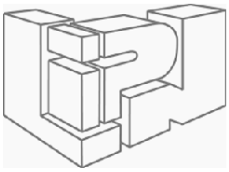
Reconnaissance d'expériences vécues dans les avis d'utilisateurs : une méthode basée sur les événements

Ehab Hassan






UNIVERSITÉ PARIS 13,
PARIS SORBONNE CITÉ

Laboratoire d'Informatique de Paris-Nord (LIPN)
Representation des Connaissances et Langage Naturel (RCLN)

THÈSE

présentée par
Ehab HASSAN

pour obtenir le grade de
DOCTEUR D'UNIVERSITÉ
SPÉCIALITÉ : INFORMATIQUE

Event-Based Recognition Of Lived Experiences In User Reviews.

Reconnaissance d'expériences vécues dans les avis d'utilisateurs : une méthode basée sur les événements.

soutenue publiquement le 03 Mai 2017
devant le jury composé de

Directeur			
Mr	Aldo GANGEMI (Pr)	-	LIPN, Université Paris 13
Co-encadrement			
Mr	Davide BUSCALDI (MCF)	-	LIPN, Université Paris 13
Rapporteurs			
Mr	Diego REFORGIATO RECUPERO (Pr)	-	Université de Cagliari
Mr	Yannick TOUSSAINT (Pr)	-	LORIA Nancy
Examinateurs			
Mme	Valentina PRESUTTI (Chercheur, HDR)	-	ISTC-CNR, Italie
Mr	Thierry CHARNOIS (Pr)	-	LIPN, Université Paris 13

Abstract

The quantity of user-generated content on the Web is constantly growing at a fast pace. A great share of this content is made of opinions and reviews on products and services. This electronic word-of-mouth is also an important factor in decisions about purchasing these products or services. Users tend to trust other users, especially if they can compare themselves to those who wrote the reviews, or, in other words, they are confident to share some characteristics. For instance, families will prefer to travel in places that have been recommended by other families. We assume that reviews that contain lived experiences are more valuable, since experiences give to the reviews a more subjective cut, allowing readers to project themselves into the context of the writer.

With this hypothesis in mind, in this thesis we aim to identify, extract, and represent reported lived experiences in customer reviews by hybridizing Knowledge Extraction and Natural Language Processing techniques in order to accelerate the decision process. For this, we define a lived user experience as an event mentioned in a review, where the author is among the participants. This definition considers that mentioned events in the text are the most important elements in lived experiences: all lived experiences are based on events, which on turn are clearly defined in time and space. Therefore, we propose an approach to extract events from user reviews, which constitute the basis of an event-based system to identify and extract lived experiences.

For the event extraction approach, we transform user reviews into their semantic representations using machine reading techniques. We perform a deep semantic parsing of reviews, detecting the linguistic frames that capture complex relations expressed in the reviews. The event-based lived experience system is carried out in three steps. The first step operates an event-based review filtering, which identifies reviews that may contain lived experiences. The second step consists of extracting relevant events together with their participants. The last step focuses on representing extracted lived experiences in each review as an event sub-graph.

In order to test our hypothesis, we carried out some experiments to verify whether lived experiences can be considered as triggers for the ratings expressed by users. Therefore, we used lived experiences as features in a classification system, comparing with the ratings of the reviews in a dataset extracted and manually annotated from Tripadvisor. The results show that lived experiences are actually correlated with the ratings.

In conclusion, this thesis provides some interesting contributions in the field of opinion mining. First of all, the successful application of machine reading to identify lived experiences. Second, the confirmation that lived experiences are correlated to ratings. Finally, the dataset produced to test our hypothesis constitutes also an important contribution of the thesis.

Keywords: Lived Experiences Extraction; Event Extraction; Machine Reading; Semantic Web; Natural Language Processing; Sentiment Analysis; User Reviews.

