Università di Pisa

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di laurea specialistica in Informatica

Tesi di laurea

Multi-Facet Rating of Online Hotel Reviews: Issues, Methods and Experiments

Laureando:
Stefano Baccianella

Relatori:
Dott. Andrea Esuli
Prof. Paolo Ferragina
Dott. Fabrizio Sebastiani

Controrelatore:
Dott. Salvatore Ruggieri

Anno Accademico 2007/2008

Abstract

Online product reviews are becoming increasingly popular, and are being used more and more frequently by consumers in order to choose among competing products. Tools that rank competing products in terms of the satisfaction of consumers that have purchased the product before, are thus also becoming popular. We tackle the problem of rating (i.e., attributing a numerical score of satisfaction to) consumer reviews based on their textual content. In this work we focus on multi-facet rating of hotel reviews, i.e., on the case in which the review of a hotel must be rated several times, according to several aspects (e.g., cleanliness, dining facilities, centrality of location). We explore several aspects of the problem, including the vectorial representation of the text based on sentiment analysis, collocation analysis, and feature selection for ordinal-regression learning. We present the results of experiments conducted on a corpus of approximately 15,000 hotel reviews that we have crawled from a popular hotel review site.

Acknowledgements

This work of thesis represent the last trip of a very long journey and, like all human works, is not the work of only one person. All the people encountered during this journey gave me a little pieces of their soul, and part of this work is the sum of those small pieces.

This journey lasted 6 years and this thanks are for all the people that accompanied me along this way, some are still with me, others are flown away.

First of all i would to thank Fabrizio and Andrea, during this year of work they helped me in every moment and i am grateful to them for the chance they give me. Another thank is for Prof. Ferragina to have given us the idea for this thesis and for his strong knowledge.

I also want to thank Prof. Cappelli for introducing me to Fabrizio and Andrea.

A lovely thank is to my family, my mum, my sisters, my aunt, my uncle, my grandparents, thank to their support i could do all my stuff thinking only to my future and not to my past.

This three years in Pisa were really enjoying and i want to thank all the new and old friend i have met, a great thank to Alessandro with i shared more than a house, to Giacomo and Lorenzo with i shared more than a beer, to all my Casapiddu's cohabitants Rambo, Jiimbo, the Weird Boy and Sguatty with i shared all that i can share.

A man without past is a man without future and i never forget my old lake's friends. They're unique and i cannot forget them, we have lived more than ten years together and my life would never be the same without them.

Finally i want to thank Elena, my best friend, the only girl that never abandoned me during this journey, raising my soul when the lights goes off and always loyal to me like no one, she has all my love and my respect, a great piece of this thesis belongs to her.

Contents

1 Introduction **7**
1.1 The importance of ordinal rating for product reviews 8
1.2 Our goal 9
1.3 Outline of the thesis 10

2 Tools **12**
2.1 JaTeCS 13
2.2 libSvm 14
2.2.1 Support Vector Machines 15
2.2.2 Ordinal regression 16
2.2.3 ϵ-SVR 17
2.3 NLTK 18
2.4 General Inquirer 20
2.5 TripAdvisor 21
2.5.1 The fake reviews problem 24

3 The System **25**
3.1 Building the corpus 25
3.1.1 Crawling TripAdvisor 26
3.1.2 Filtering data 28
3.1.3 Saving data 30
3.1.4 Creating the corpus 31
3.2 Pattern extraction 33

3.2.1 POS tagging 33
3.2.2 Parsing 36
3.2.3 Collocations 41
3.3 Pattern selection 45
3.3.1 Minimum variance 46
3.3.2 Information gain for ordinal regression 48
3.3.3 Round robin 49
3.4 Sentiment analysis 50
3.5 Experimentation 53
3.5.1 Classifier 53
3.5.2 Evaluation 55
3.5.3 Baseline 56
3.5.4 Results 57

4 Related work **63**
4.1 Dave, Lawrence and Pennock 63
4.2 Pang and Lee 64
4.3 Popescu and Etzioni 65
4.4 Zhang and Varadarajan 66
4.5 Snyder and Barzilay 66
4.6 Pekar and Ou 67

5 Conclusions **69**
5.1 Future work 71

Chapter 1

Introduction

In recent years the massive use of the Web for social purposes, called *Web 2.0* or *Social Web*, has radically changed the habits of online users in many ways. In particular the Social Web makes it really easy to share our thoughts and opinions, and a growing number of users are taking advantage of this opportunity.

The availability of a large amount of opinions can affect online marketing, and recent research[1] shows that in the US 35% of internet users publish their thoughts on the Web, that about 75.2 million online users use this content to form an opinion, and this number is expected to grow to 101 millions by 2011. Searching for travel-related information is one of the most popular online activities, and travelers are expected to take advantage of the growing Web 2.0 content.

According to a study [13] performed on TripAdvisor[2], one of the most popular online review platforms for tourism-related activities, with over 10 million travel reviews and 750,000 photos all posted by travelers, travel review readers perceive reviews posted by other consumers as having several advantages over information obtained from travel service providers. Almost

[1]eMarketer, *User-Generated-Content Users Outnumber Creators*, `http://www.eMarketer.com`

[2]`http://www.tripadvisor.com`

two thirds of the readers think that reviews written by consumers contain up-to-date, enjoyable and reliable information.

This study also highlights the fact that among the users that use the TripAdvisor online booking system, 97.7% are influenced by other travelers' reviews, and among them 77.9% use the reviews as a help to choose the best place to stay. In this survey the respondents were asked to indicate how the reviews posted by other travelers influenced their travel planning. Almost all readers think that reviews are a good way to learn about travel destinations and products, help with the evaluation of alternatives and help to avoid places they would not enjoy. A clear majority of them also think that reviews increase confidence and help reduce risk by making it easier to imagine how a place will be like.

1.1 The importance of ordinal rating for product reviews

With the proliferation of tourism-related Web sites and services, along with the diffusion of social networks technology, it has become difficult to effectively locate useful information. Most of the review-based information gathered over the Web is in a simple textual form, without a very clear indication of opinion status, and is fragmented across a large number of sites and services; this makes it very difficult for travelers to form a complete opinion on a place or a hotel.

Online tourist services give a help to the readers by using a simple visualization device in order to convey an evaluation concisely: "star rating". The star rating of a hotel is a measure of the quality of a hotel as perceived by its users, where this measure ranges on an ordinal scale of values (very often this scale is from 1 to 5 "stars"), with the implied convention that a high star rating means a good evaluation by the rater (this is not to be confused with the usual indication of the quality of the hotel as decreed by the local Chamber of Commerce; a five-star hotel can get a very poor "star rating" from its customers!). The use of the star rating is very common on sites hosting online product reviews, as it enables the reader to get a quick idea of the global quality of a product.

Some sites, like TripAdvisor, give a detailed report, together with a star rating, for several different aspects of the product (e.g., in TripAdvisor different scores are attributed to different aspects of a hotel, such as cleanliness, centrality of location, and dining facilities), but in other sites the star rating only represents a global evaluation, as in epinions.com. In some other sites the star rating is instead completely absent, and a product review consists of text only.

1.2 Our goal

The difficulty to effectively locate useful information is a plague for new-generation Web services, and users often need to spend a lot of time searching for the information they really need, examining several Web sites in order to form a general idea on a given product. Tools that rank competing products in terms of the satisfaction of consumers that have purchased the product before, are thus becoming popular.

In the work we are presenting our goal has been to build a system that can collect a large number of unstructured (i.e., with no star rating attached) hotel reviews, process all the data collected, evaluate the reviews, and for each hotel predict a star rating, i.e., guess the rating the reviewer would have attributed to the hotel based on an analysis of the textual content of the review. For each hotel review not only we predict a star rating for the global quality of the hotel, but we also predict a star rating for each among a predefined set of "facets", i.e., a set of aspects for which a hotel is evaluated.

This system can potentially work as a building block for other larger systems that implement some meaningful functionality. For instance, a Web site containing product reviews whose users only seldom rate their own reviews may use our system to learn from the rated reviews and rate the others. Another Web site containing only unrated product reviews may learn from the rated reviews of some other site which contains rated reviews, and then rate its own reviews. Still another Web site could be set up that acts as a “meta” site, i.e., as an aggregator of the reviews contained in other Web sites, in the way a meta-search engine aggregates content (i.e, rankings) from other search engines; such a meta-site would need to rate al the reviews according to its own multi-point scale, irrespectively of the multi-point scales used by the “base” sites. We stress that realizing a full-fledged Web site such as the ones discussed above was not the aim of the present work.

The work presented in this thesis mostly focuses, rather than on the learning device used for generating a review rater, on the generation of the vectorial representations of the reviews that must be given as input to the learning device. These representations cannot be the usual bag-of-words representations used in classification by topic, since expressing opinions (which is the key contents of reviews) is done in natural language by means of much subtler means than captured by bag-of-words analysis. Two sentences such as “The room was clean” and “The room was not clean” would receive, after stop word removal, identical representations, while expressing opposite opinions; similar arguments can be done for comments such as “A great hotel in a horrible town” and “A horrible hotel in a great town”. We have focused on three aspects involved in the meaningful generation of vectorial representations of the reviews: (a) the extraction of complex features based on patterns of parts of speech; (b) enriching the extracted features through the use of a lexicon of opinion-carrying words; and (c) the selection of features through techniques explicitly devised for ordinal regression. Original work has been carried out especially concerning issue (c).

1.3 Outline of the thesis

In the next chapters we describe how the system has been built. In Chapter 2 we describe the tools we have used to build the system. In Chapter 3 we describe the individual components of the system, the experimental settings and the results. Chapter 4 presents works related to ours, while Chapter 5 concludes.

Chapter 2

Tools

In this chapter we describe all the tools and mathematical instruments we have used to build the system.

In Section 2.2.2 we present the rating inference problem, which is the main problem we face in our work. In fact, assigning the correct number of stars for each review is a typical rating inference problem.

The solution we propose to this problem is based on the use of supervised ordinal regression methods, implemented with the aid of the JaTeCS toolkit, described in Section 2.1.

As the learning device we have used support vector machines, explained in Section 2.2.1, built using the libsvm library, described in Section 2.2. In particular we have used the ϵ-SVR module; it has been necessary to write several classes in order to integrate this module with the JaTeCS toolkit.

For the preprocessing part, the part that requires linguistic knowledge, we needed some Natural Language Processing tools, like the part-of-speech tagger, and we used a collection of tools called Natural Language ToolKit (NLTK). This collection is described in Section 2.3.

In Section 2.4 we present the General Inquirer corpus used in the sentiment analysis of the patterns. Finally, in Section 2.5 we present a short overview of the TripAdvisor website from which we have downloaded the reviews used to build our corpus.

2.1 JaTeCS

JaTeCS[1] (Java Text Categorization System) is a Java framework that provides a comprehensive set of tools to perform automatic text categorization tasks, developed by Andrea Esuli and Tiziano Fagni at ISTI-CNR, Pisa.

The purpose of JaTeCS is to help in the execution of text categorization tasks by providing a large number of wrappers for classification libraries, and several tools to prepare data for these classifiers, like index building, feature weighting, or feature selection.

The central concept of JaTeCS is the *index*, the data structure that holds all the information about the textual data being processed.

An index is built around three entities: documents, features and categories. An index is composed by several databases, each one containing the information about a specific entity or a relation between two or more entities.

The level-1 databases, i.e., those containing information about the entities, are:

- `documents`: identifies which are the documents composing the analyzed corpus.
- `features`: identifies which are the features (e.g. words, bigrams...) which are going to represent a document;
- `categories`: identifies which are the categories according to which the documents are classified.

The level-2 databases, i.e., those containing information about the relations between pairs of entities, are:

- `content`: maps the relation between documents and features, thus allowing to determine which features appear in a document (and their frequency, weigth...), and in which documents a given feature appears;
- `classification`: maps the relation between documents and categories;

[1]`http://jatecs.isti.cnr.it/`

- `domain`: maps the relation between features and categories. This database is typically used when a *local feature selection* policy is in use.

The index itself is a level-3 database, in which the relation between all the three entities are stored (e.g., one could check if a feature appears in a document by considering the "view" from a given category, in the case of local feature selection).

JaTeCS provides a rich toolkit to ease the execution of text categorization experiments:

- a large number of *readers* for the many well known corpora and standard data formats, e.g.: Reuters21578, RCV1v-2, wipo-Alpha, $\text{SVM}_{\text{light}}$ file format, CSV file format.

- implementation of the most common text processing methods, e.g.: stopword removal and stemming for various languages (English, French, German, Italian, Spanish), term weighting (tf, tfidf, user-defined), feature selection (global, local);

- the most common learners, e.g.: naive Bayes, Rocchio, knn, AdaBoost, SVMs.

All the elements of the framework are described by interfaces so that any user can modify one or more elements (e.g. a new learning algorithm, a new storage type for databases) without requiring to modify the other elements they interact with.

