english words. In *In proceedings of the 30th Annual Meeting of the ACL*, pages 183–190. ACL.

Picard, R. (1997). *Affective computing*. MIT Press, Cambridge, MA, USA.

Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods, Support Vector Learning, B. Schoelkopf, C. Burges, and A. Smola, eds., MIT Press*.

Plough, N. (2000). *The Oxford English Dictionary*. Oxford University Press, 2nd edition. http://dictionary.oed.com.

Polanyi, L. and Zaenen, A. (2004). Contextual valence shifters. In *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text*.

Popescu, A.-M. and Etzioni, O. (2005). Extracting product features and opinions from reviews. In *Proceedings of HLT-EMNLP*.

Qu, Y., Shanahan, J., and Wiebe, J. (2004). Exploring attitude and affect in text: Theories and applications. In *AAAI technical report SS-04-07*. AAAI Spring Symposium, AAAI Press.

Rambow, B. L. O. (1997). A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, DC*, pages 265–268.

Razavi, A. H., Inkpen, D., Matwin, S., and Uritsky, S. (2010). Offensive language detection using multi-level classification. In *Proceedings of the 23rd Canadian Conference on Artificial Intelligence (AI 2010)*, pages 16–27, Ottawa, ON, Canada, May 2010.

Read, J. (2004). Recognizing affect in text using pointwise mutual information. Master's thesis, University of Sussex.

Read, J. (2005). Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of ACL-2005*.

Reiter, E. (2007). An architecture for data-to-text systems. In *Proceedings of ENLG-2007, pp. 97-104*, pages 97–104.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems (Studies in Natural Language Processing)*. Cambridge University.

Reiter, E. and Sripada, S. (2002). Squibs and discussions human variation and lexical choice. In *ACL*.

Riloff, E. (1996). Automatic generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049.

Riloff, E. and Jones, R. (1999). Learning dictionaries for information extraction by multi-level boot-strapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, page 10441049. The AAAI Press/MIT Press.

Riloff, E. and Wiebe, J. (2003). Learning extraction patterns for subjective expressions. In *Natural Language Processing (EMNLP-2003)*, pages 105–112.

Rubin, V., Stanton, J., and Liddy, E. (2004). Discerning emotions in texts. *AAAI-EAAT*.

Sack, W. (1994). On the computation of point of view. In *Proceedings of AAAI, Student abstract*, page 1488.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing, Manchester, UK*.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):147.

Siegel, S. and Castellan, J. (1988). *Non Parametric Statistics for Behavioral Sciences*. McGraw-Hill.

Silla, C. N. and Freitas, A. A. (2010). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

Springer, S., Buta, P., and Wolf, T. (1991). Automatic letter composition for customer service. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference (CAIA-1991)*, pages 67–83.

Stamatatos, F. and Kokkinakis (2000). Text genre detection using common word frequencies. In *Proceedings of the 18th International Conference on computational Linguistics (ACL)*.

Stone, P., Dunphy, D., Smith, M., and Ogilvie, D. (1966). The general inquirer: A computer approach to content analysis. *MIT Press*.

Stoyanov, V., Cardie, C., Litman, D., and Wiebe, J. (2004). Evaluating an opinion annotation using a new multi-perspective question and answer corpus. In *James G. Shanahan, Janyce Wiebe, and Yan Qu, editors, Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications, Stanford, US*.

Strapparava, C. and Mihalcea, R. (2007). Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on the Semantic Evaluations (SemEval 2007), Prague, Czech Republic, June 2007*.

Strapparava, C. and Mihalcea, R. (2008). Learning to identify emotions in text. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560.

Strapparava, C. and Valitutti, A. (2004). Wordnet-affect: an affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004), Lisbon, May 2004*, pages 1083–1086.

Subasic, P. and Huettner, A. (2001). Affect analysis of text using fuzzy semantic typing. *IEEE Transactions on Fuzzy Systems*, 9(4):483–496.

Swartout, W. (1983). Xplain: a system for creating and explaining expert consulting systems. . *Artificial Intelligence*, 21:285–325.

Tellegen, A., Watson, D., and Clark, L. A. (1999). On the dimensional and hierarchical structure of affect. *Psychological Science. University of Minnesota, University of Iowa*, 10(4):297–303.