Résumé

La quantité de contenu généré par l'utilisateur sur le Web croît à un rythme rapide. Une grande partie de ce contenu est constituée des opinions et avis sur des produits et services. Vu leur impact, ces avis sont un facteur important dans les décisions concernant l'achat de ces produits ou services. Les utilisateurs ont tendance à faire confiance aux autres utilisateurs, surtout s'ils peuvent se comparer à ceux qui ont écrit les avis, ou, en d'autres termes, ils sont confiants de partager certaines caractéristiques. Par exemple, les familles préféreront voyager dans les endroits qui ont été recommandés par d'autres familles. Nous supposons que les avis qui contiennent des expériences vécues sont plus précieuses, puisque les expériences donnent aux avis un aspect plus subjective, permettant aux lecteurs de se projeter dans le contexte de l'écrivain.

En prenant en compte cette hypothèse, dans cette thèse, nous visons à identifier, extraire et représenter les expériences vécues rapportées dans les avis des utilisateurs en hybridant les techniques d'extraction des connaissances et de traitement du langage naturel, afin d'accélérer le processus décisionnel. Pour cela, nous avons défini opérationnellement une expérience vécue d'un utilisateur comme un événement mentionné dans un avis, où l'auteur est présent parmi les participants. Cette définition considère que les événements mentionnés dans le texte sont les éléments les plus importants dans les expériences vécues: toutes les expériences vécues sont basées sur des événements, qui sont clairement définis dans le temps et l'espace. Par conséquent, nous proposons une approche permettant d'extraire les événements à partir des avis des utilisateurs, qui constituent la base d'un système permettant d'identifier et extraire les expériences vécues.

Pour l'approche d'extraction d'événements, nous avons transformé les avis des utilisateurs en leurs représentations sémantiques en utilisant des techniques de "machine reading". Nous avons effectué une analyse sémantique profonde des avis et détecté les cadres linguistiques les plus appropriés capturant des relations complexes exprimées dans les avis. Le système d'extraction des expériences vécues repose sur trois étapes. La première étape opère un filtrage des avis, basé sur les événements, permettant d'identifier les avis qui peuvent contenir des expériences vécues. La deuxième étape consiste à extraire les événements pertinents avec leurs participants. La dernière étape consiste à représenter les expériences vécues extraites de chaque avis comme un sous-graphe d'événements contenant les événements pertinents et leurs participants.

Afin de tester notre hypothèse, nous avons effectué quelques expériences pour vérifier si les expériences vécues peuvent être considérées comme des motivations pour les notes attribuées par les utilisateurs dans le système de notation. Par conséquent, nous avons utilisé les expériences vécues comme des caractéristiques dans un système de classification, en comparant avec les notes associées avec des avis dans un ensemble de données extraites et annotées manuellement de Tripadvisor. Les résultats montrent que les expériences vécues sont corrélées avec les notes.

Cette thèse fournit des contributions intéressantes dans le domaine de l'analyse d'opinion. Tout d'abord, l'application avec succès de "machine reading" afin d'identifier les expériences vécues. Ensuite, La confirmation que les expériences vécues sont liées aux notations. Enfin, l'ensemble de données produit pour tester notre hypothèse constitue également une contribution importante de la thèse.

Mots-clés : Extraction des Expériences Vécues; Extraction d'événements; Machine Reading; Web Sémantique; Traitement du Langage Naturel; Analyse des Sentiments; Avis des Utilisateurs.

Acknowledgements

First and foremost I would like to thank my supervisors, Aldo Gangemi and Davide Buscaldi, for their great guidances and timely support during my Ph.D. study. I appreciate all their contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. Their encouragement, supervision and support enabled me to grow up as a Ph.D. for independently carrying out research. During my Ph.D. pursuit, they taught me how to do research, gave me suggestions when I met problems, and supported me to attend summer schools as well as international conferences. I benefited a lot from their profound knowledge and rigorous attitude toward scientific research.