2.2 libSvm

LibSvm[2] is a library for support vector machines. Its goal is to let users easily use SVMs as a tool. LibSvm uses five formulations: C-support vector classi-

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

fication (C-SVC), ν-support vector classification (ν-SVC), distribution estimation (oneclass SVM), ϵ-support vector regression (ϵ-SVR), and ν-support vector regression (ν-SVR)[2]. In our system we use the ϵ-SVR formulation to solve the regression problem. We choose to use the ϵ-SVR formulation because it solves the linear regression problem that we reduce to the ordinal regression problem.

2.2.1 Support Vector Machines

A classification task usually involves training and testing data which consist of some data instances. Each instance in the training set contains one or more "target values" (class labels) and several "attributes" (features). The goal of SVMs is to produce a model which predicts target values of data instances in the testing set for which only the attributes are given.

Given a training set of instance-label pairs $(x_i, y_i), \quad i = 1, ..., l$, where $x_i \in R_n$ and $y \in \{1, -1\}$, support vector machines (SVM) [15] require the solution of the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^{\mathrm{T}} w + \mathcal{C} \sum_{i=1}^{l} \xi_i \\ \textit{subject to} \quad & y_i(w^{\mathrm{T}} \Phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \end{aligned} \tag{2.1}$$

Here training vectors x_i are mapped into a higher-maybe infinite-dimensional space by the function Φ. Then SVMs find linear separating hyperplane that has the maximum margin in this higher-dimensional space. $\mathcal{C} > 0$ is the penalty parameter of the error term.

Furthermore, $K(x_i, x_j) \equiv \Phi(x_i)^T \Phi(x_j)$ is called the *kernel* function. Though new kernels are being proposed by researchers, there are four well known basic kernels:

- linear: $K(x_i, x_j) = x_i^{\mathrm{T}} x_j$

- polynomial: $K(x_i, x_j) = (\gamma x_i^{\mathrm{T}} x_j + r)^d, \gamma > 0$
- radial basis function (RBF): $K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2), \gamma > 0$
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^{\mathrm{T}} x_j + r)$

Here, γ, r, and d are kernel parameters.

2.2.2 Ordinal regression

The *rating inference problem* (also known as *ordinal regression*) consists in estimating a *target function* $\Phi : X \rightarrow Y$ which maps each object $x_i \in X$ into exactly one of an ordered sequence $Y = \langle y_1 \prec \ldots \prec y_n \rangle$ of *ranks* (also known as "scores", or "labels"), by means of a function $\hat{\Phi}$ called the *classifier*.

This problem is somehow intermediate between *single-label classification*, in which Y is instead an unordered set, and *metric regression*, in which Y is instead a continuous, totally ordered set (typically: the set $\mathbb{R}$ of the reals). A key feature of ordinal regression is also that the "distances" between consecutive ranks are not known, and may be different from each other; this sets ordinal regression even further apart from metric regression, in which distances between scores are exactly quantifiable.

Rating inference has recently arisen a lot of interest in information retrieval, where approaches to "learning to rank" are the instrument of choice for optimally solving many IR tasks that can be formulated as learning problems [17]; many approaches to learning to rank are indeed based on ordinal regression [3, 5, 14, 27]. More generally, rating inference is of key importance in the social sciences, since human judgments and evaluations tend to be expressed on ordinal (i.e., discrete) scales; an example of this is customer satisfaction data, where customers may evaluate a product or service as Disastrous, Poor, Fair, Good, Excellent.

2.2.3 ϵ-SVR

Given a set of objects, $\{(x_1, y_1), ..., (x_n, y_n)\}$ such that $x_i \in R^n$ is an input and $y_i \in \mathbb{R}$ is a target output, the standard form of support vector regression is:

$$\begin{aligned}
\min_{w,b,\xi,\xi^*} \quad & \frac{1}{2}w^{\mathrm{T}}w + \mathcal{C}\sum_{i=1}^{l}\xi_i + \mathcal{C}\sum_{i=1}^{l}\xi_i^* \\
\textit{subject to} \quad & w^{\mathrm{T}}\Phi(x_i) + b - y_i \leq \epsilon + \xi_i, \\
& y_i - w^{\mathrm{T}}\Phi(x_i) + b \leq \epsilon + \xi_i^*, \\
& \xi_i, \xi_i^* \geq 0 \quad i = 1, ..., l
\end{aligned} \tag{2.2}$$

The dual is:

$$\begin{aligned}
\min_{\alpha,\alpha^*} \quad & \frac{1}{2}(\alpha - \alpha^*)^{\mathrm{T}}Q(\alpha - \alpha^*) + \epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l}y_i(\alpha_i - \alpha_i^*) \\
\textit{subject to} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\
& 0 \leq \alpha_i, \alpha_i^* \leq \mathcal{C} \quad i = 1, ..., l
\end{aligned} \tag{2.3}$$

where $Q_{ij} = K(x_i, x_j) \equiv \Phi(x_i)^{\mathrm{T}}\Phi(x_j)$. The approximate function is:

$$\sum_{i=1}^{l}(-\alpha_i + \alpha_i^*)K(x_i, x) + b \tag{2.4}$$

Using this formulation we solve our ordinal regression problem with a model that solves a linear regression problem. We map ordered classes on the natural numbers scale and approximate the results to the nearest integer, i.e., if the result is 3.2 we assume that the rank chosen is 3.

2.3 NLTK

NLTK[3], the Natural Language Toolkit, is a suite of Python modules providing many NLP data types, processing tasks, corpus samples and readers. Data types include tokens, tags, chunks, trees, and feature structures. Interface definitions and reference implementations are provided for tokenizers, stemmers, taggers, chunkers, parsers (recursive-descent, shift-reduce, chart, probabilistic), clusterers, and classifiers.

Corpus samples and readers include: Brown Corpus, CoNLL-2000 Chunking Corpus, CMU Pronunciation Dictionary, NIST IEER Corpus, PP Attachment Corpus, Penn Treebank, and the SIL Shoebox corpus format [19].

NLTK provides several advantages for its use in a text processing task. The primary purpose of the toolkit is to allow to concentrate on building natural language processing systems. All the data structures and interfaces are consistent, making it easy to carry out a variety of tasks using a uniform framework. Moreover the toolkit is extensible, easily accommodating new components, whether those components replicate or extend the toolkit's existing functionality.

The toolkit is modular, so that the interaction between different components of the toolkit is minimized, and uses simple, well-defined interfaces. In particular, it is possible to complete individual projects using small parts of the toolkit, isolating them from the rest of the toolkit. The toolkit is well documented, including nomenclature, data structures, and implementations.

NLTK is organized into a collection of task-specific components. Each module is a combination of data structures for representing a particular kind of information such as trees, and implementations of standard algorithms involving those structures such as parsers. This approach is a standard feature of object-oriented design, in which components encapsulate both the resources and methods needed to accomplish a particular task.

The most fundamental NLTK components are for identifying and manip-

[3]`http://nltk.org/`

ulating individual words of text. These include: tokenizer, for breaking up strings of characters into word tokens; tokenreader, for reading tokens from different kinds of corpora; stemmer, for stripping affixes from tokens, useful in some text retrieval applications; and tagger, for adding part-of-speech tags, including regular-expression taggers, n-gram taggers and Brill taggers.

The second kind of module provided by the toolkit is designed for the creation and manipulation of structured linguistic information. These components include: *tree*, for representing and processing parse trees; *featurestructure*, for building and unifying nested feature structures (or attribute-value matrices); *cfg*, for specifying free grammars; and *parser*, for creating parse trees over input text, including chart parsers, chunk parsers and probabilistic parsers.

Several utility components are provided to facilitate processing and visualization. These include: *draw*, to visualize NLP structures and processes; *probability*, to count and collate events, and perform statistical estimation; and *corpus*, to access tagged linguistic corpora.

Finally, several advanced components are provided, mostly demonstrating NLP applications of machine learning techniques. These include: *clusterer*, for discovering groups of similar items within a large collection, including k-means and expectation maximization; *classifier*, for categorizing text into different types, including naive Bayes and maximum entropy; and HMM, for Hidden Markov Models, useful for a variety of sequence classification tasks.

A further group of components is not part of NLTK proper. These are a wide selection of third-party contributions, often developed as student projects at various institutions where NLTK is used, and distributed in a separate package called NLTK Contrib. Several of these student contributions, such as the Brill tagger and the HMM module, have now been incorporated into NLTK.

In addition to software and documentation, NLTK provides substantial corpus samples. Many of these can be accessed using the corpus module, avoiding the need to write specialized file parsing code before doing NLP tasks [20].

2.4 General Inquirer

The General Inquirer[4] is basically a mapping tool that maps each text file with counts on dictionary-supplied categories [29]. The currently distributed version combines the Harvard IV-4 dictionary content-analysis categories, the Lasswell dictionary content-analysis categories, and five categories based on the social cognition work of Semin and Fiedler, making for 182 categories in all.

Each category is a list of words and word senses. A category such as "self references" may contain only a dozen entries, mostly pronouns. Currently, the category negative is the largest with 2291 entries. Users can also add additional categories of any size.

In order to map category assignments with reasonable accuracy, the General Inquirer software spends most of its processing time identifying commonly used word senses. For example, it distinguishes between "race" as a contest, "race" as moving rapidly, "race" as a group of people of common descent, and "race" in the idiom "rat race". The General Inquirer also cautiously removes common regular suffixes so that one entry in a category can match several inflected word forms. A category entry can be an inflected word (for example, "swimming"), a root word ("swim" would match "swimming", if "swimming" is not a separate entry) or a word sense (for example, swim#1) identified by the disambiguation routines of an inflected or root word form. These English stemming procedures, integrated with English dictionaries and routines for disambiguating English word senses, limit the current General Inquirer system to English text applications.

Even though these disambiguation routines often require the General Inquirer to make several passes through a sentence, the General Inquirer is designed to process large amounts of text in a reasonable amount of time.

[4]`http://www.wjh.harvard.edu/~inquirer/`

Text files are grouped for processing into folders (compatible with input formats used by some other systems such as LIWC). The output is a matrix of "tag counts" for each category, with separate rows of counts for each file processed.

Depending on the software and computer system used, as well as the length of each file, the General Inquirer can assign counts for all 182 categories to text files at the rate of about a million words of text per hour. Some General Inquirer projects indeed have involved analyzing several million of words of text.

The main output from the Inquirer contains both raw frequency counts and indexes scaled for document length. Statistical tests then evaluate whether there are statistically reliable differences between the texts or groupings of texts being studied.

Text analysis tools span a wide range as to whether they just provide a tool for manipulating text, or whether they come with content categories and language-specific routines. Those tools that are primarily text-processing tools identify words (characters bounded by spaces or punctuation) as units and, except for problems arising from the use of special alphabets, tend to be language-independent.

The 182 General Inquirer categories were developed for social-science content-analysis research applications, not for text archiving, automatic text routing, automatic text classifying, or other natural language processing objectives, although they may be relevant to some of them.

2.5 TripAdvisor

TripAdvisor, part of Expedia Inc., operates a variety of consumer-facing user-generated content websites including bookingbuddy.com, indipendenttraveler.com, seatguru.com, smartertravel.com and of course TripAvisor.com. According to comScore Media Metrix[5], taken collectively this set of sites attract nearly

[5]`http://www.comscore.com/metrix/`

30 million monthly visitors, making it one of the most popular sources of travel information on the Web. TripAdvisor claims to have over 5,000,000 registered members and to feature over 10,000,000 user-generated reviews and opinions on over 250,000 hotels and attractions worldwide. According to Travel Weekly[6], about 8% of all leisure travellers who use the Web for travel research visit TripAdvisor. In 2007 the site was named one of the "Top 25 Travel Milestones" by USA Today. It was the only website included in the list and was cited as being instrumental in changing the way in which consumers search for travel-related information. Online shoppers who look at TripAdvisor reviews on the Hayes & Jarvis site[7] book trips at double the rate of online shoppers who have not seen the TripAdvisor reviews[8]. Hayes & Jarvis is the first tour operator to provide customers with reviews from TripAdvisor.

