SARE: A SENTIMENT ANALYSIS RESEARCH ENVIRONMENT

by
MUS'AB HABIB HUSAINI

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University
July 2013

SARE: A SENTIMENT ANALYSIS RESEARCH ENVIRONMENT

Approved by:

Assoc. Prof. Dr. Yücel Saygın .....
(Thesis Supervisor)

Assoc. Prof. Dr. Berrin Yanıkoğlu .....
(Thesis Co-Supervisor)

Asst. Prof. Dr. Hakan Erdoğan .....

Asst. Prof. Dr. Hüsnü Yenigün .....

Asst. Prof. Dr. Cemal Yılmaz .....

Date of Approval: July 18, 2013

SARE: A SENTIMENT ANALYSIS RESEARCH ENVIRONMENT

Mus'ab Habib Husaini

Computer Science and Engineering, MS Thesis, 2013

Thesis Supervisor: Yücel Saygın

Keywords: sentiment analysis, opinion mining, aspect lexicon extraction, set cover approximation, integrated research environment

Abstract

Sentiment analysis is an important learning problem with a broad scope of applications. The meteoric rise of online social media and the increasing significance of public opinion expressed therein have opened doors to many challenges as well as opportunities for this research. The challenges have been articulated in the literature through a growing list of sentiment analysis problems and tasks, while the opportunities are constantly being availed with the introduction of new algorithms and techniques for solving them. However, these approaches often remain out of the direct reach of other researchers, who have to either rely on benchmark datasets, which are not always available, or be inventive with their comparisons.

This thesis presents Sentiment Analysis Research Environment (SARE), an extendable and publicly-accessible system designed with the goal of integrating baseline and state-of-the-art approaches to solving sentiment analysis problems. Since covering the entire breadth of the field is beyond the scope of this work, the usefulness of this environment is demonstrated by integrating solutions for certain facets of the aspect-based sentiment analysis problem. Currently, the system provides a semi-automatic method to support building gold-standard lexica, an automatic baseline method for extracting aspect expressions, and a pre-existing baseline sentiment analysis engine. Users are assisted in creating gold-standard lexica by applying our proposed set cover approximation algorithm, which finds a significantly reduced set of documents needed to create a lexicon. We also suggest a baseline semi-supervised aspect expression extraction algorithm based on a Support Vector Machine (SVM) classifier to automatically extract aspect expressions.

SARE: BİR DUYGU ANALİZİ ARAŞTIRMA ORTAMI

Mus'ab Habib Husaini

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2013

Tez Danışmanı: Yücel Saygın

Anahtar Kelimeler: duygu analizi, düşünce madenciliği, görüş sözlüğü çıkarımı, set kaplama yaklaştırımı, entegre araştırma ortamı

Özet

Duygu analizi geniş kapsamlı uygulama alanı olan önemli bir öğrenme problemidir. Online sosyal medyanın hızlı yükselişi ve burada ifade edilen kamuoyunun artan önemi, pek çok zorluğun yanı sıra bu araştırma için fırsat kapılarını açmaktadır. Zorluklar gittikçe büyüyen duygu analizi problemlerinin ve görevlerinin yer aldığı bir listeye eklenerek literatürde ifade edilirken, fırsatlar bu zorlukları çözmek için önerilen yeni algoritmalar ve teknikler ile avantaja dönüştürülmektedir. Ancak bu yaklaşımlar çoğunlukla diğer araştırmacıların doğrudan erişimine uzak omaktadır. Bu araştırmacılar ya her zaman mevcut olmayan kıyaslama veri setlerine dayanmak zorunda kalmakta veya karşılaştırma yaparken yaratıcı olmak durumundadırlar.

Bu tezde genişletilebilir, temel ve modern yaklaşımları entegre ederek duygu analiz problemlerini çözmek için tasarlanmış ve kamuya açık bir sistem olan Duygu Analizi Araştırma Ortamı (SARE) sunulmaktadır. Araştırma alanını tüm genişliğiyle ele almak bu çalışmanın kapsamı dışında olduğu için, bu ortamın yararlılığı bir kısım görüş tabanlı duygu analizi problemlerinin çözümlerinin ortama entegrasyonuyla gösterilmektedir. Şu anda sistem, altın standardında bir sözlük oluşturulmasını sağlayan yarı otomatik bir yöntem, görüş ifadelerini otomatik çıkarmak için bir yöntem, ve önceden varolan temel bir duygu analiz motoru içermektedir. Kullanıcılara bizim önerdiğimiz set kaplama yaklaştırımı algoritması kullanılarak altın standardında bir sözlük oluşturmak için yardım edilmektedir. Önerilen bu algoritma, sözlüğü oluşturmak için gerekli olan belgeler setinin eleman sayısını ciddi miktarda düşürmektedir. Ayrıca, görüş ifadelerini ayıklamak için yarı denetimli ve Destekçi Vektör Makinası (SVM) sınıflandırıcı tabanlı otomatik bir algoritma önerilmiştir.

جَہاں بانی سے ہے دُشوار تر کارِ جہاں بینی
جِگر خوں ہو تو چشمِ دِل میں ہوتی ہے نظر پیدا

اِقبال

ACKNOWLEDGMENT

I am deeply indebted to my academic advisor, Assoc. Prof. Dr. Yücel Saygın, for his unwavering support, which has been essential to my academic and personal growth, and I cannot thank him enough for it. The kind advice and feedback from Assoc. Prof. Dr. Berrin Yanıkoğlu has shaped this project and given it the direction it has today. The enthusiasm I received from Dr. Dilek Tapucu from the very beginning has been an indispensable source of confidence and inspiration for me, for which I am very grateful.

Parts of this project were developed in the context of UBIPOL (Ubiquitous Participation Platform for Policy Making) project funded by European Commission, FP7, and I would especially like to acknowledge the work done by Ahmet Koçyiğit on the aspect-based sentiment analysis engine. My continued education at Sabancı University would not have been possible without the help of Dr. Brooke Luetgert of the Faculty of Arts and Social Sciences, whose research project I have been funded by for the last year. I would also like to mention the kindness and encouragement of my professors. I am especially grateful to Asst. Prof. Dr. Cemal Yılmaz, Asst. Prof. Dr. Hüsnü Yenigün, and Asst. Prof. Dr. Hakan Erdoğan for agreeing to be on the thesis committee.

In the end, it comes down to one's support system and I am blessed to have the strongest one. In particular, I would like to recognize the assistance given to me by my friends Salim Sarımurat, Iyad Hashlamon, and Gizem Gezici. They never turned me down when I needed favors and I cannot be more grateful to have met each one of them. I have also been fortunate enough to have the most caring and loving in-laws that anyone could ask for. Their constant concern and encouragement made things much easier than they otherwise seemed. It is impossible to describe the debt I owe to my parents and siblings. I have mostly been away from them, but being able to talk to them and laugh with them has kept me going. To my mother, especially, I will be eternally thankful for always believing in me, trusting me against all odds, and praying for my success. Finally, but most importantly, none of this would have been possible without Alia, who has supported me in every situation, whose comfort has kept me grounded, and whose happiness has created happiness for me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS

α an instance of an algorithm.

b a parameter that indicates the extent of automatic labeling; greater than or equals to 0.

$\mathcal{D}$ a corpus of documents.

$\hat{\mathcal{D}}$ an approximation of a corpus of documents.

D a document.

$\hat{D}$ an approximation of a document.

Δ the extent of data reduction.

$\hat{E}$ the set of all candidate aspect expressions.

$\hat{e}$ a single candidate aspect expression.

k an arbitrary number.

L labeled data.

λ a probability acceptability threshold in the range [0, 1].

$\hat{\tau}$ error tolerance.

U unlabeled data.

LIST OF ABBREVIATIONS

API Application Programming Interface.

CRF Conditional Random Fields.

CSS Cascading Style Sheets.

CSV Comma-Separated Values.

GPL General Public License.

HMM Hidden Markov Models.

HTML HyperText Markup Language.

JPA Java Persistence API.

JSON JavaScript Object Notation.

JVM Java Virtual Machine.

LDA Latent Dirichlet Allocation.

MVC Model-View-Controller.

NLP Natural Language Processing.

ORM Object-Relational Mapping.

pLSA Probabilistic Latent Semantic Analysis.

PoS Part-of-Speech.

REST REpresentational State Transfer.

SaaS Software as a Service.

SARE Sentiment Analysis Research Environment.

SVM Support Vector Machine.

XML Extensible Markup Language.

1

INTRODUCTION

What others think has always been a topic of deep curiosity for people, societies, governments, organizations, and commercial enterprises; and in the age of democracy and consumerism, finding an answer to it has become more important than ever before. Opinions about a particular public personality or commercial product are considered highly relevant and even essential sources of information for the respective stakeholders in determining future courses of action and overall strategies. While research on sentiment analysis and opinion mining within the field of computer science might have started almost a decade before the beginning of the new millennium, it was not until the rise of the social web and the subsequent explosion and mass availability of opinionated data that in-depth research in this area received greater impetus [39, 29, 28]. It is from this practically infinite amount of data that we draw not only great opportunities but also immense challenges for understanding opinions and representing them in a manner useful for consumption by the target audience.

In the context of computer science research, sentiment analysis (or more generally opinion mining) is "the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes" [29]. While to a human mind, the problem of understanding the opinion contained a single document may seem solvable if not quite trivial, attempting to find an automated solution to it opens up a range of other problems that defy any assumption of triviality. Even for humans, the problem no longer remains minor, and in fact becomes prohibitively expensive, when we consider the sheer volume of data that need to be processed and summarized. Thus, analyzing sentiment can mean any number of things in a given context and can be open to several limitations, not one of which can be easily eliminated without introducing yet another one in exchange.

We can try to solve what is known as the subjectivity analysis problem by attempting to separate subjective or opinionated text[1] from objective or factual text. This does not go a long way in determining the type of sentiment that the text expresses, but can at least provide us with some indication of whether the text contains any sentiment at all and also serves to eliminate noise created by purely objective and factual parts of the text. We can take it a step further and separate text into three categories, viz.: positive, neutral, and negative; where neutral denotes objective text and subjective text is classified as containing either positive or negative sentiment [28]. This is referred to as the sentiment classification problem. We can make the classification more granular and deal with the polarity estimation problem which seeks to express sentiment polarity on an arbitrary scale such as -1 to $+1$, 1 through 5, or A through F, etc.

At the same time, sentiment at the document level is not always uniform; a single document can simultaneously contain many seemingly opposing sentiments expressed in different sentences or even about different entities or aspects of the same entity within the same sentence [29]. Consequently, various strands of sentiment analysis research focus on solving the sentiment detection problem at each of these levels. The case of aspect-level analysis poses some other interesting questions. What are the target entities in a given text? What are the various aspects or features of the target entity that opinion is being expressed about? Any successful technique must solve these challenging problems in order to produce useful results. An important step towards solving these problems is the construction of an aspect (or feature) lexicon for a given corpus or domain, which will be further discussed later in this thesis.

Then there is the primary problem of quantifying the sentiment expressed at any of the levels mentioned above. Researchers have realized that certain words and phrases, known as sentiment words and sentiment phrases, are important, albeit neither exclusive nor even reliable, indicators of sentiment within a given text or portion thereof. A collection of these expressions with some measure of their polarity orientation is thus essential to assigning sentiment values to an aspect, sentence, or document. The construction of such a collection, commonly called a polarity or sentiment lexicon, is another important task of sentiment analysis [15, 9, 28, 27, 29, 42]. Equipped with a sentiment lexicon, researchers use a combination of bag-of-words analysis, rule-based analysis, and other approaches together with Natural Language Processing (NLP) techniques to obtain sentiment values at the desired level.

[1] Although sentiment analysis is not limited to text and a great amount of opportunities exist in analyzing sentiment contained in audible and visible human expressions, the present work will assume for the sake of simplification that "sentiment analysis" and equivalent terms refer to the same as applied to natural language texts.

Aside from the more obvious challenges mentioned above, there is yet a growing list of problems that a sentiment analysis researcher faces along the way; for example, capturing comparative opinions (e.g., of type: *A* performs better than *B*), filtering out opinion spam, detecting sarcasm and irony, adopting sentiment lexica to particular opinion domains, disambiguating word senses, and increasingly when working with data from online social media, correcting spellings and dealing with unconventional language.

Many algorithms and techniques have been developed to tackle these problems, sometimes individually sometimes in tandem with other approaches. Even though some benchmark datasets exist for individual problems, a comprehensive system to determine reliable accuracies for different approaches is still not available which makes it difficult to compare various methods [27]. Thus, a researcher must find creative ways to compare their proposed algorithm or technique with an existing one. There is a need for a system that can house implementations for baseline and state-of-the-art methods and provide their performance on any given dataset.

This thesis presents Sentiment Analysis Research Environment (SARE), a modular, extendable, open source, and web-based framework developed with the aim of filling this gap. SARE is a generalization and extension of our tool presented previously in [19, 18, 46], and provides various tools for managing opinion corpora, extracting domain information from these corpora both manually and automatically, and performing various sentiment analysis tasks on them such as aspect- and document-level sentiment analysis. The goal of this platform is to provide an environment that integrates baseline and state-of-the-art approaches to various sub-problems of sentiment analysis. Such a system will allow researchers working on a particular sub-problem to see the effects of their research on the overall problem and compare the performance of algorithms that they develop with existing baseline and state-of-the-art ones. At the same time, industry users – for example a public relations firm – can use SARE to analyze opinions on products, persons, events, or other entities of interest.

Given the vastness of the field of sentiment analysis and the limited scope of this research, our system currently only deals with certain facets of the aspect lexicon extraction problem. An aspect lexicon is a simple two-level ontology consisting of aspects and aspect expressions where aspects are features of the target domain and aspect expressions are terms used by opinion holders to express the aspects. While there are several methods proposed in the literature to automatically or semi-automatically extract aspect lexica from domain corpora, true or gold-standard aspect lexicon for a new domain needs to be defined manually. Since this can be a very time consuming task for a human, we approximate aspect expressions with corpus nouns and propose a set cover approximation algorithm to find the smallest set of documents needed to create the aspect lexicon. We also propose a Sup-

port Vector Machine (SVM) based machine learning algorithm to automatically extract aspect expressions from a corpus, which can be used as a baseline for aspect expression extraction techniques. We additionally augment our aspect extraction tools with a baseline aspect-based sentiment analysis engine that can be used by researchers to compare with new approaches.