I express my heartfelt gratitude to the reviewers of my dissertation, Prof. Diego Reforgiato Recupero and Prof. Yannick Toussaint, for accepting to review and evaluate this thesis. They provided me encouraging and constructive feedback. They took the time to very precisely read and criticize my work and I am grateful for their thoughtful and detailed comments. They greatly helped me improve the quality of this thesis. I am also very thankful for Dr. Valentina Presutti and Prof. Thierry Charnois for accepting to be the examiners of my thesis defense.

I would like to thank all members of our RCLN team and my colleagues within the LIPN laboratory for sharing the good ambiance during my stay at LIPN.

Finally, I am deeply thankful to my parents for their endless love and support, to my brothers, and their families. I wish to thank my wife Jwdi for her patience, love, and encouragement.

Thanks to all my friends and colleagues for being there with me.

Contents

1 General Introduction **3**

1.1 Motivation . . . 3

1.2 Outline . . . 4

2 Related Work **5**

2.1 Knowledge Extraction from Texts . . . 5

2.1.1 Events . . . 7

2.1.2 Event Extraction . . . 11

2.1.3 Lived Experience Extraction . . . 20

2.2 Semantic Web . . . 24

2.2.1 Semantic Web Architecture . . . 25

2.2.2 Resource Description Framework (RDF) . . . 27

2.2.3 SPARQL Query Language . . . 30

2.3 Sentiment Analysis . . . 32

2.3.1 Polarity classification . . . 32

2.3.2 Subjectivity analysis . . . 34

2.3.3 Aspect-based Sentiment Analysis (ABSA) . . . 35

2.3.4 Opinion Summarization . . . 36

2.3.5 Opinion Spam Detection . . . 38

3 FRED as an Event Extraction System **41**

3.1 Introduction . . . 41

3.2 FRED: A machine reader tool . . . 42

3.2.1 From Natural Language into DRT Languages . . . 42

3.2.2 From DRS Form to RDF/OWL ontologies . . . 43

3.2.3 Other FRED Components . . . 44

3.3 FRED as Event Extraction Tool . . . 45

3.4 FRED Quality & Importance . . . 49

3.5 Conclusion . . . 52

4 From events to Lived Experiences **55**

4.1 Introduction . . . 55

4.2 Our Approach . . . 57
4.2.1 User Review . . . 58
4.2.2 Event Extraction . . . 58
4.2.2.1 Event Negation Extraction . . . 61
4.2.2.2 Event Quality Extraction . . . 62
4.2.2.3 Event Participants Extraction . . . 63
4.2.3 Classification Methods . . . 66
4.3 Experiments . . . 68
4.3.1 DataSet . . . 69
4.3.2 Development of the Review Classifier . . . 69
4.3.3 Evaluation . . . 73
4.3.4 Error Analysis . . . 75
4.3.5 σ Estimation . . . 76
4.3.6 BaseLine . . . 76
4.3.7 Application to ESWC2014 challenge . . . 78
4.4 Conclusion . . . 79

5 Lived Experience Recognition 81
5.1 Introduction . . . 82
5.2 Defining a Lived Experience . . . 83
5.3 Extracting Lived Experiences . . . 84
5.3.1 Event Extraction . . . 84
5.3.2 Personal Events Identification . . . 87
5.3.3 Identification of Reviews containing LEs . . . 88
5.3.3.1 Annotating User Reviews . . . 89
5.3.3.2 Development of the Review Classifier . . . 89
5.3.4 Event Filtering . . . 92
5.3.5 Event participant extraction . . . 95
5.3.6 Extraction of sentences containing lived experiences . . . 98
5.3.7 Lived Experience Graph Representation . . . 99
5.4 Experimental Evaluation . . . 99
5.4.1 Dataset . . . 99
5.4.2 Review identification evaluation . . . 100
5.4.2.1 Comparison to other approaches . . . 102

5.4.2.2 σ Estimation . . . 103
5.4.3 Lived experience extraction evaluation . . . 104
5.5 LEE as a RESTful API . . . 106
5.6 Conclusion . . . 108