Tong, R. M. (2001). An operational system for detecting and tracking opinions in on-line discussion. In *SIGIR Workshop on Operational Text Classification*.

Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 252–259.

Tsou, B., Yuen, R., Kwong, O., La, T., and Wong, W. (2005). Polarity classification of celebrity coverage in the chinese press. In *Proceedings of International Conference on Intelligence Analysis*.

Turney, P. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In (ELRA), E. L. R. A., editor, *Proceedings of ACL-2002*, Marrakech, Morocco.

Turney, P. and Littman, M. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM (TOIS)*, 21(4).

Wan, X. (2008). Using bilingual knowledge and ensemble techniques for unsupervised chinese sentiment analysis. In *Proceedings of EMNLP-2008*.

Wang, K., Zhou, S., and Liew, S. C. (1999). Building hierarchical classifiers using class proximities. *Proceedings of VLDB*, pages 363–374.

Wang, X., Lo, D., Jiang, J., Zhang, L., and Mei, H. (2009). Extracting paraphrases of technical terms from noisy parallel software corpora. In *Proceedings of ACL-IJCNLP 2009, Singapore*, pages 197–200.

Watson, D. and Tellegen, A. (1985). Toward a consensual structure of mood. *Psychological Bulletin*, (98):219235.

Wiebe, J. and Wilson, T. (2002). Learning to disambiguate potentially subjective expressions. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 112–118.

Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation 39(2-3)*, pages 165–210.

Wiebe, J. M. (1994). Tracking point of view in narrative. *Computational Linguistics*, 20(2):233–287.

Wiebe, J. M. and Rapaport, W. J. (1988). A computational theory of perspective and reference in narrative. In *Tracking point of view in narrative. Computational Linguistics*, pages 131–138. Proceedings of the ACL.

Wiebe, J. M., Wilson, T., and Bell, M. (2001). Identifying collocations for recognizing opinions. In *Proceedings of the ACL/EACL Workshop on Collocation.*

Williams, S. and Reiter, E. (2006). Generating basic skills reports for low-skilled readers. *Natural Language Engineering*, 1(1):1–32.

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP).*

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Ye, Q., Shi, W., and Li, Y. (2006). Sentiment classification for movie reviews in chinese by improved semantic oriented approach. In *Proceedings of 39th Hawaii International Conference on System Sciences*.

Yi, J., Nasukawa, T., Bunescu, R., and Niblack, W. (2003). Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*.

Yi, J. and Niblack, W. (2005). Sentiment mining in webfountain. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*.

Yu, H. and Hatzivassiloglou, V. (2003). Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*.

Zhao, S., Lan, X., Liu, T., , and Li, S. (2009). Application-driven statistical paraphrase generation. In *Proceedings of ACL-IJCNLP 2009, Singapore*, pages 834–842.

Appendix A

Testing the Authoring System (Chapter 5)

A.1 House Purchase Negotiation Game Content Generation

Background:

A negotiation game is being created to train the player in being able to recognize and react to offers being made. It is designed to focus on the issues, and purchase offers only.

House Purchase Offers

- Price [260000, 270000, 280000, 290000, 300000]
- Closing Date [2 weeks, 4 weeks, 6 weeks, 8 weeks]
- Appliances [included, not-included]

Rejection Reasons:

- Price: Too-low, no-problem
- Closing: Too-early, no-problem, too-late
- Appliances: included, no-problem

Scenario 1: (create manual output)

Purchase of a house [Player is the buyer; Computer AI is the seller]
You have offered to purchase the house for 260000, with appliances included, with a proposed closing date of two weeks from now.
Congratulations, you are the new owner. The closing is in four weeks, the appliances are not included, and it costs you 300000.
Your offer was rejected. The seller said that the price was too-low and the appliances were included.