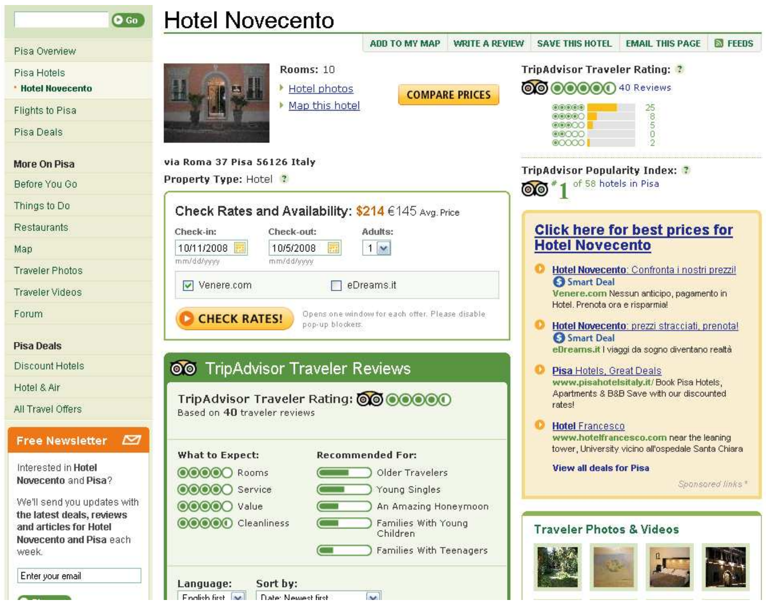
TripAdvisor, like all Web 2.0 sites, is difficult to categorize. However it is clear that its primary function is the collection of user-generated content on a highly specific domain, the travel domain [22]. Its value-adding features are its user-generated reviews and ratings. Travel consumers can go into the site and consult users' feedback on any hotel, restaurant or other travel attractions, all posted by other travelers. When adding their own reviews, users are asked to rate each experience on a five-point scale (from excellent to terrible), and to consider not only the whole experience but other aspects of the hotel like check-in, location, quality and comfort of the room, etc. Reviewers are also asked if they would recommend the hotel to their best friend, or if they were traveling alone, and whether they feel that the experience in question is suitable for different types of trips (e.g. a romantic getaway, a family trip with children, etc.). TripAdvisor also offers them the opportunity to upload photos and videos to support their review.

All data entered by users is examined by TripAdvisor to ensure that it

[6] `http://www.tripadvisor.ie/PressCenter-c2-Press_Coverage.html`

[7] `http://hayesandjarvis.co.uk/tripAdvisor`

[8] *TripAdvisor reviews double conversion rates*, `http://www.hotelmarketing.com/index.php/content/arti` `_reviews_double_conversion_rates/`



Go

Pisa Overview
Pisa Hotels
Hotel Novecento
Flights to Pisa
Pisa Deals

More On Pisa
Before You Go
Things to Do
Restaurants
Map
Traveler Photos
Traveler Videos
Forum

Pisa Deals
Discount Hotels
Hotel & Air
All Travel Offers

Free Newsletter

Interested in **Hotel Novecento** and **Pisa**?

We'll send you updates with **the latest deals, reviews and articles for Hotel Novecento and Pisa** each week.

Enter your email

Hotel Novecento

ADD TO MY MAP | WRITE A REVIEW | SAVE THIS HOTEL | EMAIL THIS PAGE | FEEDS

Rooms: 10
Hotel photos
Map this hotel

COMPARE PRICES

via Roma 37 Pisa 56126 Italy
Property Type: Hotel ?

Check Rates and Availability: $214 €145 Avg. Price

Check-in:	Check-out:	Adults:
10/11/2008 mm/dd/yyyy	10/5/2008 mm/dd/yyyy	1

[✓] Venere.com   [ ] eDreams.it

CHECK RATES! Opens one window for each offer. Please disable pop-up blockers.

TripAdvisor Traveler Reviews

TripAdvisor Traveler Rating:
Based on **40** traveler reviews

What to Expect:	Recommended For:
Rooms	Older Travelers
Service	Young Singles
Value	An Amazing Honeymoon
Cleanliness	Families With Young Children
	Families With Teenagers

Language: English first **Sort by:** Date: Newest first

TripAdvisor Traveler Rating: ?
40 Reviews

●●●●●	25
●●●●○	8
●●●○○	5
●●○○○	0
●○○○○	2

TripAdvisor Popularity Index: ?
#1 of 58 hotels in Pisa

Click here for best prices for Hotel Novecento

Hotel Novecento: Confronta i nostri prezzi!
Smart Deal
Venere.com Nessun anticipo, pagamento in Hotel. Prenota ora e risparmia!

Hotel Novecento: prezzi stracciati, prenota!
Smart Deal
eDreams.it I viaggi da sogno diventano realtà

Pisa Hotels, Great Deals
www.pisahotelsitaly.it/ Book Pisa Hotels, Apartments & B&B Save with our discounted rates!

Hotel Francesco
www.hotelfrancesco.com near the leaning tower, University vicino all'ospedale Santa Chiara

View all deals for Pisa

*Sponsored links **

Traveler Photos & Videos

Figure 2.1: An example of a hotel summary page in TripAdvisor.

conforms to content guidelines [22]. Once approved, reviews are added consecutively to each hotel's page and displayed indefinitely. The quantitative data provided by users is consolidated to generate a summary score and to rank the hotels within a destination in terms of overall popularity. Details of the algorithm used to calculate this ranking are not public knowledge, but take into consideration the quantity, quality and age of the reviews submitted to the site [22]. TripAdvisor also claims that the calculations take external data into consideration by incorporating "guidebook entries, newspaper articles and other Web content to determine traveler satisfaction". This index (also knows as the TripAdvisor Traveler Rating) is then used to determine the order in which hotels within a destination are displayed to subsequent visitors, with the most popular at the top of the list showed. Hotels have the opportunity to post a response to each review, but requests to remove or edit reviews are not allowed.

2.5.1 The fake reviews problem

The problem with authenticity is one of the key challenges faced by TripAdvisor. Several press reports (and a large amount of hotel industry) call into question the legitimacy of the reviews posted[9], TripAdvisor does little to verify that the reviewer has actually stayed in the hotel being reviewed, but assures that each review is assessed by personnel trained in fraud detection. Anyway it is widely believed that some reviews are not genuine[10], being posted in some cases by jealous competitors to decrease a hotel's rating, or in other cases by the hotel itself in an effort to improve scores. (One of TripAdvisor's competitors, SideStep.com, estimates that approximately 2% of its own reviews are not genuine)[11]. TripAdvisor attempts to minimize the problem by posting notices prominently throughout the site warning that fake reviews will not be tolerated, and that hotels attempting to manipulate the system will be penalized in their rankings and have a notice posted indicating that they post fake reviews. The "power of the crowd" that typifies Web 2.0 sites is also relevant here. As the number of reviews grows, the impact of fake reviews falls as they are overwhelmed by genuine consumer-generated content [16].

[9]Ginny McGrath and Steve Keenan, *"We're clean" pledges Tripadvisor*, `http://www.timesonline.co.uk/tol/travel/news/article1831095.ece`

[10]Christopher Elliott, *Hotel Reviews Online: In Bed With Hope, Half-Truths and Hype*, `http://www.nytimes.com/2006/02/07/business/07guides.html?_r=1&oref=slogin`

[11]Chris Reiter, *Travel Web sites clamp down on bogus reviews*, `http://www.reuters.com/article/HotelsandCasinos07/idUSN1422466620070215?pageNumber=1`

Chapter 3

The System

In this chapter we describe how we have constructed the system. In Section 3.1 we describe how we have built the dataset used for the experiments, in Section 3.2 we describe the linguistic processing to which we subject our reviews including the phases of the pattern and collocation extraction. In Section 3.3 we describe the mechanism of feature selection used to preprocess the reviews, Section 3.4 is about analyzing the reviews according to the "opinion" dimension. Finally, in Section 3.5 we present the experiments and the results obtained.

3.1 Building the corpus

The first thing we need, in order to build the system and to run experiments on it, is a large corpus of reviews, all in English and with a related ordinal rating. In order to collect a large enough set of reviews with this constraint in a fast way we choose to crawl the TripAdvisor website and save the reviews in a structured format. Since the TripAdvisor website contains far more reviews that we need or we can feasibly use for many experiments we have focused on the reviews about Pisa and Rome hotels, leaving the collection of a larger and more varied corpus to a later study.

3.1.1 Crawling TripAdvisor

The TripAdvisor website offers a variety of service and tools but does not offer a simple way to download reviews, such as a webservice or similar, so the only way to download the reviews is to crawl the website. We have implemented the crawling in three steps:

1. Search for the city and get the hotel list page;
2. From every hotel page get the overview of the hotel;
3. From the overview get the list of the reviews and process all the reviews.

The first step is implemented by sending a query to the TripAdvisor site with the address: `http://www.tripadvisor.com/HACSearch?q=city+state`. This query returns a summary page on the service available for the city requested; following the `Hotel` link the crawler gets the hotel list for the selected city.

In the second step the hotel list is scanned and hotel id, name, location, and the link to the overview are saved. The list only shows 25 hotels per page, so the crawler follows recursively the next page link until the link is available.

In the third step the crawler gets the saved list of hotel's URL and starts to scan the overview pages. From every overview it collect the overall rating, the global facet rating, the user recommendation, and the link to the full reviews.

The crawler follows the link and scans the reviews, from the review get id, title, full text of the review, pros and cons (if available) and the user rating for every facet. Like the hotel list the reviews are shown only 10 per page, and like in the hotel list the crawler follows recursively the next page link until the last page. Finally the crawled text is passed to the review filter.

Tripadvisor's pitfall

The TripAdvisor website is not an easy site to crawl, since the absence of id in the html elements forces one to visit all the DOM tree using only tree functions: we cannot use DOM functions like *getElementById* for instance, and this functions are slow and constrained to follow the order of the elements in the DOM tree.

Additionally the TripAdvisor website proposes two type of summary pages for two different types of cities: the biggest cities, like Rome, and the other cities, like Pisa. For the biggest cities a minisite is provided[1], which is completely different from the single-page summary of smaller cities, like Pisa.

The differences between the minisite and the normal hotel page, combined with the absence of id in the html elements, has required us to write several parts of the crawler in a minisite version and in a normal version.

PHP Multithreading

The crawling process required a very long time (in part because of the latency of the site, in part for the DOM functions performance). In order to speed up the crawling job, we implemented the crawler core using multi-threading. The crawler is written in PHP, and the PHP language provides a native multi-threading library: the `pcntl` extension. This extension can be used to run multiple processes in parallel; however it is only available in Linux or Unix-like operating systems.

An alternative solution consists in sending multiple HTTP requests to the same Web server on which PHP is running. Each HTTP request triggers the execution of a different task.

Multithreading support is implemented in one class, called `Thread`, that, when called, creates a closure of the target function, specified by the programmer, called when the thread starts.

When the thread is started it creates an asynchronous socket and executes a request to the server calling itself with some GET parameter indicating the function arguments. In this way the threading management is moved to the Apache server, the function is executed and the result returned to the socket.

[1]e.g., `http://rome-hotels.tripadvisor.com/`

The crawler is divided in two classes, the *master* and the *worker* crawler; in the master class the workers are called after the hotel list has been retrieved to crawl the single hotel's page.

The work balance is implemented in a daisy chain: the master checks if a worker has finished the job and, if so, the master sends a new job to it.

Using this structure with 6 workers the crawling time is decreased by 5 times.

3.1.2 Filtering data

After the crawl all the data is filtered by using a regular expression in order to delete all characters except letters, numbers and apostrophes. After this simple filter the text is passed to the language filter.

The website provide reviews in five languages: English, Italian, Spanish, French and German. For this work we decided to restrict our analysis to English, so we have implemented a "language filter" that automatically detects the language the review is written in.

Language filter identifies following the character n-gram approach by Cavnar and Trenkle [1]: for language identification one calculates the n-gram profile of a document to be identified, and compares it to the language-specific N-gram profiles: the language profile which has the smallest distance to our sample text n-gram profile identifies the language of our text. The algorithm used for filtering uses only 3-grams, instead of using general n-grams with $n = 2, 4, 5$; this difference has no impact on the precision of the filter (as stated in [1]).

The first step consists in building the 3-grams language profile. This is done in the following way:

1. Read a large number of sample texts of the target language, deleting all the punctuation marks.

2. For each document produce the character 3-grams.

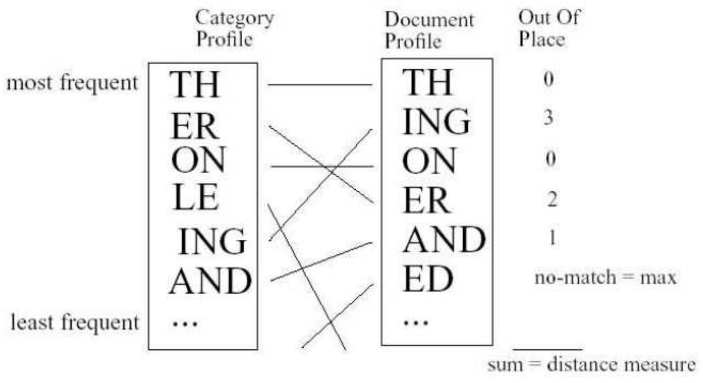


Figure 3.1: An example comparison of bigram profiles (taken from [1])

3. Store the character 3-grams in a dictionary and for each occurrence increase the counter of the 3-gram in question.

This procedure is repeated for each language. The dictionaries are sorted by frequency and constitute our 3-gram profile for each language.