6 Applications 111
6.1 Correlating open rating systems and lived experiences extraction from text . 111
6.1.1 Introduction . . . 111
6.1.2 Our Approach . . . 113
6.1.2.1 Event Extraction . . . 113
6.1.2.2 Personal Events Identification . . . 114
6.1.2.3 Event Filtering . . . 114
6.1.2.4 Event Participants Extraction . . . 114
6.1.3 Experiments . . . 115
6.1.3.1 Dataset . . . 115
6.1.3.2 Review Classifier Construction . . . 116
6.1.3.3 Evaluation . . . 117
6.1.3.4 Error Analysis . . . 120
6.1.3.5 Application to ESWC2014 challenge . . . 121
6.1.3.6 σ Estimation . . . 121
6.1.4 Conclusion . . . 122
6.2 User Reviews Summarization . . . 123
6.2.1 Introduction . . . 123
6.2.2 The proposed Techniques . . . 124
6.2.2.1 Features and Opinion Extraction . . . 125
6.2.2.2 Opinion Orientation Identification . . . 127
6.2.2.3 Features Regrouping . . . 128
6.2.2.4 Summary Generation . . . 129
6.2.3 Experiments and Results . . . 130
6.2.4 Conclusions . . . 131

7 Conclusion and Perspectives 133

List of Figures

2.1 Linked Open Data 26
2.2 Semantic Web Stack 27
2.3 RDF/XML Description for Syria 28
2.4 RDF Description for Syria using N-Triple 29
2.5 RDF Description for Syria using Turtle 29
2.6 RDF Description for Syria using RDF/JSON 29
2.7 A SPARQL query structure 30
2.8 An example summary [Hu and Liu (2004)] 38

3.1 Boxer output for the sentence: *People love movies.* 43
3.2 FRED output for the sentence: *People love movies.* 44
3.3 A FRED graph depicting the core subset of triples representing event-related knowledge. 46
3.4 A diagram showing the FRED graph for the *Black Hand* sentence. 48
3.5 A summarized FRED graph showing only event relations and agentive participants for the Black Hand sentence. 50
3.6 Time to provide answers in function of the number of sentences per document as reported in [Presutti et al. (2012)]. 51
3.7 The graphical output of FRED for the sentence: *Miles Davis was an american jazz musician.*. 52

4.1 Proposed user questions when decision making process 56
4.2 Overview of the proposed approach 57
4.3 The output of FRED for the example text: *"When I asked for refund he was very rude, slamming things down on the counter and swearing. He finally refunded me after 5 minutes of arguing"* 60
4.4 68

5.1 The number of user review on TripAdvisor for the Talbott Hotel 82
5.2 Event-based Lived Experience Extraction 85
5.3 An event sub-graph representing lived experiences extracted from our first user review in Section 5.1. 100
5.4 LEE RESTful Interface 107

5.5 LEE RESTful Pipeline. 108

6.1 Overview of the proposed approach 113

6.2 Overview of the proposed approach 124

6.3 A FRED graph depicting the core subset of triples representing event-related knowledge. 126

6.4 An excerpt (80 out of 140) of the attributed features extracted from the 15 reviews concerning Affinia hotel. 130

List of Tables

2.1 Attack event template and sample extracted attributes 10

2.2 The top 15 features for each class in [Gordon (2008)] system 22

3.1 The main translation rules from DRS to OWL [Presutti et al. (2012)]. . . . 43

3.2 Boxer built-in types and relations [Presutti et al. (2012)]. 44

3.3 Summary of basic semantic tasks [Gangemi (2013)]. 50

3.4 Summary of evaluation results for basic tasks indicating accuracy values in the interval [0,1] with 1 expressing the best possible accuracy [Gangemi (2013)]. 51