Notwithstanding the limited set of operations supported at present, the fact that the system has been consciously built in a modular fashion means that new modules can be easily added to extend existing functionality and support an even wider array of tasks. When this system achieves a higher level of maturity, we envision that those interested in opinion trends will be able to use it to analyze sentiment contained in data that they provide and sentiment analysis researchers will be able to compare the performance of their approaches with baseline and state-of-the-art ones. The latest version of SARE can be accessed online from `http://sare2.sabanciuniv.edu`.

The rest of the thesis is laid out as follows. In Ch. 2, we provide further background to the problem with a discussion of related work. The specific problem dealt with in this work is defined and formalized in Ch. 3, while Ch. 4 and 5 are devoted to describing the architecture of our system and proposed algorithms for solving the problem. We discuss implementation details and experimentally show the performance of our algorithms in Ch. 6. Finally, we conclude in Ch. 7 along with directions for future work on SARE.

2

BACKGROUND AND RELATED WORK

The work presented in this thesis is connected with the broader field of sentiment analysis and more specifically with the problem of aspect lexicon extraction. In the previous chapter, we introduced sentiment analysis as a multidimensional and complex problem that not only poses great challenges, but also furnishes equally rewarding opportunities. Consequently, the breadth of research being conducted in this field and the volume of work being produced has dramatically increased over the last decade or so. Some of the earlier works such as [53, 52, 16, 44, 15] and others explored the concepts of beliefs, points of view, perspectives, and semantic orientations. While the terms "sentiment analysis" and "opinion mining" started appearing around 2003, some researchers had already started venturing into the territory of sentiment classification earlier in the century as evidenced by works such as [51, 48]. Some excellent surveys of the sentiment analysis field and literature have been presented in [39], [31], and [29] that shed more light on the history and current state of the field of sentiment analysis at large. To the best of our knowledge, there is no single system that has been designed with the aim of bringing all major sentiment analysis techniques into one environment. Having said that, there are tools that support operations in the larger fields of machine learning and data mining such as Weka cited in [14] and RapidMiner described in [36], both of which provide further inspiration for building an integrated system for sentiment analysis. We are similarly unaware of an online system that supports the extraction of gold-standard lexica in an interactive manner.

Using aspect extraction for sentiment analysis has also been studied extensively in the literature. In this area, [17] introduced a technique that uses association rules to find the most frequent nouns in a given set of documents. Based on these rules, a set of aspects can be synthesized for the domain. Another work presented in [40] uses a similar concept: it first finds frequent noun phrases from the opinion corpus and then extracts the product's parts and properties based on point-wise mutual information scores between these phrases

and meronymy descriptors related to the product. The technique presented in [30] mines the pros and cons field available in some online reviews and uses sequential pattern rules to learn aspect. In [13] and [34], the authors suggested using a clustering algorithm for aspect identification. These techniques only focus on the overall aspects and not the expressions associated with them. In wider domains, this approach would yield large aspect sets not necessarily useful for aggregation and summarization. In our approaches, we assume that the real set of aspects is limited and that various expressions are used to represent these aspects in opinion documents. It is these expressions, and not the actual aspect descriptors that we find in our documents.

Several approaches for utilizing frequency-based information to extract aspects have been proposed in [45, 61, 32]. These approaches use various measures such as TF-IDF, Cvalue, and information distance to identify aspect expressions within a corpus. A method presented in [21] takes into account the connection between a term and its related opinion information. They split each document into sentences before processing. Blair-Goldensohn et al. [3] have also reported a sentiment summarizer with aspect information for local service opinion documents. In this work, a double-propagation technique was employed, which makes use of the relationship between sentiment expressions and aspect expressions to discover both in conjunction with each other. A natural language dependency parser can also be used for discovering these relations as has been reported in [54, 43]. We have also used frequency-based information in our techniques, but have not experimented with double-propagation, which provides some inspiration for future work.

Supervised learning is another approach that is frequently used for aspect discovery. Hidden Markov Models (HMM) is a commonly applied supervised technique, which was used by [23] to extract aspects from opinion documents. A Conditional Random Fields (CRF) learning model has also been shown by [20] to provide good performance, and Yang and Cardie in [55] have adopted the use of semi-Markov CRFs to improve the results of methods based purely on CRF. Another CRF method has been utilized in [59] to extract product aspects. This method combines frequency, syntax tokens, and domain knowledge to find aspects. The induction of domain knowledge is aimed to improve the quality of extraction. In [56], a one-class SVM is first used to identify aspects. Synonym clustering is then performed on these aspects to eliminate duplicates. A combination of supervised and semi-supervised methods have also been used such as Naïve Bayes combined with a multi-view semi-supervised algorithm in [12] and [41]. Supervised learning requires labeled data, which is not always available and often expensive to create. Our approach to aspect expression discovery is based on a bootstrapped semi-supervised algorithm that does not require any labeled data.

Topic modeling approaches are also increasingly being used for aspect discovery. Mei et al. proposed a topic-sentiment mixture model in [35] that uses the Probabilistic Latent Semantic Analysis (pLSA) topic model combined with HMM to discover topics and extract sentiment dynamics from weblogs. Similarly, a combination of Bayesian frameworks and Latent Dirichlet Allocation (LDA)-style topic modeling was suggested in [4] that harvests the pros and cons fields of certain review formats to find aspects in review texts. Titov and McDonald proposed a statistical model called the Multi-Aspect Sentiment model, which uses multi-grain LDA to learn aspects and aspect-based sentiment predictors for sentiment analysis. Several other joint topic-sentiment modeling techniques that extend LDA have been suggested, examples of which can be found in [26, 5]. Most of these joint techniques do not separate aspect and sentiment expressions, for which a joint model of Maximum Entropy and LDA was proposed in [60] which leverages syntactic features to separate aspects and sentiment words. This approach uses a supervised method and therefore requires some amount of labeled data. The case of entities with few reviews, the so-called cold start problem, has been dealt with recently by [37] using an adaptation of LDA called Factorized LDA and has been shown to provide promising results. Most of these topic modeling techniques do not separate sentiment expressions from aspect expressions, a differentiation that is crucial to the problem we have chosen to tackle. Our approaches focus on aspect lexicon extraction independently of sentiment expressions.

As previously mentioned, much of the extant literature follows the assumption that aspect expressions appear as nouns or noun phrases in opinion documents. This assumption can be utilized in several ways to provide a starting point for extracting true aspect expressions. Hu and Liu further extrapolate from this assumption in [17] that frequent nouns within a corpus have a higher likelihood of being aspect expressions. This is an assumption we have utilized in our work as well. In the OpinionMiner system presented in [22], a bootstrapped machine learning algorithm augmented with linguistic features has been used to extract aspect expressions from documents. Their algorithm provides very high accuracy, but it is optimized for the camera domain and not altogether domain independent. However, an exploration into combining their technique with ours will provide an excellent opportunity for future work. In [58], a novel approach to grouping aspect expressions has been presented which utilizes aspect expression contexts to classify each expression into an aspect. In our work, we have used a similar technique to classify candidate aspect expressions as being aspect expressions or not. Mukherjee and Liu in [38] suggested a semi-supervised model called Seeded Aspect and Sentiment model, which allows the user to specify some seed categories and uses a variation of LDA to discover aspect and sentiment words. While the use of LDA offers many opportunities for this task, our approach does not use user-provided seed words. Experimentation using our semi-supervised technique with an LDA-based approach would be a worthy area for fu-

ture research. Another interesting approach called OFESP is presented in [57], which extracts aspect expressions based on sentiment patterns. While our approach does not use explicit sentiment patterns, we will compare our results to those presented in this paper for evaluation.

3

PRELIMINARIES AND PROBLEM DEFINITION

We have previously highlighted the vastness of the sentiment analysis research area by listing some of the sub-problems and sub-tasks of the larger problem. Since building a complete environment that encompasses this entire field is a massive undertaking beyond the scope of this thesis, the problem tackled here is limited to: 1) developing and introducing a research environment that can be incrementally extended to include support for solving various problems and performing different tasks in the sentiment analysis domain; and 2) proving the capabilities of the aforementioned environment by tackling one particular sentiment analysis problem – that of domain aspect lexicon extraction – and providing an integrated solution for the same. In this chapter, we will define these problems more specifically before presenting our solution.

3.1. Definition of Terms

Similar to other scientific fields, sentiment analysis has accumulated a large vernacular of varied and often synonymous technical terms. In order to prevent confusion, we have attempted to use uniform language and avoided using terms interchangeably in this work. While the previous chapter touched upon some of this jargon, an informal but more definitive introduction to some essential terms will help contextualize the problem and provide a better basis for understanding the proposed approach. Bing Liu, a prominent researcher in this area, has provided more standard and formal definitions that can be found in [28, 27, 29].

Definition 1 (Opinion Corpus): An opinion corpus is a collection of opinion documents. Documents in an opinion corpus are presumed to belong to a particular opinion domain. When a corpus is large enough, it can be assumed to encompass all of the terms and expressions generally used in that domain.

Definition 2 (Opinion Domain): An opinion domain is a general but consistent and finite subject towards which opinion can be expressed. All terms and expressions within a domain can generally be assumed to carry the same meaning and connotations. Alternatively, we can say that texts within the same domain use similar expressions to describe opinion targets and sentiment towards those targets. "Hotels," "cars," "movies," etc. are all examples of opinion domains.

Definition 3 (Opinion Document): An opinion document is a single coherent text that represents a collection of opinions expressed using some sentiment expressions about a target entity or aspects thereof. The aggregation of all opinion polarities is the overall document polarity. Some sentiment analysis problems de-contextualize the individual words in a text and treat it as a collection or bag of words. Opinion documents can be any opinionated text such as a review, user comment, or blog. The following is a snippet from a hotel review:

> The hotel is in a good location – not far from Circular Quay, Opera House, Bridge and Darling Harbour. Shops are also close by. Room has everything you need – we paid a special rate due to construction, so we were pleased with what we paid for ($105 for the night).

Definition 4 (Opinion Polarity): A classification of the orientation of a given opinion expressed on a uniform (discreet or continuous) scale is termed as the opinion (or sentiment) polarity. As an example, the above review snippet could be classified as having a positive polarity (as opposed to negative or neutral), or it can be said to have a polarity of 4 on a scale of $0 - 5$ and so on.

Definition 5 (Sentiment Lexicon): A sentiment lexicon is a collection of sentiment expressions along with respective opinion polarity information. Sentiment lexica can be domain-specific or general; the former usually provide better performance on domain-specific data but generating one for each new domain can be expensive. SentiWordNet presented in [9] is an example of a widely-used general sentiment lexicon.

Definition 6 (Sentiment Expression): A sentiment expression is a word or phrase that can be used to express sentiment about a particular topic. In a sentiment lexicon, sentiment expressions are defined by their sentiment polarities expressed on a uniform (discreet or continuous) scale and the contexts in which those polarities are valid. For example, SentiWordNet provides three polarities for each word/Part-of-Speech (PoS) pair: negative, neutral, and positive; all of which add up to one. According to this scheme, the word "good" appearing as an adjective has a 0.005952 negative, 0.386904 neutral, and 0.607142 positive polarity. Based on these values, one could deduce that the adjective "good" has a much higher probability of appearing in a positive context then in negative or neutral ones.

Definition 7 (Aspect Lexicon): An aspect lexicon for a given opinion domain is a two-level ontology consisting of aspects and the aspect expressions associated with each aspect.

Definition 8 (Aspect): An aspect (or feature) of a domain or entity denotes a certain characteristic of the domain or entity that can be subject to opinion. In opinion documents, sentiment is often expressed on aspects of the target entity as well as the target entity itself. If we consider the "hotel" domain as an example, "staff" and "cleanliness" could be some of the possible aspects.

Definition 9 (Aspect Expression): An aspect expression (or keyword) of an aspect is one of many possible expressions of that aspect within a particular domain. A given aspect may have several expressions that are considered to be synonymous to each other within that domain. For example, any mention of the terms "housekeeping" or "bellboy" in the "hotel" domain will naturally be taken to refer to the "staff" aspect. Most of the work in this field assumes, as does the present work, that aspect are expressed as nouns or noun phrases.

Definition 10 (Sentiment Analysis Engine): A sentiment analysis (or opinion mining) engine is an implementation of a sentiment analysis algorithm that calculates sentiment polarities at one or more levels of sentiment analysis. Thus, opinion mining engines can be document-based, sentence-based, aspect-based, or a combination thereof.

3.1.1. Natural Language Processing (NLP)

NLP is a field of computer science concerned with formally expressing semantic and syntactic information contained in various forms of human language and vice versa [24, 1, 7]. As such, sentiment analysis is a specialized NLP problem and thus relies heavily on NLP techniques [29]. While there are several NLP operations that can be used to assist with sentiment analysis, this work only uses three of them as defined below.

Definition 11 (Text Segmentation): In text segmentation, a natural language text is broken down into its constituent parts using boundary markers combined with mechanisms that account for irregularities in the use of these boundary markers. Generally speaking, text segmentation is used to divide the text into sentences or words.

Definition 12 (Part-of-Speech (PoS) Tagging): PoS is a term used in linguistics to refer to the linguistic class of a word that describes its grammatical or morphological function within a sentence. The same word can have different PoSs in different contexts. PoS tagging is the process through which each word is assigned to one of the several PoSs such as noun, adjective, verb, adverb, etc.

Definition 13 (Syntactic Parsing): Syntactic parsing attempts to deconstruct sentences in order to reveal the grammatical relationships and dependencies between words. This information can be used to determine the effect of words on each other and is generally represented as a dependency graph.