Scenario 2: (create templates using tool)

Sale of a house [Player is the seller; Computer AI is the buyer]
An offer was made to buy the house for 260000, with appliances included, with a proposed closing date of two weeks from now.
Congratulations, you have sold your house. The closing is in four weeks, the appliances are not included, and they payed you 300000.
You have rejected the offer because the price was too-low and the appliances were included.
Example of generated Game in the following structure:
<Sent price: 260000, closing: 2 weeks, appliances: included>
You have offered to purchase the house for 260000, with appliances included, with a proposed closing date of two weeks from now. $< /Sent >$
<Sent price: 300000, closing: four weeks, appliances: not included>
Congratulations, you are the new owner. The closing is in four weeks, the appliances are not included, and it costs you 300000. $< /Sent >$
<Sent price: too low, closing: no problem, appliances: included>
Your offer was rejected. The seller said that the price was too$_{low}$ and the appliances included.$< /Sent >$

A.2 Job Offer Negotiation Game Content Generation

Note: Outputs and Templates from scenario 1 and 2 may be reused as this also involves similar game actions available to players **Job Offers:**

- Salary [60K, 70K, 80K, 90K, 100K]
- Start Date [2 weeks, 4 weeks, 6 weeks, 8 weeks]
- RRSP Co-payment [included, not-included]

Rejection Reasons:

- Salary: Too-high, no-problem
- Start Date: Too-early, no-problem, too-late
- RRSP Co-payment: included, no-problem

Scenario 1: (create manual output)

Negotiate salary and benefits for new job [Player is the job seeker;
Computer AI is the employer
You have offered to join the firm for 100K, with RRSP Co-payment included, with a proposed start date of two-weeks from now.
Congratulations, you obtained the job. The start date is in four-weeks, RRSP Co-payment is not included, and you have a salary of 60K.
Your offer to join the firm was rejected. The employer said that the salary was too-high and that the RRSP Co-payment was included.

Scenario 2: (create templates using tool)

Negotiate salary and benefits for new job [Player is the employer;
Computer AI is the job seeker
You offered the candidate to join the firm for 100K, with RRSP Co-payments included, and a proposed starting date of two weeks from now.
Congratulations, the candidate accepted the job. The starting date is in four weeks,

RRSP Co-payment is not included, and the salary is 100K.
You rejected the candidate's offer to join the firm because the salary was too-high and the RRSP Co-payment was included.

A.3 Templates Generation System Evaluation Survey 1

Answered by content writers after training.
After receiving training, is the interface easy to use? 1 (very complicated) - 5 (very easy)
How long did it take you to learn to use the interface? 1 hour - 1 day - 1 week - 1 month
(we asked this question to know how easy or difficult is to learn the system interface)
How long did it take you to feel comfortable using the interface? 1 hour - 1 day - 1 week - 1 month
How clearly are options presented in the interface?
1 (not clear) - 5 (very clear)
Indicate the type of sentences you have created:

- short sentences / long sentences
- one subject / several subjects
- one verb / several verbs
- one variable / several variables
- with many/few dependencies / without dependencies
- facts / negations / questions
- progressive / passive / perfect
- form: normal / imperative / infinitive
- verb agreement: default / plural / variable

With respect to the generated sentences:
are them always correct? In which types (from above) they were not correct? Please give examples of errors in generation. Is there any other type of sentences you would like to create?
How useful do you consider the system?
1 (not really useful) - 5 (very useful)
Please explain why you do/don't consider the system useful.
What would make the System more useful?

A.4 Templates Generation System Evaluation Survey 2

Answered ("more enjoyable" or "less of a headache") by content writers after using it for generating the content for scenario 2 of both negotiation games.
Advantages and disadvantages of writing the sentences manually.
Advantages and disadvantages of using the System.
According to their answers; "more enjoyable" and "less of a headache" than manually writing all the sentences.
Advantages and disadvantages between:
a) creating all the different rejection templates
b) having all the combinations already in a semantic class and using it for generating only one rejection template.

A.5 Results of the Authoring Tool's Evaluation

	User 1		User 2		User 3	
Game 1 Manually	1 spelling error (repeated 4 times)	40	1 spelling error (repeated 40 times)	40	4 spelling errors (repeated 99 times) 2 missing sentences 4 repeated sentences	32
Game 1 System	1 un-IDed verb 1 missed filter	21	4 missed filters	31	1 spelling error 4 missed filters	29
Game 2 Manually	no errors found	25	1 spelling error (repeated 40 times)	17	3 spelling errors (repeated 55 times) 2 extra sentences	35
Game 2 System	1 spelling error	20	1 template missing	12	1 spelling error 1 missing template 2 extra templates 2 wrong semantic classes	15

Figure A.1: Errors Made and Time Invested (minutes) when Generating Content for the Negotiation Games.