3.5 The performance comparison between FRED and Semafor on Frame detection task [Presutti et al. (2012)]. 52

4.1 Characteristics of datasets used in our experiment 69

4.2 The number of extracted events from our training set for the two classes of review. 69

4.3 Most frequent event types used for classifying positive reviews 71

4.4 Most frequent event types used for classifying negative reviews 72

4.5 Some event types removed from the dictionary 72

4.6 The number of extracted events and qualities from our training set for the two classes of review. 73

4.7 The number of extracted events and event participants from our training set for the two classes of review. 73

4.8 Overall results for review classification using NB method 74

4.9 Overall results for review classification using SVM method 75

4.10 Results for the *10-fold cross* experiments for several values of σ for our training set using NB method 76

4.11 Results for the *10-fold cross* experiments for several values of σ for our training set using SVM method 76

4.12 The number of extracted verbs from our training set for the two classes of review. 77

4.13 Overall results for review classification using NB method and verb dictionary 78

4.14 Overall results for review classification using SVM method and verb dictionary 78

4.15 Results of Polarity Detection Task at ESWC2014 79

5.1 The number of extracted personal events from our training set for the two classes of review. . . . 90
5.2 Most frequent personal event types used for classifying "LivedExperience" reviews . . . 91
5.3 Most frequent personal event types used for classifying "Non-LivedExperience" reviews . . . 92
5.4 Some removed personal event types . . . 92
5.5 Results for the event filtering algorithm. . . . 95
5.6 The number of extracted personal events and their participants from our training set for the two classes of review: **features** represents personal events & personal event participants . . . 97
5.7 Characteristics of datasets used in our experiment . . . 101
5.8 Overall results for review identification using the method SVM . . . 101
5.9 Overall results for review identification using the method NB . . . 102
5.10 Overall results for lived experience identification . . . 103
5.11 Results for the *10-fold cross* experiments for several values of σ for our training set using SVM method . . . 104
5.12 Results for the *10-fold cross* experiments for several values of σ for our training set using NB method . . . 104
5.13 Results of human annotation for lived experience extraction . . . 105
5.14 The Kappa interpretation according to [Landis and Koch (1977)] . . . 105
5.15 The results of lived experience extraction task. . . . 106

6.1 Characteristics of datasets used in our experiment . . . 115
6.2 The number of extracted lived experience events from the two classes of review. . . . 116
6.3 Overall results for review classification using NB method with four configurations. . . . 118
6.4 Overall results for review classification using SVM method with four configurations. . . . 118
6.5 Most frequent features used for classifying positive reviews . . . 119
6.6 Most frequent features used for classifying negative reviews . . . 119
6.7 Some common features removed from the dictionary . . . 120
6.8 Overall results for review classification using all event features, personal and general, for NB method . . . 120
6.9 Results of Polarity Detection Task at ESWC2014 . . . 121

6.10 Results for the *10-fold cross* experiments for several values of σ for our training set 122

6.11 The proposed rules to obtain the final polarity of an opinion word. 128

6.12 The result of the summarization of the 15 reviews for Affinia hotel. 131

List of publications

International conferences with reviewing committee (4 papers)

- Ehab Hassan, Davide Buscaldi, and Aldo Gangemi. Event-based recognition of lived experiences in user reviews. In Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20, pages 320-336. Springer, 2016.
- Ehab HASSAN, Davide BUSCALDI, and Aldo GANGEMI. Correlating open rating systems and lived experiences extraction from text. In the 12th International Conference on Semantic Systems, SEMANTiCS 2016, Leipzig, Germany, September 12-15, 2016.
- Ehab Hassan, Davide Buscaldi, and Aldo Gangemi. Correlating open rating systems and event extraction from text. In the 22nd International Conference on Neural Information Processing, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, pages 367-375. Springer, 2015.
- Aldo Gangemi, Ehab Hassan, Valentina Presutti, and Diego Reforgiato. Fred as an event extraction tool. In Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2013) at 12th International Semantic Web Conference, ISWC 2013, Zurich, Switzerland, page 14, 2013.

French speaking conferences with reviewing committee (1 paper)

- Ehab Hassan, Davide Buscaldi, and Aldo Gangemi. Machine reading for abstractive summarization of customer reviews in the touristic domain. Septième Atelier Recherche d'Information SEmantique RISE, Rennes 30 juin 2015, page 6.