3.2. Research Environment

The primary aim of this work is to provide the basis for an environment that can be used by sentiment analysis researchers and practitioners to perform tasks and solve problems pertinent to their fields. Such a platform must be architected from the ground up in a systematic way to adhere to standards that will permit it to successfully scale to its envisioned potential. We will define here these standards and traits that the system needs to maintain in order to create the promise for an all-encompassing research environment.

3.2.1. Incremental Extendability

The platform must be designed so that it can be built up to include solutions for a wider array of sentiment analysis problems and tasks. Here, we introduce the concept of modules, which is a unit of the system that performs a coherent set of operations. The system itself would be a collection of such modules in addition to the supporting logic that glues them together to form a fully functional application. Extendability is guaranteed by providing a convenient mechanism for addition of new modules, and therefore, the architecture of the system must ensure minimal inter-dependency within the modules.

3.2.2. Accessibility

Nowadays, the World Wide Web is the foremost platform for information and services consumption, and desktop applications tend to be cumbersome to install and maintain. The Software as a Service (SaaS) delivery model, realized through web services, provides a convenient and easily accessible way for external users to interact with systems and perform operations. For our system to be useful in a variety of circumstances and to a wide range of customers, it should be conveniently accessible both as a web site and a web service. Users who wish to use the services of this system may access them through the website and perform desired tasks while other systems that wish to utilize the same services may access them through an easily consumable web service structure.

3.2.3. Open Source

Open-source software creates more opportunities for extendability by allowing community input and extension. Since the development of this platform is an ambitious project that requires broader contribution and support from the sentiment analysis research community, it should use open-source technologies and libraries, and should in turn make itself available as an open-source project.

3.2.4. Multilingual Support

Sentiment analysis is a wide area of research with applications in all languages and domains. While some algorithms are language-independent, many tend to be dependent on the language of the target text and their correct operation is consequently contingent on the presence of an NLP package for the target language. Therefore, the extendability provided by our proposed environment must also include support for multiple languages. Specifically, it should provide a convenient mechanism for adding NLP packages for languages other than English as well as a transparent method for accessing the NLP functionality thereof.

3.3. Aspect Lexicon Extraction

In sentiment analysis, domain aspect lexicon extraction is crucial for gaining a deeper and more refined understanding of the opinions expressed in a given document [39, 27]. Without domain-specific aspects, the sentiment analysis process remains prone to generalizations and dilution of opinions. As explained stated, a domain aspect lexicon is a two-level ontology that consists of a set of aspects that are broad features of the domain; and for each aspect, a set of aspect-related expressions that represent those aspects in text. For example, in the "hotel" domain, "room quality" might be one such aspect and the terms "furniture" and "size" could be keywords associated with this aspect.

The problem of extracting such lexica is well-recognized within the sentiment analysis domain. In this work, we tackle two sub-problems of aspect lexicon extraction, viz., 1) creation of gold-standard aspect lexica; and 2) aspect expressions extraction. In the following, we will further define these problems.

3.3.1. Gold-Standard Lexicon Creation

Several automatic and semi-automatic methods have been proposed in the literature to extract a domain aspect lexicon from a given domain corpus, as discussed under Ch. 2. In evaluating their methods, researchers either compare the coverage of the extracted lexicon to that of a hand-built one considered to be the gold standard; or they compare the performance of a baseline sentiment analysis system using the generated lexicon versus

some other available lexica. The gold-standard lexicon mentioned in the former case is obtained through one of the following ways: a) by manually tagging a large corpus; b) by one or more domain experts choosing aspects and aspect expressions without the use of a corpus; or c) using review sets that have already been annotated with aspects and related expressions by the original reviewers. The first approach is naturally rather tedious as domain corpora are usually too large to be manually processed. The second one is vulnerable to generalization error since the experts' vocabulary tends to be narrower compared to the broader vocabulary of a mass of reviewers. Finally, the third approach is not always applicable, since such review sets are not available in all cases. It is also difficult to verify and evaluate a hand-built lexicon to make sure that it contains all the relevant words and only the relevant words. The common aspect of the three approaches is the need for human annotation. However, the burden of human annotation should be as light as possible. Thus, we redefine our problem as that of discovering the smallest set of documents that, in a given corpus, provide the highest amount of information needed to develop a domain ontology. Alternatively, we would like to obtain the smallest set of documents that contain all of the aspect expressions contained in that corpus.

We recall the definition of a corpus as a collection of documents and consider each document as a bag of words, where the words can either be aspect expressions or regular words. Since aspects are features of the target entity that opinion can be expressed about, we can assume that they appear, in a linguistic sense, as nouns within their respective texts. This is commonly taken to be true as noted by [27] and other researchers. Thus, we can approximate the smallest set of document containing all the aspect expressions with the smallest set of documents that encompasses all the corpus nouns.

This problem of finding the smallest set of documents containing all corpus nouns can be reduced to the classical set cover problem with documents representing individual sets (each taken as a bag of words) and the corpus representing the problem universe. The set cover problem has long been known to be NP-complete due to [25], and therefore a heuristic method is required to solve it. Fortunately, several such heuristics have been suggested over the years, the most notable of which is the greedy technique of iteratively selecting sets that provide the highest coverage until the entire universe is covered [49]. This method has been proven to be the best possible approximation for the problem by [33], [10], [2], inter alia. However, for large corpora, a greedy algorithm that iterates over the entire dataset at each step is still not the most optimal solution. Furthermore, set covers produced by the canonical greedy algorithms are still too large to afford human consumption. This is often due to the fact that they contain a number of sets that, when considered in conjunction with other sets, contribute very little to the universe at large, but which the algorithm is not designed to identify and eliminate. To account for these shortcomings, we will suggest an alternative approach in Ch. 5.

3.3.2. Aspect Expression Extraction

In dealing with the problem of aspect-based sentiment analysis, we are motivated by the desire to understand opinion expressed on the various aspects of an entity. This is not trivial since these aspects are not always known in advance and need to be discovered from data. Furthermore, it is known that various words and phrases can be used to refer to the same aspect which makes the task even harder to accomplish. The problem of aspect expression extraction deals with the discovery of such words and phrases from a corpus. The extracted expressions can then be grouped into aspects using other algorithms that have been reported in the literature for this task as discussed under Ch. 2.

Once again, we start by assuming that aspect expressions are a subset of corpus nouns. Our problem in this context is to provide a binary classification method to separate aspect expressions from regular nouns. We will present such a scheme in Ch. 5.

4

SYSTEM DESIGN

SARE is designed to be a modular platform. This modularity facilitates extendability within the system, which has previously been highlighted as a basic requirement for such an environment. In this section, we will outline the architectural and functional specifications of the system as well as explain some of the more crucial mechanisms that contribute to the modularity of the environment.

4.1. Application Layers

The architecture of SARE follows a layered design pattern; that is, it consists of several layers, each of which performs functions at a particular level of the system and provides abstraction for layers at higher levels. This pattern allows for separation of concern within the layers and makes it easier for each layer to operate while being oblivious to finer details of operations that take place at lower levels. The overall architecture of SARE, as depicted in Fig. 4.1, is divided into four main layers that are discussed below.

4.1.1. Persistence Layer

The persistence layer stores data representing final as well as intermediate results of module operations in a relational database. To anticipate addition of new modules and data objects, the database is designed in a very flat manner with only two tables so as not to require changes to the database model every time a new module or data object is introduced. The first table (`persistent_objects`) stores the actual data and only contains

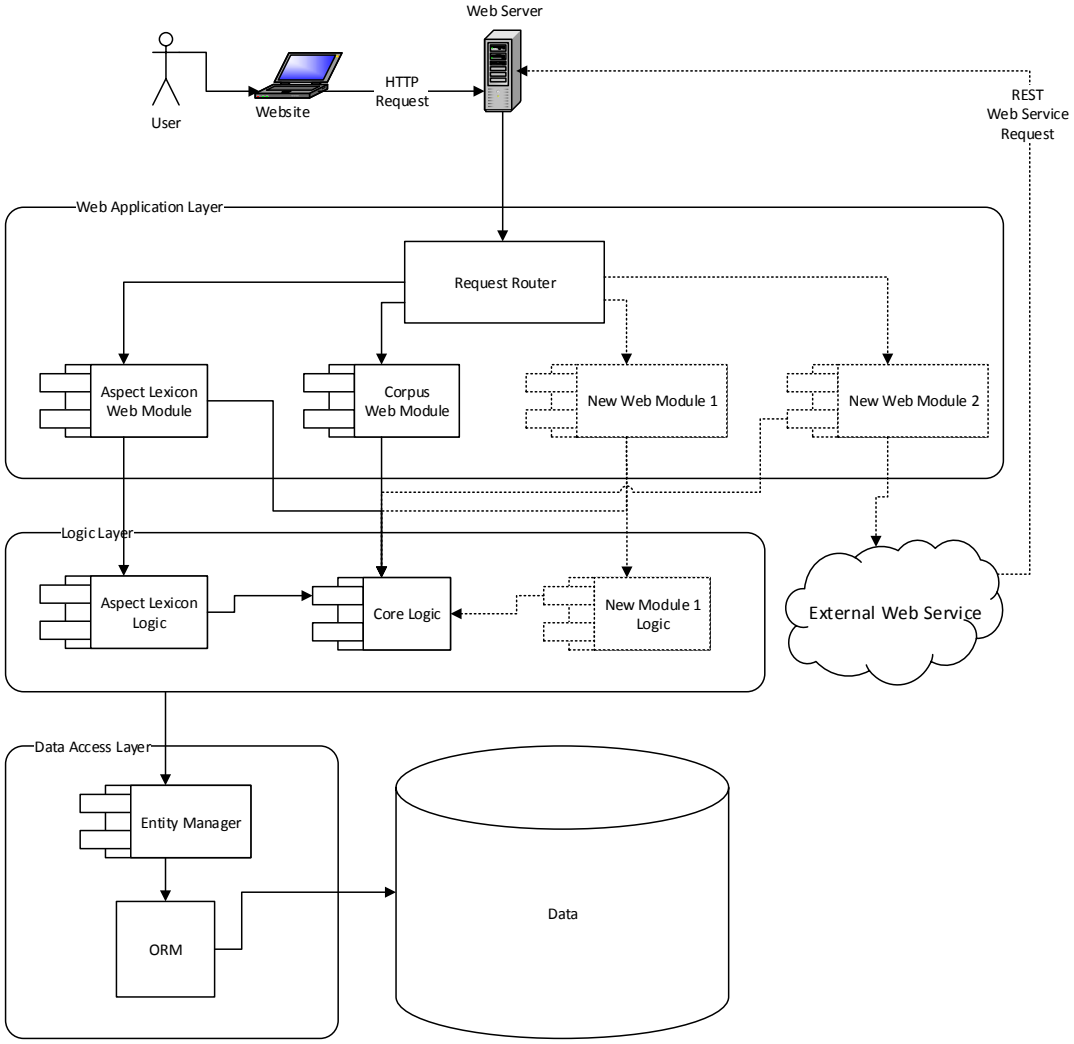

Figure 4.1: Overall architecture of SARE

such fields as are essential to the entire hierarchy of data objects. In this scheme, while not all data objects make use of the full set of columns available, we achieve better query performance as a trade-off. Additionally, this table has a multi-purpose column, which can be used to store any arbitrary data in the JavaScript Object Notation (JSON) format, thereby allowing the table to support any number of logical columns. It should be noted that this extendability comes at the cost of the ability to perform reliable database-level queries on these logical columns, which must instead be performed in the data access layer. The second table (`jt_object_references`) is used to maintain many-to-many relationships between the data entities. Fig. 4.2 shows a simplified graphical representation of this database.

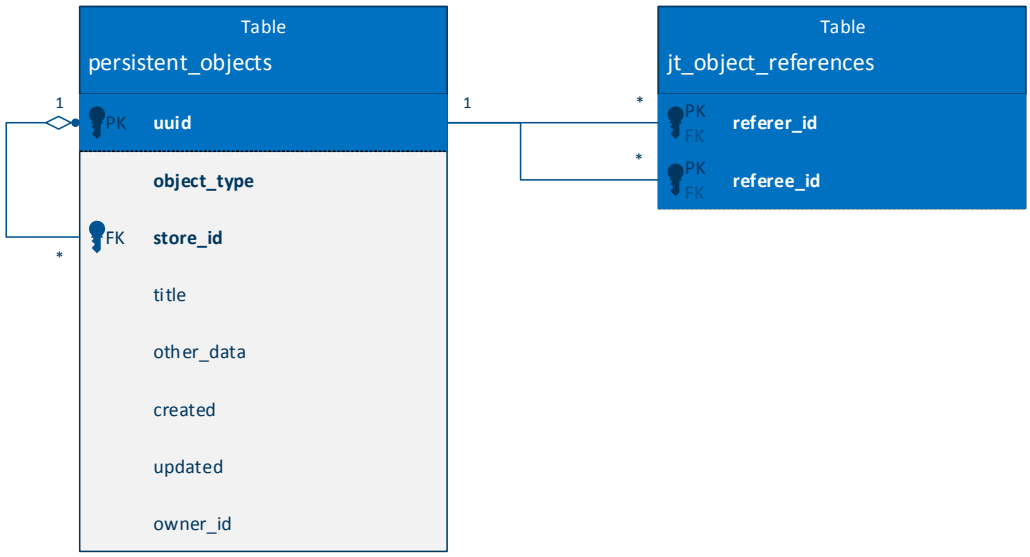

Figure 4.2: Simplified database design

4.1.2. Data Access Layer

The data access layer provides an abstraction between the persistence and higher-level layers. We use an Object-Relational Mapping (ORM) library to manage data access as well as the database itself. To provide better query performance and to mirror the database setup described above, we use a single-object hierarchy to model our data objects. A simplified class hierarchy of data objects is shown in Fig. 4.3. Conceptually, data objects all derive from the same type (`PersistentObject`) and are divided into two main types: documents (`PersistentDocument`), and document stores (`PersistentDocumentStore`). Documents are used to store units of information such as opinion documents and aspect expressions, and document stores are used for organizing multiple documents into collections of related documents such as opinion corpora and aspect lexica.