In order to identify the language of a given document, we must build the 3-gram profile for that document. Then compare the document profile to the existing language profiles by calculating the distance between the 3-gram profile of the document and the profiles of the languages.

The distance for a given language is calculated in the following way:

1. Load the 3-gram profile for the language P_l and the document P_d.

2. Calculate the ranking difference of the profiles as $\sum_{i=1}^{n} |i - P_l[w_i]|$, where w_i represents the 3-gram at position i with $w_i \in P_d$, and $P_l[w_i]$ represents the position of 3-gram w_i in P_l. If the 3-gram is not present in P_l the distance is the length of the dictionary P_d.

These steps are repeated for each language. The smallest ranking difference then indicates the correct language. Tested with about 1000 reviews of the Pisa Hotels our filter achieved a precision of 100%.

The tool we built is also available online[2].

3.1.3 Saving data

After the filtering phase all the data is saved in XML format and stored in archive folder. The format reflects the structure of the TripAdvisor review page but is general enough to be extended to other review sites, for example venere.com. Figure 3.2 shows the XML format we have defined. The following is a brief explanation of the nodes of this format.

`Hotel` is the root node and it has various children:

- `id` - The absolute id assigned at the hotel;
- `name` - The name of the hotel;
- `location` - Textual description of hotel's location (e.g., Rome, Lazio, Italy);
- `georef` - With two children, Latitude and Longitude, stores the coordinate of the hotel (if available);
- `overview` - the root node of the overview subtree;
- `reviews` - the root node of the reviews subtree.

The `Overview` subtree stores the summary information about the hotel; its children are:

- `score` - The global score of the hotel (calculated by the TripAdvisor algorithm);
- `features` - Contains several `feature` nodes; every `feature` node contains the name of the facet and the global score (calculated by the TripAdvisor algorithm);

[2]`http://www.cli.di.unipi.it/~bacciane/ling`

- `users` - Contains several `user` node, every `user` node contains the user type label and the score (calculated by the TripAdvisor algorithm).

Finally the `Reviews` subtree contains several `Review` node, every node is a single review and its structure is:

- `id` - The id assigned to the review;
- `title` - Title of the review;
- `text` - The entire text of the review;
- `score` - The global score of the review (Assigned by the reviewer);
- `pros` - Text of the pros field (if present);
- `cons` - Text of the cons field (if present);
- `features` - As in the `overview`, this contains several `feature` node but here the score is assigned by the user.

3.1.4 Creating the corpus

After the saving procedure all the XML files have to be merged in order to create the corpus. The goal of the system is to predict not only the global score of a hotel but also the score of the single facets, so we have to create a corpus for each facet and one corpus for the global score.

The corpus files are in the format accepted by the JaTeCS package. Each line of the corpus file is in the format *id TAB text TAB score*; the text is the serialization of the review section, concatenating the title text, the review text, the pros and cons text in a single line.

Every review can have a maximum of seven facets scored: Business Service, Check In / Front Desk, Cleanliness, Location, Rooms, Service, Value. A review is inserted in the relative facet corpus only if the review contains

```
<?xml version='1.0'?>
<hotel>
    <id></id>
    <name></name>
    <location></location>
    <georef>
        <latitude></latitude>
        <longitude></longitude>
    </georef>
    <overview>
        <score></score>
        <features>
            <feature>
                <name></name>
                <score></score>
            </feature>
            ...
        </features>
        <users>
            <user>
                <name></name>
                <score></score>
            </user>
            ...
        </users>
    </overview>
    <reviews>
        <review>
            <id></id>
            <title></title>
            <text></text>
            <score></score>
            <pros></pros>
            <cons></cons>
            <features>
                <feature>
                    <name></name>
                    <score></score>
                </feature>
                ...
            </features>
        </review>
        ...
    </reviews>
</hotel>
```

Figure 3.2: XML Format used to represent reviews.

the score for the specific facet. As a result the numbers of reviews contained in each corpus are very different. There are a total of 15,763 reviews. Obviously the global score corpus contains all the reviews, since all the reviews have a global score; the smallest corpus is the Business Service corpus, with a total of 4,148 reviews.

Concerning the distribution of the scores, it seems that among the people who write a review for a hotel the most are satisfied consumers. This casts a shadow on the authenticity of the reviews of TripAdvisor (a topic yet discussed in Section 2.5.1), and this tends to make the classifier's task for the least represented scores difficult. On average 45% of reviews have a global score of 5 stars, 34.5% have 4 stars, 9.4% have 3 stars, 7.2% have 2 stars and only 3.9% have 1 star.

Before starting the experiment we have split the 8 corpora so that; for each corpus 75% of the reviews are in the training set, while the other 25% are in the test set. In order to create the corpus the reviews have been splitted randomly.

In the results section we present the performance of the system for every corpus, with a short discussion on how much the different composition of the corpus influences the results.

3.2 Pattern extraction

The pattern extraction is the first part of the preprocessing work and is the only part of the system that uses language-specific knowledge. The first step in the extraction is the part-of-speech (POS) tagging. After this step, using the POS information, the reviews are parsed and the patterns are extracted.

3.2.1 POS tagging

For the operation of POS tagging the system uses a cascade of four taggers, all included in the Natural Language Toolkit (NLTK).

```
^-?[0-9]+(.[0-9]+)?$    # cardinal numbers
(The|the|A|a|An|an)$    # articles
.*able$                 # adjectives
.*ness$                 # nouns formed from adjectives
.*ly$                   # adverbs
.*s$                    # plural nouns
.*ing$                  # gerunds
.*ed$                   # past tense verbs
.*                      # nouns (default)
```

Figure 3.3: Regular Expressions used by regexp tagger

The first tagger is the trigram tagger. Like all the n-gram taggers, it is trained on a large POS-tagged corpus and uses trigrams of pairs (word, tag) to assign the POS tag; if the tagger cannot assign any tag it calls the backoff tagger, in this case the bigram tagger.

Bigram and unigram taggers are like the trigram tagger but they use two, respectively, one pairs instead of three: the unigram tagger is the backoff tagger of the bigram tagger, and it assigns all the pairs not assigned by the other two. If none of them it can assign the tag it means that the pair is not in the training corpus, so the last backoff tagger, the regexp tagger, is called.

The regexp tagger is the simplest tagger, it is not trained on a tagged corpus but uses several regular expressions to guess the type of a pair, and is used at the last attempt; all the unrecognized pairs are tagged as noun. Figure 3.3 displays all the regular expression used for tagging.

Since the training phase of the n-gram taggers requires a lot of time, every tagger object has been serialized and stored in a file using Python's `pickle` library.

The Brown Corpus

The word n-gram tagger are trained on a large corpus of tagged sentences available in the NLTK; this corpus is the The Brown University Standard

Corpus of Present-Day American English (or simply *Brown Corpus*) [12].

The corpus consists of 1,014,312 words of running text of edited English prose printed in the United States during 1961. So far, as it has been possible to determine, the writers were native speakers of American English. Although all of the material first appeared in print in 1961, some of it was undoubtedly written earlier. However, no material known to be a second edition or reprint of earlier text has been included.

The Corpus is divided into 500 samples of 2000+ words each. Each sample begins at the beginning of a sentence but not necessarily of a paragraph or other larger division, and each ends at the first sentence ending after 2000 words. The samples represent a wide range of styles and varieties of prose. Verse are not included on the ground that it presents special linguistic problems different from those of prose. Drama was excluded. Fiction was included, but no samples were admitted which consisted of more than 50% dialogue.

In 1982 the corpus has been tagged (Francis and Kucera, 1982), in a semi-automatic way, creating the tagged version of the corpus (called Form C). In the tagged version of the Corpus each individual word is associated to a brief tag which assigns it to a specific word class. There are 82 of these tags of six kinds:

1. mayor form classes (parts of speech): noun, common and proper, verb, adjective, adverb, etc.

2. function words: determiners, prepositions, conjunctions, pronouns, etc.

3. certain important individual words: not, existential there, infinitival to, the forms of the verbs do, be, and have, whether auxiliaries or full verbs;

4. punctuation marks of syntactic significance;

5. inflectional morphemes, especially noun plural and possessive, verb past, present and past participle, and 3rd singular concord marker,

comparative and superlative adjective and adverb suffixes. These have the form of modifiers in a way explained later in the parsing section;

6. two tags, FM and NC, are hyphenated to the regular tags to indicate that a word is a foreign word or a cited word, respectively

Note that the corpus handles only single words, so the collocations, or poly-words (e.g., "continental breakfast"), needs a special treatment that is discussed later in the text.

3.2.2 Parsing

The second part of the preprocessing phase is the parsing phase. In this phase we use the information given by the POS tagging of the review in order to extract significant patterns from the reviews.

We observed that in hotel reviews, or product reviews in general, non-professional reviewers tends to explain their own opinion in a simple grammatical form and the aim of the parsing is to extract these opinion. We try to extract these opinions with the parser using grammatical patterns and use them as features for the classifier.

All the patterns are composed by three main parts: the noun form, the adjectival form and the verbal form.

The noun form (NN) represents the subject of the pattern and might be composed of a noun, or a saxon genitive and a noun, or an article and a noun form (e.g. "cafe", "street's cafe", "the streets's cafe").

The adjectival form (ADJ) represent the feature of the noun form in the pattern, which in our case is likely to express an opinion on the subject, this form is composed by a combination of adverb and adjective and can be joint with another adjectival form by a conjunction ("very nice room", "very little" and "nice room").

The verbal form (V) is the simplest form and represent all the verbs (including verbs "to be" and "to have"), this form is used in one pattern to join noun form with adjectival form.

Manual inspection of the text helped us in identifying three patterns that covers near all the form used in the reviews:

- Pattern A: ADJ NN
- Pattern B: NN V ADJ
- Pattern C: HV (only verb Have) [AT] (optional Article) ADJ NN

Pattern A models the simplest opinion expression on some subject. An example of pattern A are "nice room" or "very rude staff".

The pattern B models an explicit sentence and is very similar to the pattern A, except for the verb. Examples of pattern B are "the hotel was very nice and clean" or "the room is noisy".

The pattern C models a sentence that states if the subject has or not some property, this pattern can capture expressions that are not explicit opinions. Examples of this pattern are "have a nice restaurant" or "have a bar".

The parser

We write the parser used for the pattern extraction from scratch and is a procedural recursive descent predictive parser, a simple parser where every non-terminal token is a procedure that calls another procedure, if the next token is a non-terminal, or the match procedure, is the next token is a terminal.

This type of parser has been chosen for the ease of implementation, although the initial grammar has needed to be refactored in order to delete the ambiguity and the left recursion of some production.

The refactored grammar is shown in Figure 3.4. Not all the tags of the brown corpus have been considered in the grammar but, only a little subset, relevant for our field that represent the terminal token of the grammar:

- AP determiner/pronoun

```
PATTERN::= A|B|C
A::= ADJ NOUN
B::= NOUN VERB ADJ
C::= HV C'
C'::= AT ADJ NOUN | ADJ NOUN
NOUN::= NN | NN$ NN | AT NOUN
ADJ::= ADV ADJ'
ADJ'::= CONG ADV ADJ' | ADV ADJ' | ε
ADV::= RB ADV | QL ADV | JJ | AP ADV
CONG::= CC | CS
VERB::= V | B
```

Figure 3.4: Refactored grammar for the pattern parser.

- AT Article
- B Verb Be
- CC,CS Conjunctions
- JJ Adjective
- NN,NN$ Noun and Saxon genitive
- QL qualifier
- RB Adverb
- V Verb (not be or have or do)
- * Negation (also composed at the end of other tags, not in the grammar but checked at every match)

Canonical form

The three patterns extracted may state in different way the same opinion about a subject: for example the type B pattern "the room was very nice but small" and the type A pattern "very nice but small room" express the same opinion, which is also the same opinion expressed by the two separate type A patterns "very nice room" and "small room".

Starting from this point is necessary to find a way to capture this property of the patterns. The most straightforward way is to transform all the patterns in a "canonical form".

We have identified two canonical forms:

- ADJ NN (Pattern A and B)
- HV ADJ NN (Pattern C)

The transformation in canonical form is a simple procedure, every pattern can be transformed in the canonical form by applying a simple set of rules:

1. Erase the article (the hotel was very nice and good located $\mapsto$ hotel was very nice and good located)
2. Split the conjunction creating a pattern for every adjectival form (hotel was very nice and good located $\mapsto$ hotel was very nice — hotel was good located)
3. Erase the verb (hotel was very nice — hotel was good located $\mapsto$ hotel very nice — hotel good located) (Applied only on Pattern B)
4. exchange place of noun and adjectival form (hotel very nice — hotel good located $\mapsto$ very nice hotel— good located hotel (Canonical))

The use of canonical patterns decrease the sparsity of pattern in the text and help significantly the pattern selection phase increasing the frequency of meaningful patterns. Table 3.1 reports the comparison of 10 most frequent patterns in the value dataset, and clearly shows that the use of canonical form can increase the frequency of relevant patterns and reduce the sparsity of the patterns (the 100th pattern of canonical form distribution has a frequency of 114, the 100th pattern of non-canonical distribution has a frequency of 72).