To my parents, my wife and my brothers...

Chapter 1

General Introduction

Contents

1.1 Motivation 3
1.2 Outline 4

1.1 Motivation

In the last years, the Web has significantly changed how people express themselves and interact with others. People can now share their reviews of products and services on commercial websites and express their opinions and interact with others through platforms such as blogs, social networks, and discussion forums.

All this user-generated content constitutes a valuable source of information in the context of various fields (e-commerce, politics, finance, etc.). In the case of e-commerce, online reviews constitute the backbone of electronic word-of-mouth communication (eWOM) and they are often the primary factor in a user's decision to purchase a product or service [Hennig-Thurau et al. (2004)]. As other types of content on the Internet, user-generated content is also affected by the problem of information overload: the number of sites allowing users to share their opinions is continuously growing, as the number of users. The consequence is that the problem of making an informed decision about choosing a certain product or service becomes more and more difficult and time-consuming.

In certain cases, users share not only an opinion but also their experiences. This is particularly true in the case of travel and catering, where the opinion is often motivated by some "lived experience" when staying at some hotel and/or eating in some restaurant. Sites like TripAdvisor[1], Booking[2] and AirBnB[3] incentivise users to share their experiences, both good and bad ones. These reviews often contain non-fictional narrative or stories that people tell about their experience, that allow other users to project themselves as potential customers, comparing their desires and expectations to those of others in a similar context. Therefore, lived experiences can give them specific and more interesting information than general opinions, and provide a larger palette of perspectives than traditional sentiment analysis, since experiences differ among users and hint at their own preferences and reasons for judgment.

[1] https://www.tripadvisor.com
[2] https://www.booking.com
[3] http://www.airbnb.com/

Extracting lived experiences from user reviews may therefore improve the process of taking a decision quickly about products and services, highlighting the reasons that trigger a particular opinion, allowing users to project themselves in a particular situation and understanding if they would be happy to share the same experience or not.

One of the problems with lived experiences is that it is sometimes difficult to tell what constitutes a lived experience or not. In general, we had to deal with a new kind of knowledge extraction task, Lived Experience Extraction (LEE). LEE may be considered partly related to event extraction, since events play a central role in lived experiences, being also well delimited in space and time. In this thesis, we focused on the problems of defining, finding and representing lived experiences. To this purpose, we relied on methods originated from Natural Language Processing and the Semantic Web, in particular deep parsing, event extraction, and sentiment analysis.

1.2 Outline

The rest of this thesis is organized as follows:

In Chapter 2 we show the state of the art related to Knowledge Extraction from text, in particular event extraction, in the Semantic Web context. We also collected some existing definitions of Lived Experiences.

In Chapter 3, we discuss the steps necessary to adapt the machine reader tool FRED[4] as an event extraction tool.

In Chapter 4, we show how event extraction on user reviews can be used to predict ratings in open rating systems. This result is important because it shows that events are correlated to the user ratings, and by consequence Lived Experiences too, since they are based on events.

In Chapter 5 we describe how we identify and extract lived experiences from user reviews, together with an experimental evaluation on hotel reviews.

In Chapter 6, we show some potential applications for lived experiences, such as the prediction of ratings in open rating systems and the summarization of reviews.

Finally, we draw a balance of the thesis in Chapter 7, summarizing the major contributions and the obtained results, with suggestions to some potential improvements and future work.