4.1.3. Logic Layer

This is where the primary logic of the application is placed. It houses data objects and algorithms for creating and manipulating these data objects, which for the most part mirror the concepts mentioned in Ch. 2 and derive from the objects mentioned above. Details on algorithms used and other primary logic will be presented in Ch. 5. Since the logic

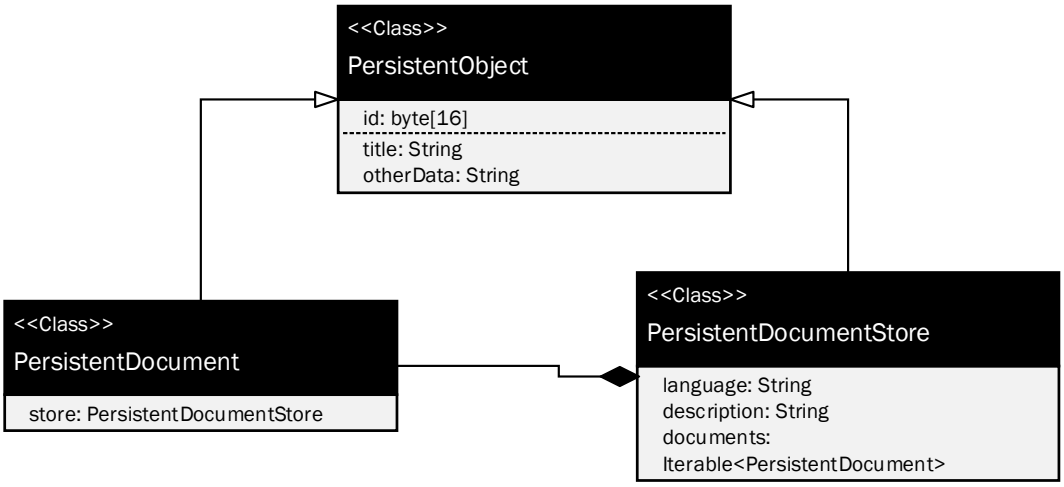

Figure 4.3: Simplified hierarchy of data objects

layer is the most crucial part of our application and the one most prone to errors, this layer contains a robust unit test suite to ensure code correctness across releases and code changes.

4.1.4. Web Application Layer

This layer contains the presentation logic of the application and uses the Model-View-Controller (MVC) paradigm. The MVC pattern consists of models that represent the data, views that represent the display logic, and controllers that represent the manipulation logic of an application. Controllers build both models and views, where models are based on underlying data objects and views are provided access to these models so that they can display the data. Additionally, the web application is built on the REpresentational State Transfer (REST) architecture presented in [11], which makes it a highly robust web service as well as a website. These web services can also act as Application Programming Interfaces (APIs) for external applications seeking to leverage the functionality of SARE independent of the website. Fig. 4.4 shows the basic architecture of our MVC application.

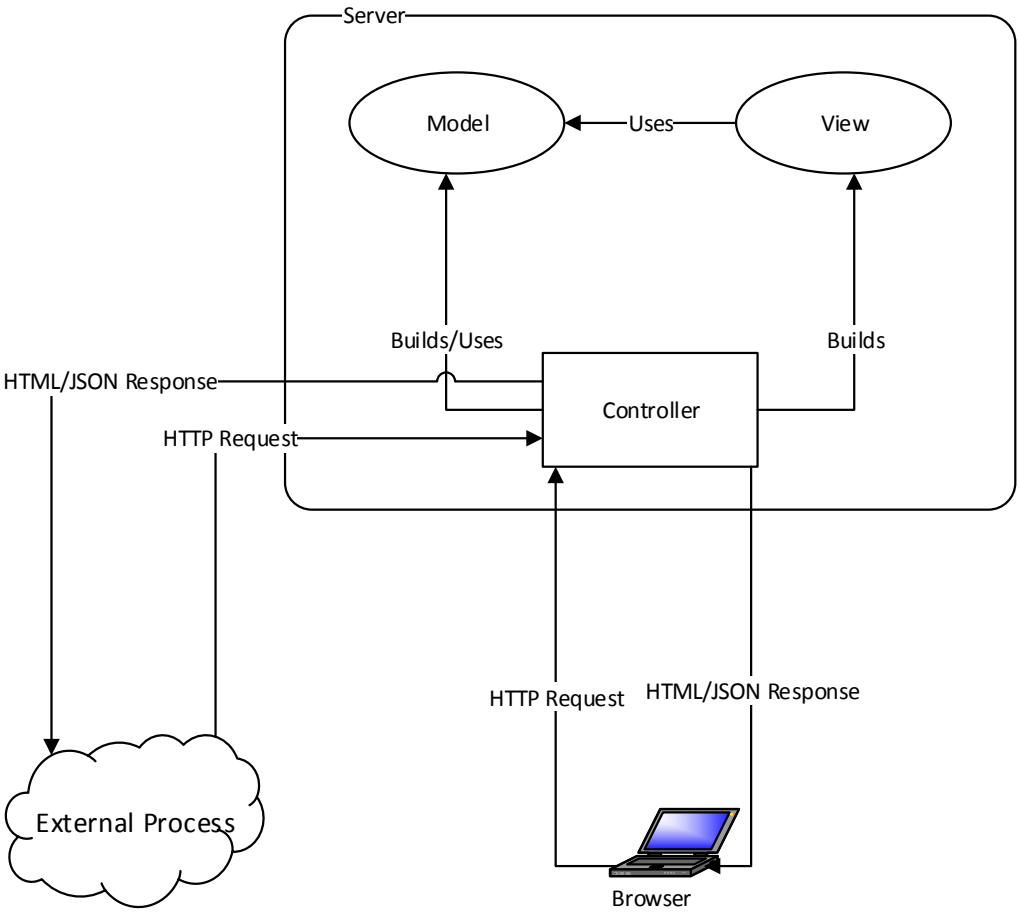

Figure 4.4: Basic architecture of the MVC application

4.2. Module Definition and Workflow

Modules provide cohesive functionality or perform operations on a particular kind of objects and are naturally the building blocks of SARE. While conceptual modules may exist at the logic layer level, they are more formally defined in the web application. It is worth noting that the architecture represented in Fig. 4.1 shows only a selection of modules available and new modules can be introduced by adding them in the web application layer. Supporting primary logic can either be placed in the logic layer or even in an external web service that the web module communicates with.

The design of SARE favors a workflow-type interaction; i.e., the user takes their data through a series of steps to obtain the final result. Each of these steps is handled by a module which performs a specific operation and provides a certain type of result. The user is then presented with a list of modules that can utilize this type of result and the process continues.

From the above description, it should be clear that each module has its own competencies; that is, it can operate on certain types of data and produce a particular type of output. For example, a module that builds aspect lexica might accept a document corpus as its input and produce an aspect lexicon as its output, which can then be consumed by yet another module. To facilitate this behavior, each module defined in the web application is annotated with the types of data objects it can operate on. As depicted in Fig. 4.5, when the website receives output from a module, it sends the same to the module resolver. The module resolver, based on the annotated module inputs, determines modules that are able to consume that result and provides to the website a list of possible next modules. The website then displays this list for the user to make their selection.

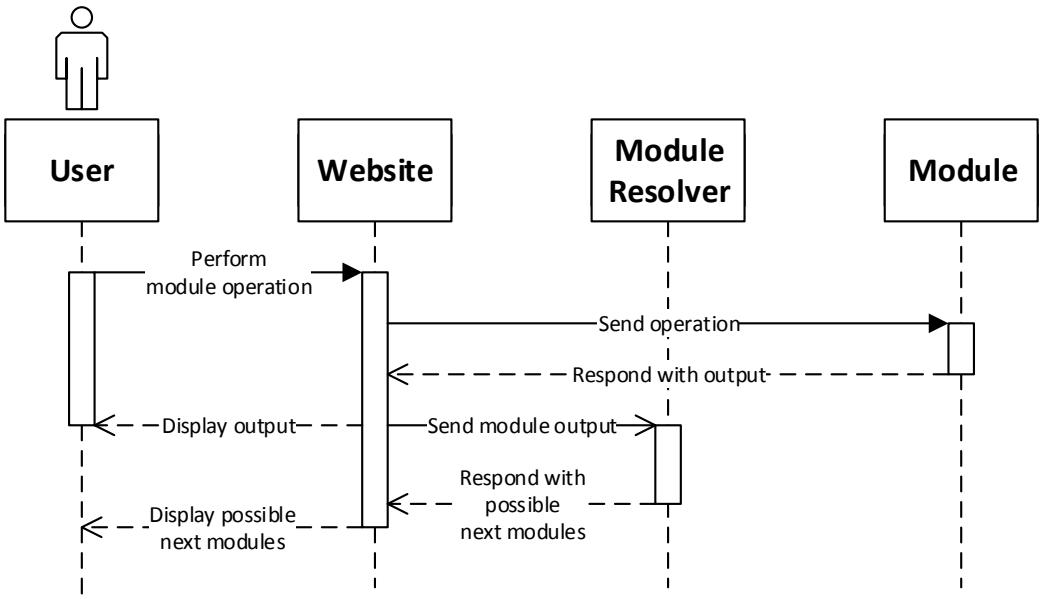

Figure 4.5: Module resolution sequence

4.3. Multilingual Support

We have designed SARE such that support for any language can be added to the system with minimum setup provided that an NLP package is available for that language. New languages can be defined by introducing wrapper classes containing language-specific NLP packages. We use the object factory design pattern to allow for transparently generating language processors for any of the supported languages. Since each document corpus stores information about the language of the corpus, any algorithm operating on that corpus can use the object factory to create a language processor and use it to process that language in an abstract manner. Fig. 4.6 shows a reduced class hierarchy of this factory design pattern.

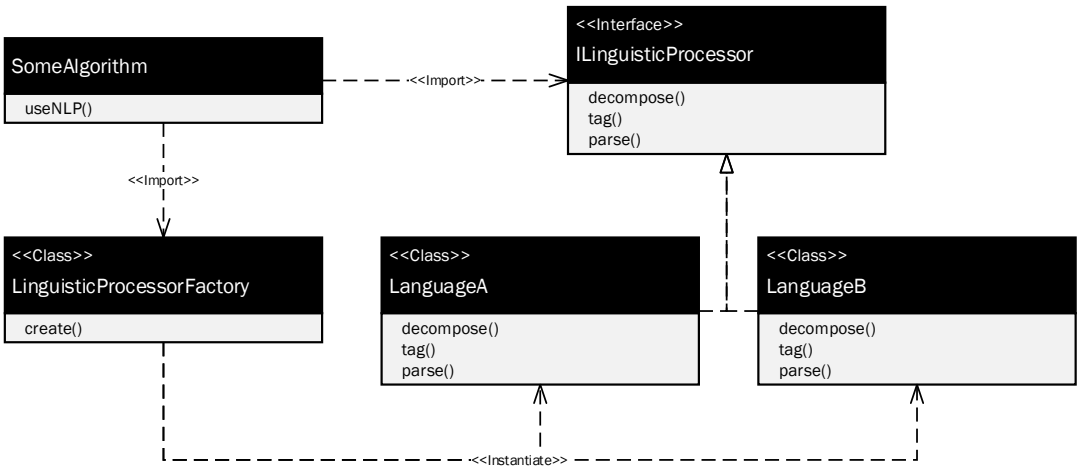

Figure 4.6: A reduced class hierarchy of the linguistic processor factory design

5

MODULES AND ALGORITHMS

SARE is a multidimensional sentiment analysis research platform that can contain numerous modules, each providing unique services and performing useful sentiment analysis tasks. However, considering the limited scope of this research, we have chosen to showcase a few selected modules with the expectation that future research will extend the functionality of this application to fulfill its potential as a larger sentiment analysis research environment. In this section, we will describe the current lineup of modules in SARE and explain their process and algorithms.

5.1. Corpus Reduction Module

In Sec. 3.3.1, we discussed the problem of extracting gold-standard aspect lexica and defined it as one of finding the smallest set of documents containing all the corpus nouns. We now present a generalized solution to this problem that deals with discovering the smallest set of documents containing all of a given PoS tag or a combination thereof. This generalization will serve to make this solution useful for other annotation-related tasks such as sentiment lexicon creation.

We have stated previously that the problem mentioned above is reducible to the problem of finding a minimal set cover of a collection of sets. While the classical greedy set cover algorithm has been proven to be the best approximation, we propose an algorithm inspired by the greedy heuristic that allows us to operate on large datasets more efficiently. We also keep a utility score for each set that allows us to ignore less significant sets in the set cover according to an error tolerance parameter. An informal explanation of this algorithm, which we term as being Eagerly Greedy, is given below.

We maintain a candidate set cover initialized to an empty set and iterate through the document corpus sequentially. For each document encountered, we consider the set of all PoSs of interest in that document as a new set, and attempt to sequentially match its elements to those of the candidate cover sets, i.e., members of the candidate set cover. Each time a candidate cover set consumes (presumes covered) an element, it increments its own utility score by one and removes the element from the new set. If the candidate cover set is a subset of the new set, then the candidate cover set is itself entirely consumed and replaced by the superset (with utility score being set to the sum of existing utility plus the size of the new set). This process continues until either all of the elements in the new set are consumed or we run out of candidate cover sets to process. In the latter case, a new candidate cover set is formed from the contents of the uncovered elements with the utility score being initialized to its size. This algorithm has been formalized in Alg. 1.