Non-Canonical		Canonical	
Pattern	Frequency	Pattern	Frequency
great location	1295	great location	1693
great hotel	963	great hotel	1329
good location	477	helpful staff	1174
great place	445	friendly staff	1031
good value	366	good location	627
nice hotel	363	nice hotel	569
short walk	362	very helpful staff	522
good hotel	304	very friendly staff	491
excellent hotel	292	excellent location	491
excellent location	279	great place	476

Table 3.1: Comparison of the 10 most frequent patterns on the value dataset.

Negation

Until now the negation treatment has not been mentioned but this is a crucial point. In the tag the negation can be alone, like the word *not*, or composed with other words, like *wasn't*. In the first case the word is inserted in the pattern as is, in the second case the word is split in two words, *wasn't* becomes *was not*.

This treatment is important in the construction of the canonical form of the pattern as this ensure to have negative canonical form. (e.g. *the staff wasn't nice* is transformed in the negative canonical form *not nice staff*)

Frequency Approach		PMI Approach	
Collocation	Frequency	Collocations	PMI
great location	9461	coy maughan	-2.0
spanish steps	8552	groucho lookalike	-2.0
front desk	7755	ponder life's	-2.0
minute walk	7419	trance d'oriente	-2.0
trevi fountain	5067	willows merseyside	-2.0
termini station	4942	barbarian horde	-2.0
good location	3893	allison helfen	-2.0
train station	3792	cordero montezemolo	-2.0
hotel staff	3669	karissa mooreanniewise	-2.0
air conditioning	3454	selon personnelle	-2.0
nice hotel	3416	jacky lott	-2.0
star hotel	3393	giuliano angellino	-2.0
breakfast room	2841	homer simpson	-2.0
great place	2668	carmine ragusa	-2.0
friendly staff	2529	castroni's franchi's	-2.0
short walk	2413	selon rience	-2.0
helpful staff	2204	pianomans renditions	-2.0
double room	2123	elvira naylor	-2.0
excellent location	2031	piazzale partigiani	-2.0
continental breakfast	2017	diversionary accomplice	-2.0

Table 3.2: Comparison of the first 20 collocations identified using frequency approach and PMI approach.

3.2.3 Collocations

The last phase in the pattern extraction is a very difficult and discussed subject in computational linguistics: identifying the collocations. As pointed out earlier the POS tagger tags one word at time and does not identify, for example, a noun formed by two nouns, this kind of word are very difficult to

“Great location”!
We loved the location of this hotel **the area was great** for **affordable restaurants**, bakeries, **small grocers** and near **several good restaurants**. Do not overlook the **lovely church** next door quite a treat! The piazza **Navona was nearby** also fun to walk through. **The rooms were servicable** and some seemed to have been more recently refurbished. Just stay away from room 54 for the money it was a suite **the comfort was not worth** the price, **poor heater** and **horrible shower**, not a single shelf in the bathroom to hold a bar of soap. But 38 also a suite was much nicer. **The basic twin rooms were fine and small** as to be expected. I recommend this hotel overall but do not expect much help from the front desk as all but one of the staff bordered on surly. That was the most disappointing aspect of this **otherwise nice hotel**, **the breakfast was fine** and the breakfast **room was lovely**.

Figure 3.5: An example of a hotel review. The patterns extracted by own system are shown in boldface.

identify and have various definitions in literature.

In [21] this definition is given: A collocation is an expression consisting of two or more words that correspond to some conventional way of saying things. Or (in the words of Firth): Collocations of a given word are statements of the habitual or customary places of that word.

In this case an example of collocation is “Continental Breakfast”, but not “short walk”, collocations are characterized by a limited compositionality. We call a natural language expression compositional if the meaning of the expression can be predicted from the meaning of the parts. Collocations are not fully compositional in that there is an element of meaning added to the combination. In the case of “Continental Breakfast”, Continental acquire the mean of European to characterize the European Breakfast from the English or American Breakfast. Idioms are the most extreme example of non compositionality. Sometimes collocations meaning is very close to the compositional meaning and correctly identifying real collocations is a difficult task.

There is a considerable overlap between the concept of collocation and notions like term, technical term and terminology phrase, used in terminology extraction. When extracting collocations from a specific domain it is very useful to have a domain knowledge base to support the decision of which of the candidate are correct Unfortunately for us a hotel domain corpus is not available.

Note that collocation identification is really important for the system because pattern like “Continental Breakfast” are recognized by parser as a pattern A (in canonical form), instead of a noun form.

Frequency

The simplest method for finding collocations in a text corpus in a text corpus is surely counting the frequency. If two words occur together a lot, then that is evidence that they have a special function that is not simply explained as the function that results from their combination.

Predictably, just selecting the most frequently occurring bigrams is not very interesting because it selects all pairs of functional words (of, the, a, etc.).

There is, however, a simple heuristics that can improve the effectiveness of this approach [18]: passing the candidate phrase through a part-of-speech filter which only lets trough those patterns that are likely to be *phrases*. Justeson and Katz when talking about bigrams suggest two bigram pattern: ADJ NN and NN NN.

Table 3.2 shows the first 20 collocations identified analyzing all the datasets using the frequency method. As we can see this method is very simple but somehow accurate, most of the word are real collocation like “front desk”, “train station”, “continental breakfast”. The result of “great location” in first position is evidently a bogus, and shows the limit of this simple filtering method: frequent phrases used in a specific domain are treated like collocations. For the find experiments the list of collocations has been manually revised and cleaned from evident bogus.

Pointwise Mutual Information

Another approach at the collocation discovery task starts from a simple observation about the limited compositionality of the collocations. An information-theoretically motivated measure for discovering interesting collocations is the *pointwise mutual information* [4].

Fano [10] defines pointwise mutual information as follows:

$$\begin{aligned} I(w_1, w_2) &= \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \\ &= \log_2 \frac{P(w_1|w_2)}{P(w_1))} \\ &= \log_2 \frac{P(w_2|w_1)}{P(w_2))} \end{aligned} \tag{3.1}$$

Pointwise mutual information, $I(w_1, w_2)$, is a measure of the amount of information provided by the occurrence of the event represented by w_2 on the occurrence of the event represented by w_1. In other words the PMI is a measure of independence of the words. Table 3.2 shows the first 20 collocations identified by this approach in our reviews. The first thing we notice is the amount of proper names discovered by this method.

Unfortunately also this method has big weakness, as pointed out by many authors, the first is that this approach works only on large corpora, the second and most important is that sparseness is a particular difficult problem for mutual information.

Consider two extreme cases:

1. w_1 and w_2 always appear together (a perfect collocation)
2. w_1 and w_2 never appear together (never a collocation)

In Case 1, the PMI is:

$$I(w_1, w_2) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)} = \log_2 \frac{P(w_1)}{P(w_1)P(w_2)} = \log_2 \frac{1}{P(w_2)} < 0 \tag{3.2}$$

That is, among perfect collocation bigram, as they get rarer their PMI increases. In Case 2, we have:

$$I(w_1, w_2) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)} = \log_2 \frac{P(w_1)P(w_2)}{P(w_1)P(w_2)} = \log_2 1 = 0 \qquad (3.3)$$

Bigrams formed by lower frequency words get higher score and the extreme case 2 takes the highest value of 0.

For the final experiments we have used the frequency approach, which gave a little better performance, but in fact we can say that the two approaches gave nearly the same results and, if a larger corpus is available, is preferable to use the PMI approach for his fully-automated construction.

3.3 Pattern selection

After the pattern extraction phase we have a lot of pattern associated with the text, but the pattern are not all the same, some of them are very meaningful but others, the most of them (as stated by Zipf's rule), are only noise or do not carry any information on the opinion of the writer. In this phase the system tries to select only the most meaningful and opinion carriers patterns.

Define what is meaningful pattern is an hard challenge and we give different definition. The first definition we give is "a pattern that is representative of one category but not of another", this definition can be right, in fact is the answer given by the IG technique, but does not take in consideration that we are not trying to guess exactly the score of a review, like in a multilabel classification, but our problem is to give a score near as possible to the exact score, an ordinal regression problem. A second definition we give is "a pattern that is *locally concentrated* around a category"; "locally concentrated" means that the frequency distribution of the pattern has a very small variance.

In this section we present the 4 methods used for pattern selection: our newly proposed method, called minimum variance (MV), another version of

MV adding the round robin technique (RRMV), the information gain for ordinal regression (IGOR), and round robin on multilabel information gain (RRIG), in the result section are presented the performance achieved by each method.

3.3.1 Minimum variance

The minimum variance approach starts from the second definition we gave of meaningful pattern. A pattern that appear in only one category has a variance of zero, the smallest, while a pattern that appear in every category has the greatest variance, the number of categories divided by two. This approach allow to select also generally positive (negative) patterns, patterns that appear only in high (low) categories but not in only one, and this is the most relevant thing we are interested in.

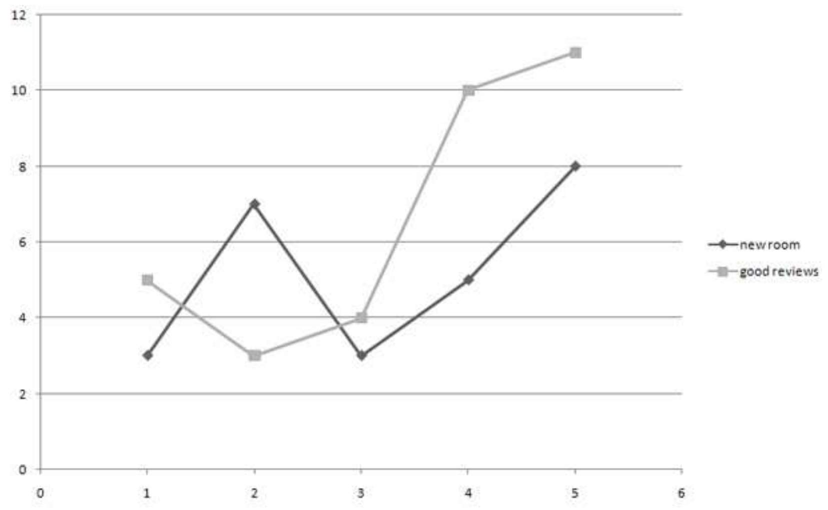


Figure 3.6: An example of the distribution of two patterns ("new room" and "good reviews") with large variance.

To better explain the idea in Figure 3.6 there is an example of two non-significant patterns, seeing their frequency distribution we note that the patterns are used both in high and low category, their variance is very large and the system will not choose this patterns. In Figure 3.7 are instead presented three patterns with a small variance, this patterns are clearly representative of a high score, and the system will choose them.

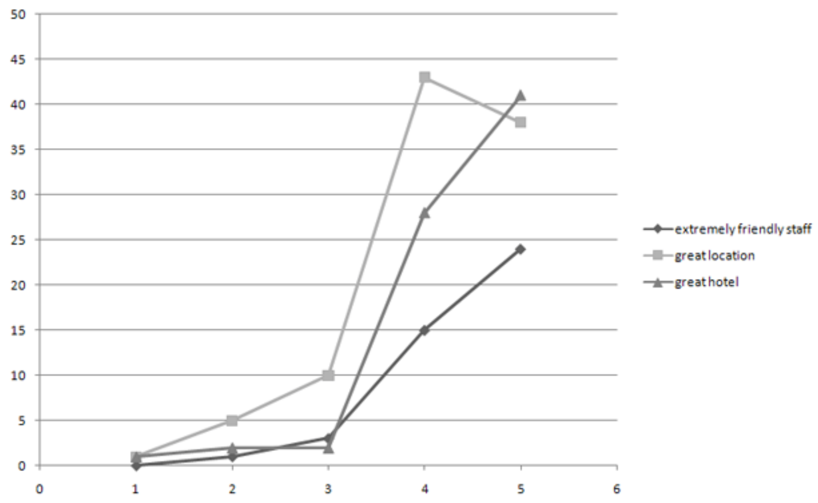


Figure 3.7: An example of the distribution of three patterns with small variance.

A small discussion must be done on the pattern absolute frequency, this approach does not take in account if a pattern is frequent or not in the text, only the patterns with one presence are ignored, and some may think that this is a weakness (and would propose to normalize the score using term frequency), this is not true, as stated in [23] “while a topic is more likely to be emphasized by frequent occurrences of certain keywords, overall sentiment may not usually be highlighted through repeated use of the same terms [...] *hapax legomena*, or words that appear a single time in a given corpus, have been found to be high-precision indicators of subjectivity”, this means that a pattern that occurs frequently in the text is no more “sentiment-significant” than another that occur few times.