Algorithm 1 An eagerly greedy minimal set cover approximation algorithm

Precondition: $\mathcal{D}$ is a set of documents

Precondition: ExtractSet is a function that extracts expressions of interest from a given document

```
1:  function SetCover(𝒟, ExtractSet)
2:      D̂ ← ∅
3:      for all D ∈ 𝒟 do                          ▷ Do for each document in the corpus
4:          X ← ExtractSet(D)                      ▷ X are uncovered elements
5:          for all D̂ ∈ D̂ do                       ▷ Do for each set cover document
6:              if X ⊃ D̂ then
7:                  D̂ ← X
8:                  D̂.utility ← D̂.utility + |X|
9:                  X ← ∅
10:             else if D̂ ∩ X ≠ ∅ then
11:                 D̂.utility ← D̂.utility + |D̂ ∩ X|
12:                 X ← X − D̂
13:             end if
14:             if X = ∅ then
15:                 exit for
16:             end if
17:         end for
18:         if X ≠ ∅ then
19:             D̂ ← X
20:             D̂.utility ← |X|
21:             D̂ ← D̂ ∪ {D̂}
22:         end if
23:     end for
24:     return D̂
25: end function
```

The result of the above-mentioned algorithm is an approximate minimum set cover of the document corpus such that all expressions of interest are represented. While the size of the dataset is reduced, we still do not achieve any improvement over the greedy set cover algorithm, and need to decrease the number of documents even further. We will suggest here a pruning technique that can be applied to reduce the size of the corpus, wherein lies the real usefulness of this algorithm.

At the end of the above procedures, we might be left with several sets having very low utility scores; that is to say, $\hat{D}.utility \approx 0$. We can therefore eliminate some of the less important sets by sorting them in decreasing order of utility and choosing the top sets whose cumulative utility makes up at least a certain fraction of the total utility. Formally speaking, we choose the top k cover sets that satisfy,

$$\frac{\sum_{i=1}^{k} \hat{D}_i.utility}{\sum_{\hat{D} \in \hat{\mathcal{D}}} \hat{D}.utility} \geq 1 - \hat{\tau}$$

where $\hat{\tau}$ is tolerance to error that is a parameter of the algorithm and can be adjusted as desired.

The application of the algorithm and the pruning step mentioned above yields a significantly reduced set of documents with a maximum error of $\hat{\tau}$.

5.2. Aspect Expression Extraction Module

This module attempts to discover aspect expressions from a document corpus using semi-supervised learning. For this purpose, we first extract all candidate aspect expressions along with their contexts from the corpus, and apply a bootstrapping technique to automatically label part of the data. An SVM classifier is then used to learn a model from this training data and applied to the entire corpus to generate a new training set. This process continues until the outputs stabilize. A detailed description of these steps follows.

5.2.1. Extracting Candidate Expressions

As noted in preceding chapters, we assume that all aspect expressions appear, in a linguistic sense, as nouns in their respective contexts. All nouns in the corpus, therefore, can be considered as candidate aspect expressions. We use a PoS tagger to identify all such candidate expressions $\hat{e} \in \hat{E}$ from the corpus. The text of each opinion document is broken down into sentences using text segmentation and each sentence containing a candidate feature expressions is considered as a document for the purposes of the learning algorithm. The following machine learning features are then drawn from all of these learning documents for each candidate expressions and their values are normalized.

Number of sentences: The number of sentences a candidate expressions appears in denotes its frequency in the corpus, which in itself is an indicator of the likelihood of the expression being a true aspect expression. Based on this assumption, we include this frequency as a machine learning feature.

PoS distribution: We expect that the words surrounding aspect expressions will generally follow a certain composition of different PoSs, an information that can be utilized by a machine learning algorithm. Therefore, we extract the number of verbs, adverbs, and adjectives surrounding each expression, each of which acts as a separate feature (number of verbs, adverbs, etc.).

Polarity: Aspect expressions are more likely to appear in highly polar contexts and the cumulative polarity of words surrounding a candidate expression can give strong clues about its status as an aspect expression. Based on this idea, we utilize the total polarity score of all surrounding words, which is the sum of positive and (unsigned) negative polarities obtained from a polarity lexicon.

Neutrality: Conversely to the above, aspect expressions are less likely to be surrounded by neutral words. Therefore, we extract the sum of neutrality score, similarly taken from a polarity lexicon, of all words surrounding a candidate aspect expression.

5.2.2. Automatic Labeling

As in all semi-supervised learning approaches, we need to assemble a set of labeled data to be used for training. Here, we utilize the assumption made by [17] that the most frequent nouns in an opinion corpus are highly likely to be aspect expressions. We further expand on this idea by making the converse assumption that the least frequent nouns in a

corpus are unlikely to be aspect expressions. Based on this assumption, our training data is extracted by labeling k aspect expressions that appear in the most number of sentences as belonging to the positive class, and k expressions that appear in the least number of sentences as negative. Here, k is given by:

$$k = \lfloor \ln(|\hat{E}|) + b \rceil$$

where $\hat{E}$ is the set of all candidate aspect expressions and b is a parameter of the algorithm and controls the extent to which automatic labeling can be relied upon. Once we have obtained labeled data L in this manner, the rest of the data is treated as unlabeled data U. It should be noted that these are soft labels and can be changed in the process of learning.

5.2.3. Semi-Supervised Learning

The labeled data L is used to train an SVM classifier. The model thus learned is used to obtain class distributions for expressions in both labeled and unlabeled sets. If either of the class score is higher than an acceptability threshold, λ, which is another parameter of the algorithm, we assign the expression to that class. Thus an expression may be reassigned to a different class if the class distribution changes after the initial labeling. At the end of such an iteration, once again, we have a set of labeled and unlabeled data that are used similarly to retrain and reclassify until the outputs stabilize. A formal definition of the algorithm is given in Alg. 2 and 3.

5.3. Aspect Lexicon Builder Module

This module can be used to manually build an aspect lexicon from a document corpus by sequentially traversing through corpus documents and trying to pick out aspects and aspect expressions from the document texts. A reduced corpus obtained from the method described in Sec. 5.1 can be used as well, thereby decreasing the annotator's burden. Since the mechanics of this module are mostly implementation-specific, we will explain its function further in Ch. 6.

Algorithm 2 An algorithm for extracting aspect expressions

Precondition: $\hat{E}$ is a collection of candidate aspect expressions ordered by descending frequency
Precondition: Machine learning features have already been extracted $\forall \hat{e} \in \hat{E}$
Precondition: $\lambda \in [0, 1]$ is a class score acceptability threshold
Precondition: $b \geq 0$ indicates the extent to which automatic labeling is to be relied upon
Precondition: TrainSVM is a function that trains an SVM classifier from the given training data and returns a classification model

1: **function** ExtractAspectExpressions($\hat{E}$, λ, b)
2: $\langle L', U' \rangle \leftarrow$ AutoLabel($\hat{E}$, b)
3: **repeat**
4: $\langle L, U \rangle \leftarrow \langle L', U' \rangle$
5: $\langle L', U' \rangle \leftarrow$ Classify(L, U)
6: **until** $L = L'$
7: **return** L'
8: **end function**

5.4. Aspect-Based Sentiment Analysis Module

The objective of this module is to use a domain aspect lexicon for calculating aspect-based and overall sentiment scores for each review and present a summarized result. Since this work was done outside the scope of this thesis, we will only present a brief description here. To calculate sentiment scores, word polarities are first obtained from a polarity lexicon. A polarity-placement algorithm is then used to calculate score values for each aspect and the overall review. Using syntactic dependencies obtained through a dependency parser, polarity values can be transferred from the polarity word to the aspect keyword, and consequently to the aspect.

Algorithm 3 Auto-labeling and classification methods for aspect expression extraction

function AUTOLABEL($\hat{E}$, b)
  $k \leftarrow \lfloor \ln(|\hat{E}|) + b \rceil$
  $\langle L^{'}, U^{'} \rangle \leftarrow \langle \emptyset, \emptyset \rangle$
  for all $\hat{e} \in \hat{E}$ **do**
    if $\hat{e} \in \text{TOP}(\hat{E}, k) \vee \hat{e} \in \text{BOTTOM}(\hat{E}, k)$ **then**
      $\hat{e}.class \leftarrow 1[\hat{e} \in \text{TOP}(\hat{E}, k)]$
      $L^{'} \leftarrow L^{'} \cup \hat{e}$
    else
      $\hat{e}.class \leftarrow null$
      $U^{'} \leftarrow U^{'} \cup \hat{e}$
    end if
  end for
  return $\langle L^{'}, U^{'} \rangle$
end function

function CLASSIFY(L, U, λ)
  $M \leftarrow \text{TRAINSVM}(L)$
  $\langle L^{'}, U^{'} \rangle \leftarrow \langle \emptyset, \emptyset \rangle$
  for all $\hat{e} \in (L \cup U)$ **do**
    $c \leftarrow \text{M.CLASSIFY}(\hat{e})$
    if $c_{+} > \lambda \vee c_{-} > \lambda$ **then**
      $\hat{e}.class \leftarrow 1[c_{+} > \lambda]$
      $L^{'} \leftarrow L^{'} \cup \hat{e}$
    else
      $\hat{e}.class \leftarrow null$
      $U^{'} \leftarrow U^{'} \cup \hat{e}$
    end if
  end for
  return $\langle L^{'}, U^{'} \rangle$
end function

6

IMPLEMENTATION AND EXPERIMENTS

We implemented SARE as proposed in Ch. 4, which is the first major contribution of this work. We also conducted some experiments on the algorithms discussed in Ch. 5. Here, we will describe the system implementation accompanied by a simple use case, and discuss the experiments with their goals, setup specifics, and results.

6.1. Implementation Details

The research environment was implemented using a combination of several state-of-the-art tools and technologies. All of these technologies are open-source and no proprietary software was used. The system is publicly-maintained and accessible as both a web service and a web site from `http://sare2.sabanciuniv.edu`, and the source code is available under the General Public License (GPL) from the same location.

The implementation employs a layered architecture as described in Ch. 4. We use a relational database provided by MySQL to implement the persistence layer. The data access layer uses Java Persistence API (JPA) backed by Hibernate as the ORM. While we have represented the data access layer separately, the set up required for it often spills in practice into the logic layer making the data access functionality not entirely distinguishable from the logic layer. The primary logic of the application is written in Java and Scala. We have also created a unit test suite for the logic layer, which is developed using the JUnit testing framework. MVC framework for the web application layer is provided by the Play Framework. In our case, the models and controllers are written in Java and some Scala, while the views are written using Play's template language, HTML, JavaScript, CoffeeScript, and LESS CSS. An outline of the different layers of implementation and the interaction of their various components is presented in Fig. 6.1.

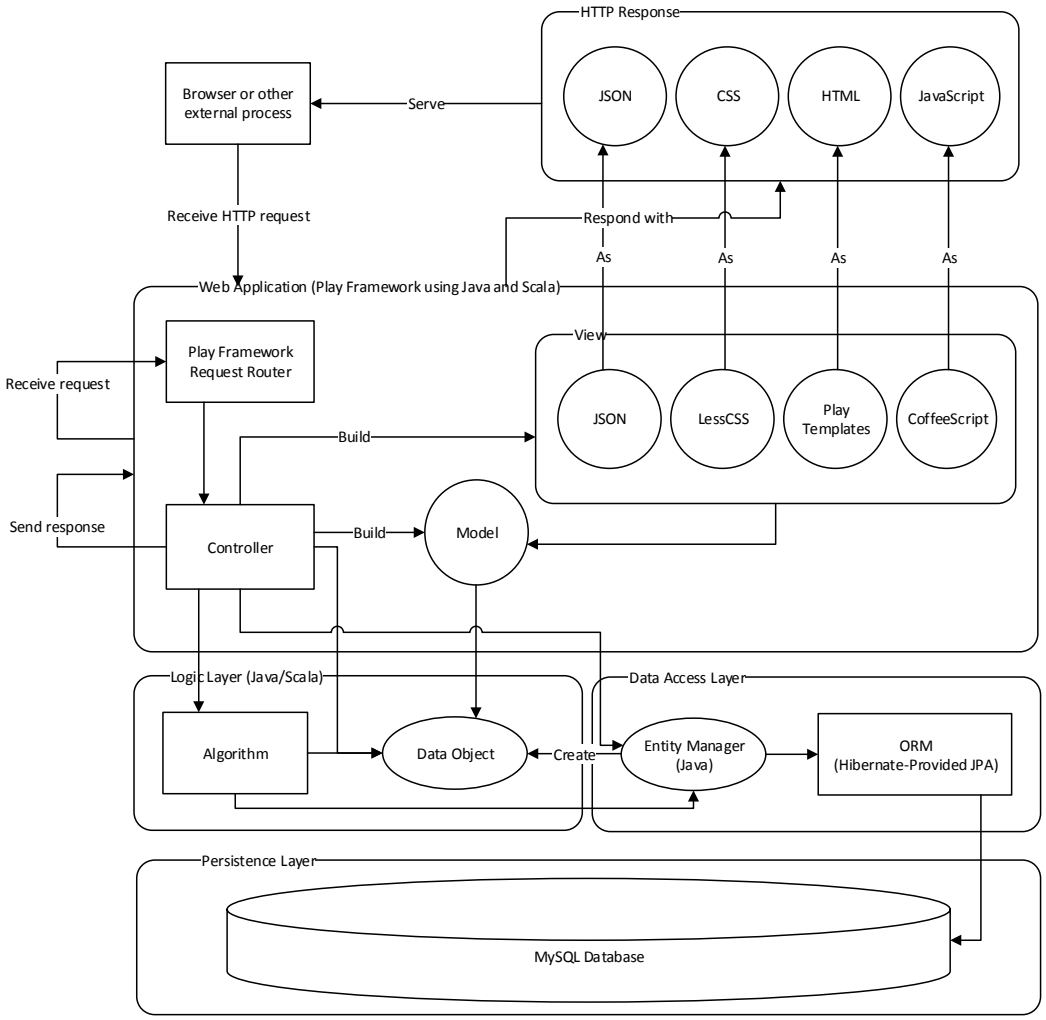

Figure 6.1: Implemented architecture of SARE

6.1.1. A Basic Use Case

We will now demonstrate the functionality of our system with a use case, in which the user attempts to create a gold-standard aspect lexicon and perform sentiment analysis on a corpus based on the created aspect lexicon. We assume that the user is already logged into the system and on the main analysis page shown in Fig. 6.2. To start, the user clicks the "Select a corpus" button and adds a new corpus by clicking the "Add" button. A new corpus is then uploaded with the help of the the interface which currently accepts specialized text, Extensible Markup Language (XML), and Comma-Separated Values (CSV) formats either as a single file or as a collection of files compressed using the ZIP format. This interaction is shown in Fig. 6.3. The "Optimize corpus" button is then clicked which takes the user to the corpus optimizer page. Here the user can select their optimization criteria, in this case by checking "Nouns" and the option to "Optimize

words with common roots" and click "Apply". The optimization engine will create an optimization profile according to the loss tolerance parameter and plot the result as shown in Fig. 6.4. The user can then select the loss tolerance they are most comfortable with, click "Apply" again, and then click the "Build aspect lexicon" button. On this page, the user can turn on options to highlight any combination of PoSs to assist them with the task of identifying aspects and expressions while traversing through the documents. Any identified aspects and related expressions can be added using the controls on the right-hand side of the page as shown in Fig. 6.5. Finally, the user can select the "Run aspect-based opinion miner" to run the opinion mining engine on the corpus using their aspect lexicon and view the results as partially shown in Fig. 6.6. A diagram detailing the activity of creating a gold-standard lexicon is depicted in Fig. 6.7 and an overview of the use case for the same activity is shown with the help of a use case diagram in Fig. 6.8.

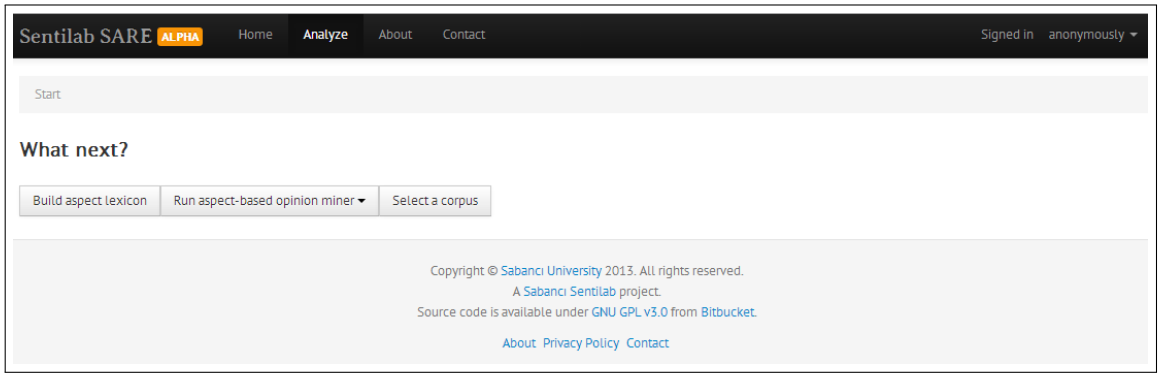

Figure 6.2: A screenshot of the main analysis page

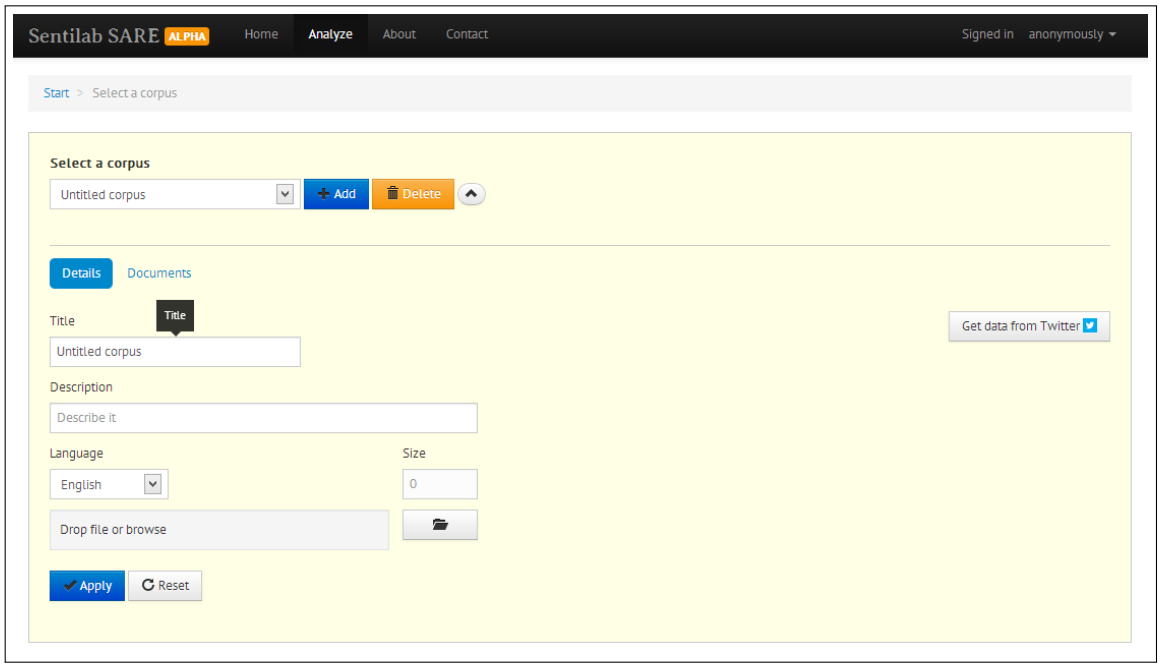

Figure 6.3: A screenshot of the add corpus page

Figure 6.4: A screenshot of the corpus optimization engine displaying the optimization profile

6.2. Experimental Results

We devised and conducted experiments to test the performance and effectiveness of the algorithms described in Ch. 5. With these experiments, we aim to show the usefulness of our algorithms and their viability as baseline operations within the larger SARE system.

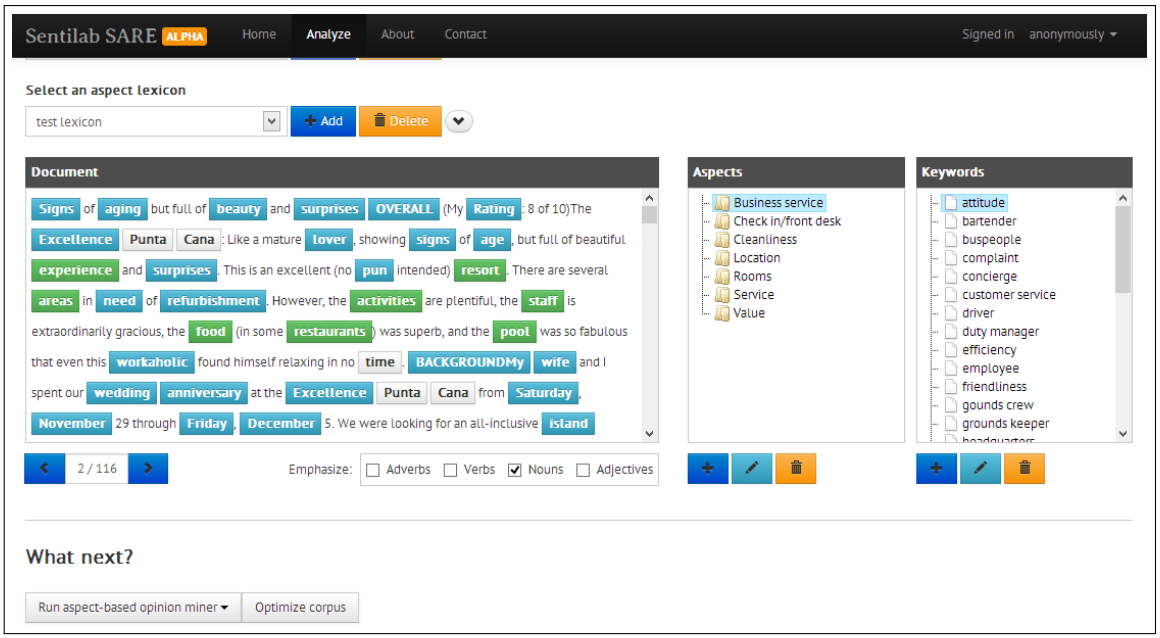

Figure 6.5: A screenshot of the aspect lexicon builder interface

6.2.1. Corpus Reduction Algorithm

This experiment measured the extent of data reduction by applying the eagerly greedy set cover algorithm presented in Sec. 5.1. In order to perform this evaluation, we need to compare the proposed algorithm with the best and the worst approximations of set cover problems. We chose a random selection approach as the worst case approximation; that is to say, randomly choosing sets from the collection until all the elements in the universe are covered. The best approximation, as referenced previously, is the classical greedy algorithm. In this comparison, we looked at the sizes of the set cover approximation computed by the individual algorithms relative to the size of the corpus, where size of each collection is given by the number of documents in the respective collection. We also experimented with various values of $\hat{\tau}$ to observe which value provides us with optimal data reduction without compromising data quality.

Since the overall goal is to reduce the amount of data presented to the human annotator, we observe the performance of these three algorithms as a function of the amount of data reduction achieved. For a given algorithm, α, the data reduction it achieves is given by:

$$\Delta(\alpha) = 1 - \frac{|\hat{\mathcal{D}}^*_\alpha|}{|\mathcal{D}|}$$

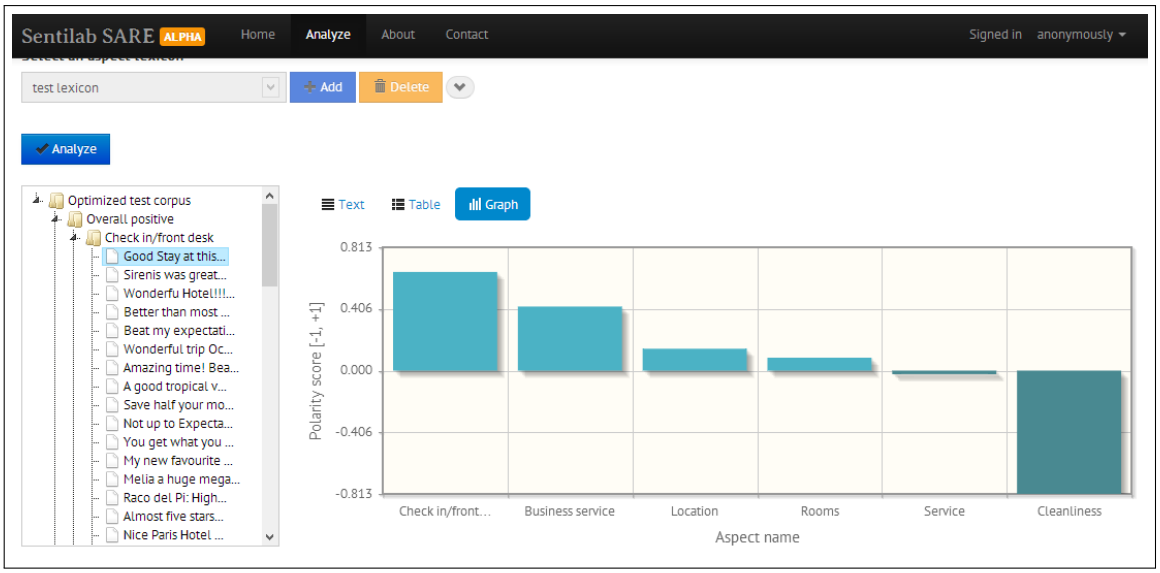

Figure 6.6: A screenshot showing partial results of the aspect-based opinion mining engine

6.2.1.1. Setup

Our opinion corpus was drawn from a set of *TripAdvisor*[1] hotel reviews as published in [50]. This dataset consists of 235,793 reviews on various hotels that were aggregated over a one month period. For our experiment, we sampled a random subset of 10,000 reviews and used review nouns as each review's elements of interest. The experiments were performed using a Java implementation, and document nouns were extracted using the Stanford PoS tagger presented in [47].

6.2.1.2. Results

The extent of data reduction achieved by each of the algorithms is given in Table 6.1. We also experimented with several values of $\hat{\tau}$ to observe the one that provides us with the best reduction while incurring the least amount of loss in utility. As shown by the plot of $\hat{\tau}$ against $\Delta(EG_{\hat{\tau}})$ in Fig. 6.9, there is a significant drop in set cover size somewhere near $\hat{\tau} = 8\%$. Thus, we can achieve very high utility coverage with a smaller part of the corpus by allowing for some outlier documents to be ignored. The comparison of algorithms shows that while the reduction achieved by the greedy algorithm is slightly higher than our algorithm (a difference of 2.81%), we can leverage our pruning technique

[1] Online travel and reviews site: `http://www.tripadvisor.com`.

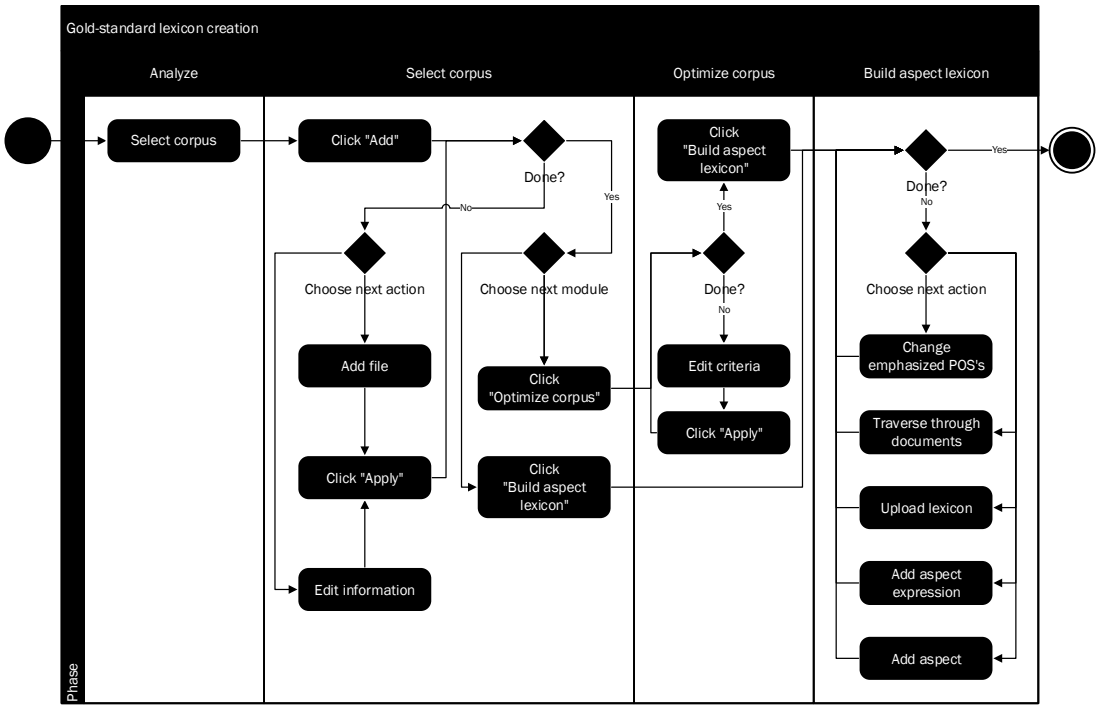

Figure 6.7: The aspect lexicon creation activity

to dramatically widen the gap in the opposite direction by introducing a small tolerance to error. The error tolerance of 8% that we have chosen for this corpus gives our algorithm an edge of 28.52% over the greedy algorithm.