3.3.2 Information gain for ordinal regression

Information gain (IG) is an information-theoretic function which measures the amount of information one random variable contains about another (or, in other words, the reduction in the uncertainty of a random variable that knowledge of the other brings about). The use of this measure in the feature selection task was firstly introduced in [30] for multi-label classification, here we present an original variant of the IG for multi-label classification problem: the IG for Ordinal Regression (IGOR).

The IG for a certain term t_k is defined as following:

$$IG(t_k) = \sum_{i} \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t,c) \log_2 \frac{P(c,t)}{P(t)P(c)} \tag{3.4}$$

In an ordinal regression task our goal, as said when talking about MV, is not to decide the exact category a document belongs but to assign a score to the document near as possible to the exact score. So we are not interested in patterns good in separating each category from the others but in patterns good in separating low categories from the high.

In order to judge if a pattern is a good separator we calculate the IG not for each category but for each interval that separe one category from another:

$$IG(t_k) = \sum_{i=1}^{n-1} \sum_{c \in \{c_{(1,i)}, c_{(i+1,n)}\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t,c) \log_2 \frac{P(c,t)}{P(t)P(c)} \tag{3.5}$$

Figure 3.8 shows an example of two patterns with an high IGOR value. As we can see the two patterns are good separators between high and low score but are not good separators for a multi-label classification task, in fact they almost cannot separate the highest and lowest categories and their IG value will be lower than the IGOR value.

Note that the patterns that take an high score with IGOR will likely have a small variance and will be chosen by the MV approach, but the two approach differ, besides how they are calculated, in how the absolute frequency is treated: the IG give in general the precedence to the terms with

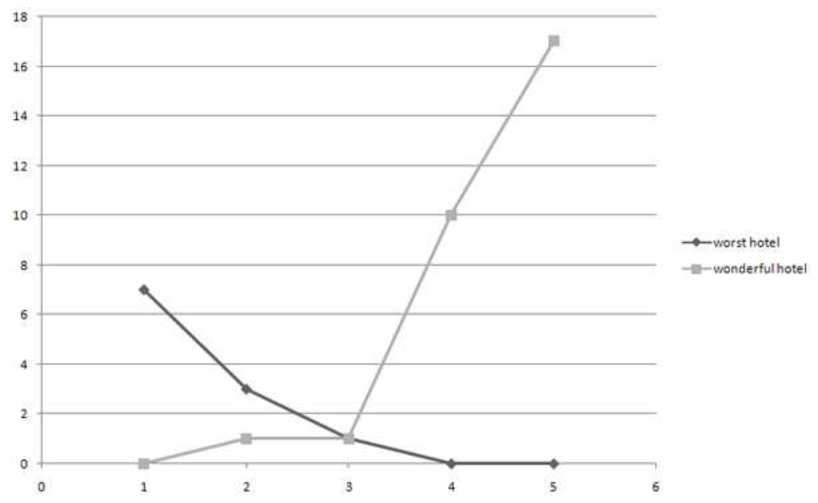


Figure 3.8: An example of the distribution of two patterns with high IGOR.

high frequency, because their joint probability is higher than the terms with low frequency.

3.3.3 Round robin

Round robin on minimum variance

Like most feature selection algorithms, the minimum variance suffer the following liability. Consider a multi-class topic recognition problem, in which one of the classes happens to contain all German texts. All the patterns extracted from German class will have a variance of zero and the system will choose all the German patterns starving the other classes. Likewise, if one class is particularly difficult all the pattern extracted will tend to have a large variance. If anything, such difficult classes need more patterns, not fewer. A solution to this problem is to perform pattern selection for each class separately via binary decompositions, and then to determine the final ranking of patterns with a round-robin algorithm in which each class gets to nominate its most desired patterns in turn [11].

In our case this selection is done assigning a category for each pattern, the category is chosen rounding the mean of the pattern frequency distribution, and for each category choose the first k pattern ordered by MV.

This variant ensure to have at least one pattern for each category but tends to penalize the two extreme category, the probability for a pattern to have a mean in the extreme category is halved.

Round robin on multi-label information gain

The information gain for ordinal regression method suffer the same liability of the minimum variance approach, unfortunately the round robin strategy is not applicable with it, to apply the round robin we need to use the multi-label IG presented before with a little variant. Instead of summing all the IG calculated for each category we take only the highest and assign to the pattern the category for round robin in which it scores higher.

Unlike the minimum variance this approach does not penalize the extreme categories but treat the problem like a multi-label classification problem not considering the different prospective of the Ordinal Regression.

The different behavior of all the methods proposed has a different impact in the pattern filtering, as highlighted in the result section.

3.4 Sentiment analysis

As the last step in the pattern extraction and filtering we have tested the use of sentiment analysis technique. After all the filtering where we have deleted all the non meaningful patterns, this time we enrich the patterns with sentiment-related information.

The goal of this last phase is to create new patterns formed not with specific adjectives but with sentiment-related tags. To do this all the selected patterns are analyzed searching on the General Inquirer (GI) corpus the sentiment associated with the words appearing in the pattern, except for nouns.

If the word is associated with a sentiment-related tag, this will be substituted by such tag in the pattern. This procedure creates new feature for the classifier more general than the original. For example, two different patterns like "beautiful hotel" and "good hotel" have the same "sentiment pattern" "positive hotel".

The GI offers a certain number of tags for the words in the corpus, other than positive and negative, some of that express the magnitude of the sentiment associated with the word, other that express the emotions and feelings.

This huge coverage of GI let us to give a better connotation to the sentiment patterns. In order to do this we have to enrich the sentiment-related added in the procedure explained above. When the filter matches a positive or negative sentiment tag, it adds also the information contained in other fields:

- *Strong* - words implying strength.
- *Power* - indicating a concern with power, control or authority.
- *Weak* - words implying weakness.
- *Submit* - connoting submission to authority or power, dependence on others, vulnerability to others, or withdrawal
- *Pleasur* - words indicating the enjoyment of a feeling, including words indicating confidence, interest and commitment.
- *Pain* - words indicating suffering, lack of confidence, or commitment
- *Feel* - words describing particular feelings, including gratitude, apathy, and optimism, not those of pain or pleasure
- *Arousal* - words indicating excitation, aside from pleasures or pains, but including arousal of affiliation and hostility
- *EMOT* - words related to emotion that are used as a disambiguation category, but also available for general use

- *Virtue* - words indicating an assessment of moral approval or good fortune, especially from the perspective of middle-class society
- *Vice* - words indicating an assessment of moral disapproval or misfortune
- *NegAff* - words of negative affect “denoting negative feelings and emotional rejection”
- *PosAff* - words of positive affect “denoting positive feelings, acceptance, appreciation and emotional support”

All those tags, if present, are added before the sentiment to create an enriched sentiment pattern. Table 3.7 reports the ten most frequent patterns, already presented in Table 3.1, with the simple sentiment filter and the enriched sentiment filter.

Pattern	Sentiment Pattern	Enriched Sentiment Pattern
great location	positive location	strong positive location
great hotel	positive hotel	strong positive hotel
helpful staff	positive staff	virtue positive staff
friendly staff	positive staff	emot virtue positive staff
good location	positive location	virtue positive location
nice hotel	positive hotel	virtue positive hotel
very helpful staff	positive staff	very virtue positive staff
very friendly staff	positive staff	very emot virtue positive staff
excellent location	positive location	virtue positive location
great place	positive place	strong positive place

Table 3.3: Comparison of first 10 most frequent patterns with the basic sentiment analysis and the enriched analysis.

A latter note must done on the word sense disambiguation, in the GI corpus an high number of words have more than one sense; filter gets the first sense and performs the analysis without disambiguating the word. The word sense disambiguation in a specific domain is an open problem and is discussed by many authors [9].

For the experiments presented in the result section we have used the enriched sentiment filter as it provides better performance than the simple.

3.5 Experimentation

After the description of the tools and the methods used in the experiments, and after presenting how the preprocessing phase works, in this last section we present the experiments done on the processed data and the ϵ-SVR Classifier used with the JATECS toolkit.

3.5.1 Classifier

The first task to complete in order to be able to run the experiments is the implementation of the svmRegression module in the JATECS toolkit. The toolkit in fact implemented the wrapper for the multi-label classification part of the libsvm package and we missing all the classes needed to use and evaluate the ordinal regression classifier contained in the libsvm package.

For our goal we need to use an ordinal regression classifier, and we selected the ϵ-SVR module.

Three type of classes have to be written, not from scratch but adapting the class from the multilabel classifier module, the libsvm wrapper in *it.cnr.jatecs.classification.svm*, SvmRegression (the learner), SvmRegressionClassifier (the wrapper to the ϵ-SVR classifier), SvmRegressionDataManager (the data manager), the classification module in *it.cnr.jatecs.classification.module*, ClassifierXofM (the class that use the svm module to do the classify), and the evaluation module *it.cnr.jatecs.evaluation*, AverageContingencyTable, AverageContingencyTableDataSet, AverageContingencyTableDataManager (implementation of the Mean Absolute Error evaluation function described some sections below).

These classes are used with the rest of the toolkit to do the experiment following these steps:

1. Read the training and validation set from file
2. Remove the stopwords
3. Create the index
4. Weight each term (see subsection below)
5. Use the svmRegression classes to create the classifier
6. Evaluate the experiment

Weighting

The JATECS toolkit already implements a well known weighting function, the *tfidf* in the form described by [7]. Many weighting methods have been developed within IR, and their variety is astounding. However there are three monotonicity assumptions that, in one form or another, appear in practically all weighting methods:

1. rare terms are no less important than frequent terms. We call this the *IDF assumption*;
2. multiple appearances of a term in a document are no less important than single appearances. We call this the *TF assumption*;
3. for the same quantity of term matching, long documents are no more important than short documents. We call this the *normalization assumption.*

These assumptions are well exemplified by the tfidf function:

$$tfidf(t_k, d_j) = tf(t_k, d_j) \log \frac{|Tr|}{\#_{Tr}(t_k)} \tag{3.6}$$

where $\#_{Tr}(t_k)$ denotes the number of documents in Tr in which tk occurs at least once and:

$$tf(t_k, d_j) = \begin{cases} 1 + \log \#(t_k, d_j) & if \#(t_k, d_j) > 0 \\ 0 & otherwise \end{cases} \tag{3.7}$$

where $\#(t_k, d_j)$ denotes the number of times t_k occurs in d_j . The $tf(t_k, d_j)$ component of equation 3.6 enforces the *tf* assumption, while the $\log \frac{|Tr|}{\#_{Tr}(t_k)}$ component of the same equation enforces the *idf* assumption. Weights obtained by equation 3.6 are usually normalized by cosine normalization, i.e.

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} tfidf(t_s, d_j)^2}} \tag{3.8}$$

which enforces the normalization assumption.

3.5.2 Evaluation

Choose an adequate evaluation function is a critical point in evaluating the experiments, in fact every task has a correct way to be evaluated. We cannot evaluate our task of ordinal regression like a multi-label classification problem, as we have said many times before in the ordinal regression problem the wrong answers are not all equal.

We are not interested in only guess the exact score of a review but, if we fail, to assign a score near as possible to the correct score. In other words we need an evaluation function that treat the errors depending on their distance from the exact score. And we have choosed a well-known measure: the Mean Absolute Error [3].

Mean Absolute Error

The mean absolute error (also called ranking loss), is defined as the average deviation of the predicted label from the true label:

$$MAE(Te) = \frac{1}{|Te|} \sum_{x_i \in Te} (\widehat{\Phi}(x_i) - \Phi(x_i)) \tag{3.9}$$

where Te denotes the test set.

The (3.9) is used to evaluate the documents and produce the score shown in the table.

3.5.3 Baseline

After defining the evaluation function we need a baseline to evaluate our experiments. We have used two method to define the baseline: the Majority Class Assignment, and the Raw Classify.

Majority Class Assignment

A well-known way to define a baseline is to create a classifier that classifies all the documents in the category that has the majority of the documents. This is a very good measure for us because our documents in the dataset are very unbalanced, as already seen talking about the dataset, and assigning all the documents to the majority class ensure to have a very low baseline.

Note that for a perfectly balanced corpus the baseline with the majority class assignment would be of 0.857.

Raw Classify

Another possible way to define the baseline is to pass the dataset to the classifier without the preprocessing phase, without any additional feature or filtering, thus using a simple bag of word representation, and only removing the stopwords and weighting the terms. This helps to evaluate if all the preprocessing work and pattern extraction bring any help to the learning process.

Table 3.4 reports the two baselines calculated with the two methods.