Algorithm [α]	Data Reduction [$\Delta(\alpha)$]
Random	0%
Greedy	68.2%
Eagerly Greedy ($\hat{\tau}$ = 0%)	65.39%
Eagerly Greedy ($\hat{\tau}$ = 8%)	96.72%

Table 6.1: Comparison of corpus reduction algorithms

6.2.2. Aspect Expression Extraction Algorithm

We devised an experiment to to assess the performance of the semi-supervised classification algorithm presented in Sec. 5.2. The results presented in [57] for the OFESP algorithm were used as the baseline for this experiment. We could also have adopted the approach presented in [38] as the baseline, but their method uses user-provided seed

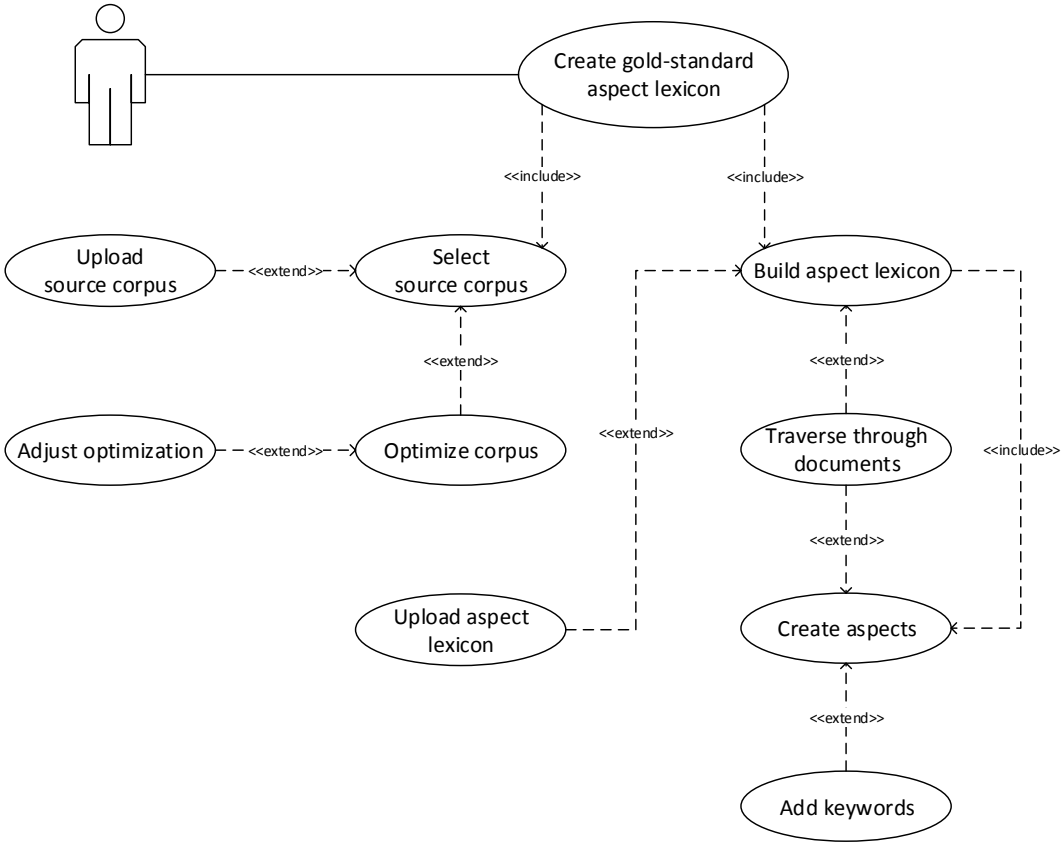

Figure 6.8: An overview of the aspect lexicon creation use case

words, which is different from our approach. We will experiment with the effects of incorporating user interaction in our future work. Algorithmic performance was measured and compared using precision, recall, and F-measure metrics.

6.2.2.1. Setup

A set of 500 hotel reviews from the TripAdvisor review corpus cited previously was used [50]. For evaluating the performance of our algorithm, we constructed a gold-standard aspect expression set with the help of the aspect lexicon builder module of SARE introduced in Sec. 5.3. The Stanford Tagger was used for PoS tagging [47], and Weka and LibSVM libraries for SVM implementation [14, 6, 8]. Word polarity and neutrality scores were adopted from the SentiWordNet polarity lexicon created by [9]. We found the most optimal values for our algorithm's parameters as well as those for the SVM learner using grid search and have reported the best results below.

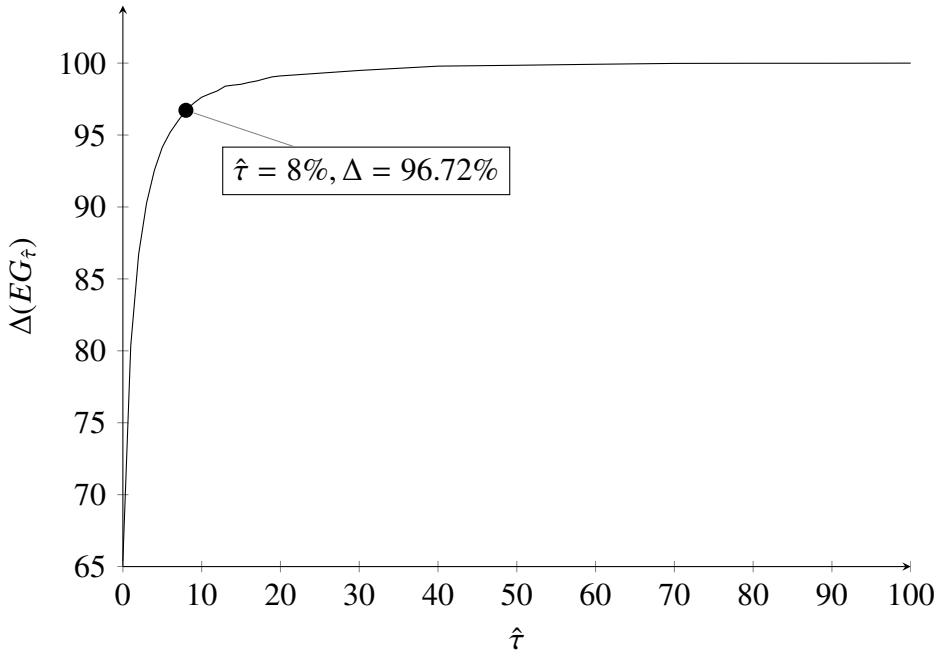

Figure 6.9: Graph showing data reduction against error tolerance

6.2.2.2. Results

A comparison of performance metrics is shown in Table 6.2. As can be seen, the performance of our approach is comparable to the baseline method. While OFESP provides a higher recall and lower precision, our approach provides a balance between precision and recall. The values are still quite low and can be further improved by introducing additional machine learning features that we have not experimented with so far. Finally, it should be noted that we did not apply the OFESP algorithm separately on our data and only used the figures provided in the source paper. Since evaluations of this kind are heavily dependent on the data itself and the gold-standard set used, a direct comparison of results in the future will provide further insight. A few examples of feature expressions that were discovered or missed are provided in Table 6.3.

Method	Precision	Recall	F-Score
OFESP	0.4413	0.7032	0.5422
Proposed	0.5164	0.5946	0.5527

Table 6.2: Performance of the aspect expression extraction algorithm as compared with the baseline

Type	Rate	Examples
True Positive	0.5164	“room service”, “furniture”, “entrance”, “hotel”, “check-in”
False Positive	0.4836	“advisor”, “friendship”, “world”, “ailment”, “occasion”
True Negative	0.8921	“paperback”, “pedestrian”, “quantity”, “intestine”, “discussion”
False Negative	0.1079	“staircase”, “bedspread”, “countertop”, “plumbing”, “bathrobe”

Table 6.3: Examples of aspect expression extracted by the algorithm

7

CONCLUSION AND FUTURE WORK

Sentiment analysis is a research area of computer science with a wide range of applications. The information produced by its processes can be used to inform everyone from policy makers to stock brokers and average consumers. The astronomical growth of the World Wide Web and more recently the explosion of web-based social media have brought about a flood of opinionated data, and with it, an increased interest in the effort to understand, manipulate, summarize, and represent this data. While the effort has sought to exploit the many opportunities presented by this immense amount of data, it has also uncovered a growing list of challenges posed by it. Sentiment analysis is thus a vast and varied field with many complex problems and tasks for which numerous techniques and solutions have been proposed and discussed in the relevant literature. Despite the breadth of research in this area, there is a lack of unified resources which provides aspiring researchers with benchmarks to compare their prospective approaches against and integrate various baseline and state-of-the-art methods into one easily-accessible system. Such a system would also be useful for industry professionals and private practitioners interested in obtaining opinion profiles of various topics and corpora.

In this thesis, we presented SARE, which is a framework designed to provide a starting point for such a unified resource. The contributions of our work are twofold, i.e.,:

1. To introduce a framework that allows for integrating various sentiment analysis tasks and methods into one system. Since sentiment analysis is a vast and growing problem and an ambitious project of this magnitude requires broader support, we proposed that the system must be incrementally extendable, publicly accessible, open-sourced, and provide mechanisms for multilingual support.

2. To exhibit the capabilities of this system by providing an integrated solution for two facets of a particular sentiment analysis problem – that of domain aspect lexicon creation and evaluation. These two facets are: gold-standard lexicon creation and

aspect expression extraction. We also integrated these solutions with a baseline aspect-based sentiment analysis engine, which was created outside of the scope of this work.

The system design is based on a layered architecture which consists of persistence, data access, logic, and web application layers. Extendability is achieved by making the system highly modular and minimizing modular inter-dependency. These modules are defined in the web application and each module is a unit of the system that adopts a particular method for solving a specific system problem. The design of the system favors a workflow-style user interaction: each module produces a result which is then passed on to one of several other modules as selected by the user until the user obtains their desired output. The environment is also designed to support data in any language as long as a NLP package for that language is integrated into the system. The individual algorithms and solutions can thus be designed in a language-independent fashion. This system has been implemented using open-source libraries and technologies, and is publicly accessible from `http://sare2.sabanciuniv.edu`. The source code has also been made available under the GPL license. We have implemented the application as a REST MVC application that allows it to be used both as a web site and a web service.

Creating gold-standard aspect lexica is a difficult problem due to the extensive amount of effort required to manually annotate large sets of documents. We approximate aspect expressions with corpus nouns and re-formulate the problem as a minimum set cover problem. This problem is then solved with an approximation algorithm termed as the Eagerly Greedy set cover approximation algorithm which also scores each document's usefulness and allows for ignoring documents with low utility while maintaining a threshold of error tolerance. The human annotator can then use this reduced set of documents to create an aspect lexicon for the corpus domain. We tested the performance of our algorithm against the classical greedy set cover approximation heuristic and a random set cover approximation approach and found that the proposed approach can significantly reduce the size of the corpus if the user is willing to tolerate even a small amount of error.

We also proposed a semi-supervised baseline method for automatically extracting aspect expressions from a corpus. Once again, we assume that all corpus nouns are candidate aspect expressions and derive machine learning features based on natural language and sentiment orientation information from all of the candidate aspect expressions. We further make the assumption that the most occurring candidates are highly likely to be aspect expressions and the least occurring candidates are highly likely not to be aspect expressions. We use this assumption to perform automatic labeling of some of the expressions and then apply the SVM learning algorithm iteratively, changing automatic labels as needed, until the results stabilize. We compared the performance of this algorithm with a baseline

method and showed that it provides a better balance of precision and recall. Moreover, unlike the baseline method, ours does not use any predefined sentiment patterns but leverages sentiment polarities to learn a classification model, which is a more generalizable approach.

7.1. Future Work

The work discussed in this thesis presents the initial design and implementation of an ambitious sentiment analysis research environment, and provides limited solutions for a major sentiment analysis problem. As such, opportunities for extensions and future work are enormous.

Sentiment analysis is a broad research area and as we have previously argued, there is need for a system that supplies baselines and implementations for state-of-the-art algorithms that can be used as benchmarks by sentiment analysis researchers and practitioners. In its current form, our platform mostly provides a framework that can be extended to support such tasks. Our first and foremost suggestion for future work is to provide implementations for prominent baseline sentiment analysis methods and pave the way for incorporating state-of-the-art techniques into the system. This will dramatically increase the usefulness of the system and its appeal to the wider sentiment analysis research community.

We have approximated aspect expressions with nouns and not dealt with noun phrases in our approach. It would be worthwhile to incorporate noun phrases into our approximation and obtain more accurate results. Some researchers have also challenged the view that aspects are represented as nouns and presented the possibility of other PoSs being used to express aspects. The presence of implicit aspects has also been studied in the literature. We have not dealt with either of these questions and left it as an opportunity for future work.

While we believe our system is the first one to support the task of extracting gold-standard aspect lexicon, there are still areas of improvement that can be made to our method and interface aside from those mentioned above. Presently, our approach takes each document as a whole and does not break it into smaller pieces. An investigation into whether smaller parts of documents such as sentences or paragraphs can be used instead of the entire document to achieve better corpus reduction and reduce the annotator's burden would be excellent grounds for future research. The gold-standard aspect lexicon is built by sequentially traversing through the entire, albeit reduced, set of documents and annotating

them individually. Another area for improvement would be to organize our system as an active learner so that it uses the already-annotated documents to decide which document to display next based on levels of uncertainty. We believe such a method would reduce the annotator's burden even further than our currently proposed approach.