Dataset	Majority Class	Raw Classify
Global Score	0.657	0.621
Business Service	1.006	0.906
Check-in/Front Desk	0.780	0.808
Cleanliness	0.596	0.678
Location	0.698	0.741
Rooms	0.710	0.822
Service	0.867	0.818
Value	0.756	0.847

Table 3.4: The baseline for each corpus defined with the two method

3.5.4 Results

This last section presents the experimental results, comparing the several method described in the previous sections and highlighting the best results achieved.

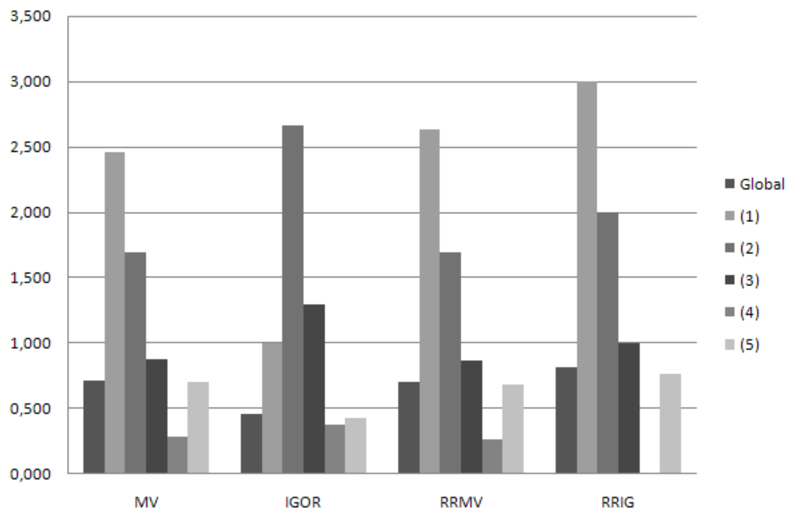


Figure 3.9: Comparison in the location dataset of the filtering method regarding the results in each category.

In the past section we have seen some methods for the pattern filtering and selection, Table 3.5 presents the global MAE achieved by each method on all the datasets.

As we can see the best method on almost all datasets is the minimum variance function, for which all the results are below the lowest baseline calculated for the datasets except for Rooms, Location and Cleanliness datasets. The results achieved for this datasets are interesting and let us to talk about global and categorial scores. The good global result achieved by IGOR on Location dataset has a clear problem, the score achieved on the single categories is unbalanced and about all the documents are classified in the first two highest categories lowering the global score but with an unacceptable categorial score.

In Figure 3.9 are graphically represented the scores of all the filtering methods on the single categories, the bad distribution of the score among the reviews make the low categories really difficult to be classified exactly and some approaches tends to filter too much patterns of the unfrequent categories obtaining exceptional scores for the frequent categories and poor scores on the others.

Dataset	Lowest Baseline	MV	IGOR	RRMV	RRIG
Global Score	0.621	0.437	0.496	0.550	0.650
Business Service	0.906	0.818	0.879	0.818	0.810
Check-in/Front Desk	0.780	0.701	0.553	0.715	0.723
Cleanliness	0.596	0.689	0.512	0.693	0.607
Location	0.698	0.715	0.464	0.703	0.821
Rooms	0.710	0.711	0.671	0.766	0.791
Service	0.818	0.757	0.565	0.747	0.759
Value	0.756	0.743	0.661	0.743	0.718

Table 3.5: Comparison of the four pattern selection methods.

So we return to the “German text problem”, explained when introducing the round robin, neither the round robin approach can afford better global results than the MV approach but achieve better results in the categorial score. This results are influenced by two main factors:

- the yet discussed unbalanced distribution of the reviews' score that make difficult the pattern extraction of the lowest classes and the relative filtering;
- the tendency of the RRMV to penalize the extreme category, in fact the 5-star reviews are the most part and if a filter penalize that category all the average result will be influenced by it.

Method	Global	1	2	3	4	5
MV	0.437	1.542	1.409	0.960	0.437	0.363
IGOR	0.496	2.200	1.545	0.719	0.342	0.503
RRMV	0.550	0.988	0.836	0.510	0.359	0.612
RRIG	0.650	1.857	1.000	0.833	0.163	0.786

Table 3.6: Comparison of categorial score in the global dataset.

Table 3.6 reports the categorial score for the global dataset. Note how in the round robin approaches the 5-star score tends to be higher than in the other approaches driving the global score to high values.

Sentiment tags

A brief discussion must done on the sentiment analysis, until now all the results presented are obtained using all the preprocessing module, but is reasonable to ask if the sentiment module brings some kind of improvement or not.

Table 3.7 compares the global error reported in the global dataset with or without the sentiment module. The improvements are not exceptional but there is some improvement, we predict that with some word sense disambiguation the results would improve significantly. Note that the sentiment

Filtering	With Sentiment	Without Sentiment
MV	0.437	0.456
IG	0.496	0.500
RRMV	0.550	0.565
RRIG	0.650	0.650

Table 3.7: Comparison of the result achieved with or without sentiment analysis on the global dataset.

module is influenced by the filtering phase, because the processed pattern by sentiment analysis are only the filtered pattern, and the improvement vary in function of this.

Collocations

Table 3.8 compares the different approaches in the collocation extraction and without any collocation. The result are mild, but while the difference between the two approach is very low, the comparison with results obtained without any extraction is significant.

Method	Global	(1)	(2)	(3)	(4)	(5)
Without Collocation	0.561	1.050	0.812	0.483	0.383	0.617
Semisupervised Frequency	0.550	0.988	0.836	0.510	0.359	0.612
Pointwise Mutual Information	0.554	1.038	0.829	0.483	0.376	0.602

Table 3.8: Comparison of the results achieved with the collocation extraction methods, the set used is the global dataset filtered with the round robin on variance approach.

Considering that the frequency approach is semiautomatic, because manually we have to clean the list from huge false collocations like “great hotel”, the PMI approach is fully automated, and is preferable.

Full Results

Table 3.9 shows all the results achieved with the experimental set-up described until now, using all the preprocessing module and the pattern selection method.

We can state that the best pattern selection method is the *round robin on minimum variance* that produces not the best global result, but minimize the average error among the rank-specific score and make more uniform the judge of system.

Also the simple *minimum variance* approach give good results and may achieve best results with a more balanced datasets. The worst approach is the *round robin on multi-label information gain* that, probably for its multi-label classification approach, obtains very bad results.

Dataset	Method	Average	1	2	3	4	5
Global	MV	0.437	1.542	1.409	0.960	0.437	0.363
	IGOR	0.496	2.200	1.545	0.719	0.342	0.503
	RRMV	0.550	0.988	0.836	0.510	0.359	0.612
	RRIG	0.650	1.857	1.000	0.833	0.163	0.786
Business Service	MV	0.818	1.958	1.179	0.642	0.491	1.063
	IGOR	0.879	2.571	1.590	0.481	0.374	1.388
	RRMV	0.818	1.958	1.179	0.642	0.491	1.063
	RRIG	0.810	2.000	1.667	0.500	0.538	1.400
Check In/Front Desk	MV	0.701	1.818	1.286	0.700	0.408	0.723
	IGOR	0.553	3.000	2.000	1.286	0.398	0.554
	RRMV	0.715	1.818	1.314	0.675	0.432	0.742
	RRIG	0.723	2.667	1.000	1.000	0.095	0.886
Cleanliness	MV	0.689	2.071	1.612	0.846	0.383	0.610
	IGOR	0.512	1.000	2.429	1.259	0.489	0.431
	RRMV	0.693	2.107	1.592	0.846	0.388	0.617
	RRIG	0.607	3.000	2.000	1.250	0.111	0.595
Location	MV	0.715	2.455	1.689	0.882	0.288	0.702
	IGOR	0.464	1.000	2.667	1.300	0.378	0.426
	RRMV	0.703	2.636	1.689	0.863	0.269	0.684
	RRIG	0.821	3.000	2.000	1.000	0.000	0.769
Rooms	MV	0.711	1.697	1.229	0.662	0.451	0.831
	IGOR	0.661	3.000	2.000	1.120	0.200	0.809
	RRMV	0.777	2.848	2.029	1.037	0.149	0.894
	RRIG	0.791	3.000	2.000	1.190	0.179	0.706
Service	MV	0.757	1.513	1.151	0.798	0.467	0.832
	IGOR	0.565	2.000	2.000	1.083	0.376	0.552
	RRMV	0.747	1.513	1.123	0.789	0.457	0.822
	RRIG	0.759	2.400	2.000	0.895	0.054	0.918
Value	MV	0.743	1.513	1.030	0.710	0.511	0.881
	IGOR	0.661	2.667	1.667	0.871	0.322	0.834
	RRMV	0.743	1.553	1.030	0.710	0.511	0.881
	RRIG	0.718	2.500	1.600	0.813	0.089	1.026

Table 3.9: Summary of the results obtained with all the datasets, complete with rank-specific scores. The last columns refer to the scores obtained for each of the five ranks, while the third column reports averages across the five scores.

Chapter 4

Related work

In this section we review related work on the analysis and rating of product reviews, focusing on the differences between these approaches and ours.

4.1 Dave, Lawrence and Pennock

The work Dave, Lawrence and Pennock is the archetype of the literature on product review analysis [6]. Like us, they build a corpus of reviews downloaded from specialized websites, where each review is labeled with a sentiment-related class (positive or negative). Based on this corpus they designed and experimented a number of methods for building product review classifiers, obtaining interesting results. They also use their classifiers to classify sentences obtained by a search engine using a product name as the search query. However, in this case, the performance of the classifiers is limited because a sentence contains much less information than an entire review.

Our work differs from theirs in several aspects. In our work we mostly focus on the text preprocessing part and feature extraction part, proposing a linguistics-intensive approach to text pattern identification, and using those patterns as features, while in their work the features are extracted by simply POS-tagging and then calculate the POS n-gram frequency. For feature selection they divided n-grams into positive and negative features, then extract the best positive and negative features using information gain. In our work we have five classes to rank instead of two, and this makes our analysis more difficult, and for the feature selection we propose four approaches to feature selection, among which one entirely new, and use sentiment-based analysis in order to create new features. Finally, their corpus consists of 6000 reviews only, and is thus much smaller than ours.

4.2 Pang and Lee

The work of Pang and Lee [24] is fairly similar in spirit to ours. In their work they face, like us, the rating inference problem, in order to determine a consumer evaluation of a product with respect to a multi-point scale. In their work they first evaluate human performance at the task through a user study. Then they use three different learning approaches: a multi-class SVM classifier, a SVM regression classifier, and a meta-algorithm, based on a metric labeling formulation of the problem, that alters a given multiclass classifier's output in an explicit attempt to ensure that similar items receive "close" labels. For the regression classifier they use, like us, a linear regression learner mapping the result onto an ordinal scale.

This work differs from ours in several aspects. The gold standard used in this work is manually generated by trained people, while our gold standard is extracted from the ratings given by the authors of the reviews. This means that the resource they work on is of a much higher quality, since some authors attribute their rating in a careless way. Also, all the focus of their work in on the learning strategy, and almost no attention is given to the preprocessing of the text (i.e., on the generation of the internal representation of the documents). Finally also in this work the corpus is much smaller than ours, since it consists of 5394 reviews, i.e., three times smaller than ours.

4.3 Popescu and Etzioni

Popescu and Etzioni [26] introduce OPINE, an unsupervised information extraction system which mines reviews in order to build a model of important product features, of their evaluation by reviewers, and of their relative quality across products. OPINE is built on top of KnowItAll [8], a Web-based, domain-independent information extraction system. Like in our system, OPINE extracts in a fully automated way the features from the reviews but, contrary to our linguistic-intensive approach, uses point wise mutual information between phrases estimated from Web search engine hit counts. After the extraction the features are analyzed to find their sentiment-based orientation, and are labeled by choosing the right label among *positive*, *negative* and *neutral*. The reviews are classified using an unsupervised classification technique called *relaxation labeling*, an iterative procedure whose output is an assignment of labels to objects. At each iteration, the algorithm uses an update equation to reestimate the probability of an object label based on its previous probability estimate and the features of its neighborhood, also considering a particular set of linguistic constraints. The final output of the system is a list product feature associated with a set of related reviews with the rating.

In this work the rating inference problem is tackled in a simplified way: while the reviews in the training set are labeled according to a five-point scale, the system described is only capable of assigning labels in the set (*positive*, *negative*, *neutral*), thus "compressing" the original rating scale to a coarser one. This is very different from what we do, since our system is capable of predicting scores on ordinal scales containing an arbitrary number of ranks.

Although in their work Popescu and Etzioni perform some kind of word sense disambiguation, they use an assumption that is not strong enough. For example, from the phrase "The hotel was clean but the rooms were extremely clean" their system will extract "(clean, hotel)" and "(extremely clean, rooms)". Obviously the adjective "clean" conveys a positive opinion, but with their disambiguation algorithm this occurrence "extremely clean" will be taken to convey a negative opinion. In fact “hotel” and “rooms” are connected with “but”, that will invert the polarity in the disambiguation routine.

Finally they measure the performance of their system using a very small corpus of 800 tuples manually annotated (with a poor inter-annotator agreement of 78%), and compare OPINE with other in-house methods without giving a state-of-the-art baseline.

4.4 Zhang and Varadarajan

Zhang and Varadarajan [31] identify a new task in product review analysis, i.e., the prediction of the utility of product reviews, which is orthogonal to polarity classification and opinion extraction. They formalize the problem in terms of linear regression. They experiment with two types of regression algorithms, the ϵ-SVR implemented in `LibSvm`, the same we used in our work, and the Simple linear regression (SLR) implemented in `WEKA`.

For the feature selection they use a set of linguistics-based approaches: Lexical Similarity Features, Shallow Syntactic Features and Lexical Subjectivity Clues. Unlike us, their concept of *good* feature is a mixture of subjective evaluation and objective information, and this requires different approach feature selection and extraction.

Finally the corpus used for their tests is comparable with ours, a total of 20,057 reviews downloaded from Amazon with three different types of keyword search.

4.5 Snyder and Barzilay

Snyder and Barzilay [28] present an algorithm that jointly learns ranking models for individual aspects of product reviews by modeling the dependencies between assigned ranks. This algorithm guides the prediction of individual rankers by analyzing meta-relations between opinions, such as agreement and contrast. Like us, the authors tried to guess not only the global score of a review, but also the score of each facet described.

Unlike us, in their work Snyder and Barzilay do not classify each feature individually but assumed that the user's opinions on one aspect can influence his/her opinions on other aspects. Their system ranks the facets individually and then uses an agreement model to correct the ranks and maximally satisfy the preferences of the individual rankers and the agreement model.

They represent each review as a vector of lexical features. The feature extraction and selection is done by extracting all word unigrams and bigrams, and discarding those that appear fewer than three times; the feature extracted are about 30,000. Unlike us they do not take into account any statistical measure of feature selection, like variance or entropy, and extract a large number of features. Finally, their corpus contains 4,488 restaurant reviews, that is, about 4 times, fewer than in our corpus.

4.6 Pekar and Ou

In their work Pekar and Ou [25] try to rank online hotel reviews in a way very close to ours. They emphasize word sense disambiguation on the specific domain of hotel reviews. Unlike us, they try to build a semantic lexicon containing terms that refer to features of a hotel that are important to the customer. In order to do this they identify and arrange the terms into semantic classes from the point of view of their importance to the customer, rather than according to an objective measure of semantic similarity. After this they manually build another lexicon of expressions conveying either negative or positive sentiment, with respect to the domain of hotel reviews.

After the construction of these two lexica, the system, as in our work, extracted a variety of lexical patterns, using linguistic cues such as the dependency relations between words as well as lexical expressions with relational semantics such as verbs and prepositions. However, unlike ours, these patterns are used not as features but in order to recognize specific relations between the object of an opinion and the opinion holder. Once a relation between words is established, the system determines topic-sentiment word pairs and, for each feature, calculates an average of the intensity scores per mention of the feature in the review, both for positive and negative sentiments. The final evaluation score for the feature is the difference between the averages of the positive and negative scores.

A weakness of this approach is the construction of the lexicon, which is done in a semi-automatic way only, and from which the feature selection depends, while in our system this entire task is done in a fully automatic way. However the major weakness of the whole work is the experimental evaluation. The authors conduct experiments on 268 reviews of hotels automatically downloaded from the epinions.com website. Besides the very low number, the reviews are highly skewed towards five-star grades (out of the 268 reviews, 240 reviews have five stars, 26 have four stars, 2 have three stars and there are no reviews with two or one star). From this small corpus they randomly selected for their experiment only 26 five-star reviews and create all the possible pairs with the 26 four-star reviews, obtaining a total of 676 pairs. The experimental tasks consists in deciding which review in each pair expressed a more positive sentiment, which is a highly artificial and applicatively uninteresting task. Furthermore, no baseline system is tested on the same corpus.

The approach followed by Pekar and Ou might be very interesting, but their experimental results lack generality and cannot establish the quality of their approach.

Chapter 5

Conclusions

In this thesis work we have faced the problem of automatic rating of product reviews, i.e., the problem of automatically summarizing into a five-point scale the evaluations expressed within textual reviews written by customers. As a type of product we have chosen hotels, but we should stress that our techniques are completely domain-independent, and could be applied to other types of products, such as, e.g., consumer electronics. In particular, we have investigated the problem of independently rating many distinct aspects ("facets") of the product, so that the same review could be given different ratings for different facets; in the case of hotels, applicable facets are "quality of service", "cleanliness", "location", etc.).

As a first activity of this thesis work we have built a dataset on which to run our experiments. We have developed a custom crawler that has allowed us to gather more than 26,000 hotel reviews from the TripAdvisor web site, all concerning hotels in Pisa and Roma. We have then developed a statistical language recognition system in order to filter out non-English reviews; this has resulted in 15,763 English reviews (after duplicate removal) that we have split into a training and a test set.

We have then implemented (also using already available open-source modules) an automatic rating system based on a supervised learning process and on the use of SVM-based ordinal regression learning algorithms. Our experimental activity has investigated the use of various methods to improve the accuracy of our system at rating reviews, and has mainly focused on the generation of the vectorial representations of the reviews to be fed to the learning system.

We have defined a baseline system that is based on generating vectorial representations by the standard "bag-of-words" technique. We have then explored methods for the generation of complex features based on the detection, within the review text, of part-of-speech patterns that we deemed relevant to identifying the opinions expressed in the text. More specifically, we have defined a set of patterns of parts of speech that we deemed semantically significant, and we have then parsed all the reviews with a part-of-speech tagger, searching the text for all possible matches of the defined patterns. All the matching expressions found in the text have been added as features to the vector space.

In order to promote the statistical robustness of the generated features we have defined a canonical form of representation for expressions matching a pattern, under which semantically equivalent expressions are grouped. This has the consequence of making the statistics more robust, and reducing the level of stochastic inter-feature dependence of the vector space.

For the extracted patterns we have also experimented the use of an opinion-related lexicon, the General Inquirer, in order to abstract away from the specific words used in the pattern and capture the general intention of the expression, so that an expression such as, e.g, "friendly staff" is turned into the feature set "positive staff", "emot positive staff" and "virtue positive staff".

Our pattern extraction system is also able to recognize the use of the most frequent forms of negation, and the use of valence shifters, such as "very", "less", "mostly", "hardly", which are all replicated into the extracted features.

In the last part of our thesis work we have focused on methods to perform effective feature selection for the ordinal regression problem we face. This work was highly original, since past research feature selection has investigated feature selection for classification and for linear regression, but not for ordinal regression.

We have compared four strategies:

- Minimum Variance;
- Information Gain for Ordinal Regression (IGOR);
- Round Robin based on Minimum Variance;
- Round Robin based on Information Gain.

The first two strategies are two original contributions of our work, and are specifically designed for ordinal regression problems. The Minimum Variance feature selection strategy is based on the intuition that a relevant feature is likely to appear frequently in documents belonging to ranks close to each other in the ordinal scale. The IGOR feature selection strategy uses an IG-based relevance measure that averages the IG of a feature on the various thresholds that separate two contiguous ranks contiguous. This is a novel use of the information gain measure, which is usually applied to the distinct categories in a one-vs-all fashion.

5.1 Future work

Rating product reviews is a fairly recent application, so a lot of research still needs to be done. In the future, we would like to work on several problems that this work has highlighted.

The first problem has to do with creating a larger and more varied dataset that can be considered representative of the many types of reviews one encounters for a given type of product. We would like to download a much larger number of reviews, about ten times the size of the current dataset. We would also like the dataset to be representative of the many types of destination which hotels cater for. The current dataset only represents towns interesting for their works of art, but other types of destination should be represented such as, e.g., seaside resorts, mountain destinations, and the like. The reason why such variety may be desirable is that different language may be used to praise a hotel in a seaside location than a hotel in a business-oriented town. Another aspect we would like to work on is multilinguality: currently, non-English reviews have been excluded from the dataset, but in the future they might be used to form a multilingual corpus, partitioned into as many sub-corpora as the languages represented inside it. For instance, this might lead to the use of techniques from cross-language information retrieval in order to profit from the existence of reviews of the same hotel in more than one language.

TripAdvisor reviews also contain some extra information that we have not used, such as for which type of customers the hotel can be recommended and for which other types it cannot. This is a dimension that we have not investigated; in the future, techniques similar to the ones we have used here might be used in order to assess, again on a multi-point scale, how much a given hotel may be recommended for a particular type of users.

Finally, it might be interesting to explore new natural language processing techniques for representing the content of the reviews in a way more effective than we have done here. To this end, the NLTK toolkit we have used here contains many more tools than we have been able to exploit here, such as natural language parsers, and other. Evaluating to what extent a better analysis of natural language can help will also be of interest.

Bibliography

[1] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, pages 161–175, Las Vegas, US, 1994.

[2] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. Unpublished manuscript, 2001.

[3] Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, pages 145–152, Bonn, DE, 2005.

[4] Kenneth Church, William Gale, Patrick Hanks, and Donald Hindle. Using statistics in lexical analysis. In Uri Zernik, editor, *Lexical Acquisition: Using On-line Resources to Build a Lexicon*, pages 115–164. Lawrence Erlbaum Associates, Hillsdale, US, 1991.

[5] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems*, volume 14, pages 641–647. The MIT Press, Cambridge, US, 2002.

[6] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International Conference on the World Wide Web (WWW'03)*, pages 519–528, Budapest, HU, 2003.

[7] Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Proceedings of the 18th ACM Symposium on Applied Computing (SAC'03)*, pages 784–788, Melbourne, US, 2003.

[8] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in KnowItAll (preliminary results). In *Proceedings of the 13th International Conference on the World Wide Web (WWW'04)*, pages 100–110, New York, US, 2004.

[9] Angela Fahrni and Manfred Klenner. Old wine or warm beer: Target-specific sentiment analysis of adjectives. In *Proceedings of the AISB 2008 Symposium on Affective Language in Human and Machine*, volume 2, pages 60–63, Aberdeen, UK, 2008.

[10] Robert M. Fano. *Transmission of information: A statistical theory of communication*. The MIT Press, Cambridge, US, 1961.

[11] George Forman. A pitfall and solution in multi-class feature selection for text classification. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, pages 38–45, Banff, CA, 2004.

[12] W. N. Francis and H. Kucera. Brown corpus manual. Unpublished manuscript, 1979.

[13] Ulrike Gretzel and Kyung Yan Yoo. Use and impact of online travel review. In *Proceedings of the 2008 International Conference on Information and Communication Technologies in Tourism*, pages 35–46, Innsbruck, AT, 2008.

[14] Edward F. Harrington. Online ranking/collaborative filtering using the perceptron algorithm. In *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 250–257, Washington, US, 2003.

[15] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Unpublished manuscript, 2008.

[16] Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th International Conference on the World Wide Web (WWW'07)*, pages 1189–1190, Banff, CA, 2007.

[17] Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai. SIGIR workshop report: Learning to rank for information retrieval (LR4IR 2007). *SIGIR Forum*, 41(2):58–62, 2008.

[18] John S. Justeson and Slava M. Katz. Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27, 1995.

[19] Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, US, 2002.

[20] Edward Loper, Steven Bird, and Ewan Klein. NLTK tutorial: Introduction to natural language processing. Unpublished manuscript, 2005.

[21] Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*, chapter 5: Collocations, pages 151–189. The MIT Press, Cambridge, US, 1999.

[22] Peter O'Connor. User-generated content and travel: A case study on tripadvisor.com. In *Proceedings of the 2008 International Conference on Information and Communication Technologies in Tourism*, pages 47–58, Innsbruck, AT, 2008.

[23] Bo Pang and Lilian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1/2):1–135, 2008.

[24] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, US, 2005.

[25] Viktor Pekar and Shiyan Ou. Discovery of subjective evaluations of product features in hotel reviews. *Journal of Vacation Marketing*, 14(2):145–156, 2008.

[26] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP'05)*, pages 339–346, Vancouver, CA, 2005.

[27] Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In *Advances in Neural Information Processing Systems*, volume 15, pages 937–944. The MIT Press, Cambridge, US, 2003.

[28] Benjamin Snyder and Regina Barzilay. Multiple aspect ranking using the good grief algorithm. In *Proceedings of the Joint Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technology Conference (NAACL/HLT'07)*, pages 300–307, Rochester, US, 2007.

[29] P. J. Stone, D. C. Dunphy, M. S. Smith, and D. M. Ogilvie. *The General Inquirer: A Computer Approach to Content Analysis.* MIT Press, Cambridge, US, 1966.

[30] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 412–420, Nashville, US, 1997.

[31] Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM'06)*, pages 51–57, Arlington, US, 2006.