Sentiment lexica are similar to aspect lexica except that they are generally more focused on adjectives and adverbs, and structured more like a dictionary and less like an ontology. Our gold-standard aspect lexicon creation tool can also be extended to create a tool for manually creating domain-based gold-standard sentiment lexica, which would be useful for performing comparisons with automatic sentiment lexicon creation algorithms.

While our aspect expressions extraction algorithm provides promising baseline results, its performance can be improved by adding more machine learning features and experimenting with other sentiment analysis lexica. An exploration into using domain-adapted sentiment lexica is especially bound to provide interesting outcomes. Several other supervised learning methods and topic modeling approaches have been reported as being successful in extracting aspect expressions, as discussed in Ch. 2. In our future studies, we are interested in investigating the impact of using such approaches in combination with ours as a means to advance the state of the art. Many methods have also been proposed in the literature to cluster aspect expressions into higher-level aspects, which we intend to utilize in order to complete the task of automatic aspect lexicon extraction.

On a more general note, we have used a particular sentiment lexicon throughout our application. This can easily be generalized by supplying the user with the option of uploading or choosing their own sentiment lexicon, which will provide more customizability for them. Furthermore, our platform only has an English-language NLP package installed at present, but we are interested in extending the linguistic reach of our application by introducing other languages such as Turkish.

While our application has the capability to interact with external web services and modules can be made, with limited effort, to interact with external processes, we have not made this functionality available to the user. One of our future goals is to allow the user to authorize any external web service to plug in to our system and interact with it just like any other part of the application. This will make the application more customizable and allow the user to harness the capabilities of our application and combine them with those supplied by external services.

As previously mentioned, SARE is an ambitious undertaking that requires broad support and contribution. We have already made our project open source, and will promote it further as an open-source project so that researchers and developers can provide their contributions and enhance the standing and usefulness of this environment. To this end,

we also need to expand the system documentation, which only covers the logic layer and some parts of the web application layer at present. More comprehensive documentation will help future collaborators to easily understand the application, perform maintenance operations, and extend its functionality with new modules and languages. Scalability is another issue that we have not thoroughly studied thus far. Currently, our application is deployed on a single server and provides reasonable performance. However, we believe that the current capacity is not enough to handle large amounts of traffic, and if the project gains more prominence and use, this capacity will need to be reconsidered and increased accordingly.

A

LIST OF SOFTWARE, TECHNOLOGIES, AND TOOLS

The following is a list of software, technologies, and tools used by our system. As mentioned in the main text, all of these are open-source software and no proprietary software has been used. It should be noted that we have left out some items such as Java, JavaScript, HTML, CSS, etc. since we believe they have a wide-enough recognition and impact not to warrant an introduction. Additionally, there are many dependencies that are included with the libraries that we have used, and such transitive dependencies are also not included in this list.

Apache Commons: A project of the Apache Software Foundation that contains general-purpose, reusable, and open-source Java components that provide quick and standard solutions to many programming problems.
URL: `http://commons.apache.org`

CoffeeScript: A programming language that provides syntactic convenience, brevity, and other utilities over JavaScript. In our application, it only exists on the server-side and compiles predictably into JavaScript when rendered as an HTTP response.
URL: `http://coffeescript.org`

Google Gson: A Java library developed by Google for converting between Java objects and JavaScript Object Notation (JSON) objects.
URL: `https://code.google.com/p/google-gson`

Google Guava: A collection of general-purpose open-source libraries for Java developed primarily by Google that provide basic utilities and convenience methods for creating and handling Java objects.
URL: `https://code.google.com/p/guava-libraries`

Hibernate: An Object-Relational Mapping (ORM) library for the Java programming language. ORMs provide mapping from object models to classical relational databases taking away the burden of database access from the developer. We have used hibernate in the data access layer of our application. Hibernate is open-source software available under the General Public License (GPL) license.
URL: `http://www.hibernate.org`

Java Persistence API (JPA): A Java framework for managing relational data. JPA provides a specification that can be harnessed by an implementation library to furnish ORM services to an application in a standardized manner. We have used JPA provided by Hibernate in the data access layer of our system.
URL: `http://goo.gl/EwUWU`

jqPlot: A plotting and charting plugin for jQuery that can be used to draw various kinds of charts in web pages.
URL: `http://www.jqplot.com`

jQuery: A JavaScript library that simplifies many common JavaScript tasks such as querying and manipulating the HyperText Markup Language (HTML) document, making Ajax calls, and handling JavaScript objects. Many open-source jQuery plugins are also available that extend the functionality of jQuery, notably jQuery UI, which provides widgets and support for several UI controls and functions.
URL: `http://jquery.com`

jsTree: A feature-rich jQuery plugin that can be used to display and manipulate tree structures in web pages.
URL: `http://www.jstree.com`

JUnit: A popular Java-based unit testing framework that allows easy creation of robust unit tests for Java code. URL: `http://junit.org`

LESS CSS: An extension of standard Cascading Style Sheets (CSS) that allows for writing more dynamic style sheets than is allowed by the original CSS specifications. In our application, it only exists on the server-side and compiles into standard CSS once rendered as an HTTP response.
URL: `http://lesscss.org`

LibSVM: An open-source and comprehensive tool for performing machine learning tasks using Support Vector Machines (SVMs). We have used a Weka wrapper for LibSVM that integrates LibSVM with Weka.
Original tool URL: `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
Weka wrapper URL: `http://weka.wikispaces.com/LibSVM`

MySQL: An open-source relational database management system available under the GPL license. The persistence layer of our environment uses MySQL.
URL: `http://www.mysql.com`

Play Framework: An open-source web application framework based on the Model-View-Controller (MVC) application pattern and supports development in both Java and Scala. Play can be used to create REpresentational State Transfer (REST) applications with limited amount of set up.
URL: `http://www.playframework.com`

Scala: An object-functional programming language that compiles into Java bytecode and can be executed on the Java Virtual Machine (JVM). This means that Scala code can utilize Java libraries and vice versa.
URL: `http://www.scala-lang.org`

Stanford Core Natural Language Processing (NLP) Library: A Java library that provides a set of NLP tools to analyze, tag, and parse English text.
URL: `http://nlp.stanford.edu/software/corenlp.shtml`

Twitter Bootstrap: A set of design templates, UI controls, and JavaScript extensions that can be used for website development.
URL: `http://getbootstrap.com`

Weka: An open-source suite of machine learning tools and algorithms that can be used both as a standalone application and from Java code.
URL: `http://www.cs.waikato.ac.nz/ml/weka`

BIBLIOGRAPHY

[1] James F. Allen. Natural language processing. In *Encyclopedia of Computer Science*, pages 1218–1222. John Wiley and Sons Ltd., Chichester, UK, 2003.

[2] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms (TALG)*, 2(2):153–177, 2006.

[3] S. Blair-Goldensohn, K. Hannan, R. McDonald, T. Neylon, G.A. Reis, and J. Reynar. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*, 2008.

[4] S. R. K. Branavan, Harr Chen, Jacob Eisenstein, and Regina Barzilay. Learning document-level semantic properties from free-text annotations. *J. Artif. Int. Res.*, 34(1):569–603, April 2009.

[5] Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 804–812, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[6] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[7] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[8] Yasser El-Manzalawy and Vasant Honavar. *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`.

[9] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422, 2006.

[10] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[11] Roy T Fielding and Richard N Taylor. Principled design of the modern web architecture. In *Proceedings of the 22nd international conference on Software engineering*, pages 407–416. ACM, 2000.

[12] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. Text mining for product attribute extraction. *SIGKDD Explor. Newsl.*, 8(1):41–48, June 2006.

[13] M. Hadano, K. Shimada, and T. Endo. Aspect identification of sentiment sentences using a clustering algorithm. *Proceedings of FIT 2010*, 2010.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[15] Vasileios Hatzivassiloglou and Kathleen R McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181. Association for Computational Linguistics, 1997.

[16] Marti Hearst. Direction-based text interpretation as an information access refinement. In Paul Jacobs, editor, *Text-Based Intelligent Systems*, pages 257–274. Lawrence Erlbaum Associates, 1992.

[17] M. Hu and B. Liu. Mining opinion features in customer reviews. In *Proceedings of the 19th national conference on Artifical intelligence*, pages 755–760. AAAI Press, 2004.

[18] Mus'ab Husaini, Andrea Ko, Dilek Tapucu, and Yücel Saygın. Ontology supported policy modeling in opinion mining process. In Pilar Herrero, Hervé Panetto, Robert Meersman, and Tharam Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, volume 7567 of *Lecture Notes in Computer Science*, pages 252–261. Springer Berlin Heidelberg, 2012.

[19] Mus'ab Husaini, Ahmet Koçyiğit, Dilek Tapucu, Berrin Yanikoglu, and Yücel Saygın. An aspect-lexicon creation and evaluation tool for sentiment analysis researchers. In PeterA. Flach, Tijl Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *Lecture Notes in Computer Science*, pages 804–807. Springer Berlin Heidelberg, 2012.

[20] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1035–1045, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[21] H. Jeong, D. Shin, and J. Choi. Ferom: Feature extraction and refinement for opinion mining. *ETRI Journal*, 33(5), 2011.

[22] W. Jin, H.H. Ho, and R.K. Srihari. Opinionminer: a novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1195–1204. ACM, 2009.

[23] Wei Jin and Hung Hay Ho. A novel lexicalized hmm-based learning framework for web opinion mining note from acm: A joint acm conference committee has been determined that the authors of this article violated acm's publication policy on simultaneous submissions. therefore acm has shut off access to this paper. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 465–472, New York, NY, USA, 2009. ACM.

[24] Aravind K Joshi. Natural language processing. *Science*, 253(5025):1242–1249, 1991.

[25] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[26] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 375–384, New York, NY, USA, 2009. ACM.

[27] Bing Liu. Sentiment analysis: a multifaceted problem. *IEEE Intelligent Systems*, 25(3):76–80, 2010.

[28] Bing Liu. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing*, 2:568, 2010.

[29] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[30] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA, 2005. ACM.

[31] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 415–463. Springer US, 2012.

[32] Chong Long, Jie Zhang, and Xiaoyan Zhut. A review selection approach for accurate feature rating estimation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 766–774. Association for Computational Linguistics, 2010.

[33] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM (JACM)*, 41(5):960–981, 1994.

[34] D.K. Ly, K. Sugiyama, Z. Lin, and M.Y. Kan. Product review summarization based on facet identification and sentence clustering. *Arxiv preprint arXiv:1110.1428*, 2011.

[35] Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 171–180, New York, NY, USA, 2007. ACM.

[36] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 935–940, New York, NY, USA, August 2006. ACM.

[37] Samaneh Moghaddam and Martin Ester. The flda model for aspect-based opinion mining: addressing the cold start problem. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 909–918, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[38] Arjun Mukherjee and Bing Liu. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 339–348, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[39] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[40] A.M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 339–346. Association for Computational Linguistics, 2005.

[41] Katharina Probst, Rayid Ghani, Marko Krema, Andy Fano, and Yan Liu. Extracting and using attribute-value pairs from product descriptions on the web. In Bettina Berendt, Andreas Hotho, Dunja Mladenic, and Giovanni Semeraro, editors, *From Web to Social Web: Discovering and Deploying User and Content Profiles*, volume 4737 of *Lecture Notes in Computer Science*, pages 41–60. Springer Berlin Heidelberg, 2007.

[42] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1199–1204, 2009.

[43] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.*, 37(1):9–27, March 2011.

[44] Warren Sack. On the computation of point of view. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 1488–1488. JOHN WILEY & SONS LTD, 1995.

[45] Christopher Scaffidi, Kevin Bierhoff, Eric Chang, Mikhael Felker, Herman Ng, and Chun Jin. Red opal: product-feature scoring from reviews. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC '07, pages 182–191, New York, NY, USA, 2007. ACM.

[46] Dilek Tapucu, Andrea Kő, Mus'ab Husaini, Ahmet Koçyiğit, Yücel Saygın, and Habin Lee. Obome-ontology based opinion mining in ubipol. In *Proceedings of European, Mediterranean & Middle Eastern Conference on Information Systems 2012*, pages 134–145, 2012.

[47] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology NAACL 03*, 1(June):173–180, 2003.

[48] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[49] Vijay V. Vazirani. *Approximation algorithms.* Springer, 2001.

[50] Hongning Wang, Yue Lu, and Chengxiang Zhai. Latent aspect rating analysis on review text data: A rating regression approach. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792, 2010.

[51] Janyce Wiebe. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740. AAAI Press, 2000.

[52] Janyce M Wiebe and William J Rapaport. A computational theory of perspective and reference in narrative. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 131–138. Association for Computational Linguistics, 1988.

[53] Yorick Wilks and Janusz Bien. Beliefs, points of view and multiple environments. In *Proc. of the international NATO symposium on Artificial and human intelligence*, pages 147–171, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[54] Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1533–1541, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[55] Bishan Yang and Claire Cardie. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1335–1345, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[56] Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1496–1505, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[57] Y. Zhai, Y. Chen, X. Hu, P. Li, and X. Wu. Extracting opinion features in sentiment patterns. In *Information Networking and Automation (ICINA), 2010 International Conference on*, volume 1, pages V1–115. IEEE, 2010.

[58] Z. Zhai, B. Liu, H. Xu, and P. Jia. Clustering product features for opinion mining. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 347–354. ACM, 2011.

[59] S. Zhang, W. Jia, Y. Xia, Y. Meng, and H. Yu. Product features extraction and categorization in chinese reviews. In *ICCGI 2011, The Sixth International Multi-Conference on Computing in the Global Information Technology*, pages 38–42, 2011.

[60] Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 56–65, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[61] Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Muhua Zhu. Multi-aspect opinion polling from textual reviews. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1799–1802, New York, NY, USA, 2009. ACM.