[4] http://wit.istc.cnr.it/stlab-tools/fred

Chapter 2

Related Work

Contents

2.1 Knowledge Extraction from Texts 5
2.1.1 Events 7
2.1.2 Event Extraction 11
2.1.3 Lived Experience Extraction 20
2.2 Semantic Web 24
2.2.1 Semantic Web Architecture 25
2.2.2 Resource Description Framework (RDF) 27
2.2.3 SPARQL Query Language 30
2.3 Sentiment Analysis 32
2.3.1 Polarity classification 32
2.3.2 Subjectivity analysis 34
2.3.3 Aspect-based Sentiment Analysis (ABSA) 35
2.3.4 Opinion Summarization 36
2.3.5 Opinion Spam Detection 38

This chapter consists in an overview of some topics that define the context and background of this thesis. First of all, in Section 2.1, we describe the general problem of text mining, and we focus in particular on some knowledge extraction tasks that are related to lived experiences, such as event extraction. In Section 2.2 we introduce the Semantic Web technologies that are at the basis of machine reading and are used in the machine reader that we used throughout this work, FRED, described extensively in Chapter 3. Finally, in Section 2.3 we provide an overview of opinion mining and sentiment analysis, which are related to the applicative domain of the thesis.

2.1 Knowledge Extraction from Texts

Most of the content available on the Web is in the form of text documents, such as news articles, blogs, wikis or postings on social networks. This content offers a great potential for finding useful information, but this potential is hampered by two problems: the quantity of the text and the fact that the text is written to be understood by humans, not by machines. Therefore, for several years, different branches of the domain of artificial intelligence have been studying this problem, with the objective to allow machines

to "understand" the meaning of a document, or, better, to transform it in a representation on which automated reasoning is possible. In other words, the aim is to transform unstructured information, which is described in natural language, into a set of structured knowledge described in a formal language. This transformation process is what is usually named *Knowledge Extraction*. This process aims to provide the user with the desired information without consulting many documents and thus facilitate access to the information [Poibeau (2003)].

In contrast to databases where data is stored in a structured and well-organized manner, texts written in natural language are considered to be unstructured sources. With the vertiginous increase of these textual resources, the automatic extraction of information sees a growing interest during the last twenty years. Indeed, drowned in this mass of this unstructured information, it is very necessary to develop automatic systems which are able, in our daily tasks (professional or personal), to retrieve and extract in a fast and efficient way the information that we need. In response to this, systems are designed to automatically analyze text in order to extract a set of relevant information [Hobbs et al. (2010)].

The field of knowledge extraction from texts has been developed during the 1980's and 1990's. Especially, with the emergence of evaluation campaigns such as MUC (Message Understanding Conference), ACE (Automatic Content Extraction), CONLL (Conference on Natural Language Learning), or TAC (Text Analysis Conference). The most common tasks in knowledge extraction are the extraction of named entities [Nadeau and Sekine (2007)], the extraction of relations between these entities [Rosario and Hearst (2005)], and the extraction of events (Section 2.1.2). We may consider that these tasks are the "building blocks" of a more complex knowledge extraction system that can assemble entities, relations and events together. Therefore, we provide a brief overview of each of these tasks.

Named Entity Recognition (NER)

The term *Named Entity (NE)* appeared during the sixth edition of the Message Understanding Conference (MUC)[1] evaluation campaign. The named entity recognition task concerns the identification of a set of the mentioned entities in the text. These entities can be relatively general, such as proper names (names of persons, organization, places, etc.), dates, monetary units, percentages, units of measures, etc. These objects are commonly called *Named Entities* and are indispensable to understand the meaning of a text. In the scope of this PhD thesis, these entities will be the participants in the lived experience events.

There are several systems dedicated to named entities recognition task [Van Hooland et al. (2015)]. We can cite Stanford Named Entity Recognizer[2], Alchemy[3], FOX[4],

[1] http://www-nlpir.nist.gov/related_projects/muc/
[2] http://nlp.Stanford.edu/software/CRF-NER.shtml
[3] http://www.alchemyapi.com
[4] http://fox-demo.aksw.org

Zemanta[5], NERD[6], AIDA[7]. The tool FRED[8] [Gangemi et al. (2016)] is a service web which allow to automatically transform knowledge extracted from text into RDF and OWL. In his actual version, it uses Apache Stanbol[9] and integrates TAGME [d'Aquin et al. (2008)] for named entity recognition and resolution.

Relation Extraction (RE)

The purpose of the extraction of relations is to study the semantic and syntactic links between several entities mentioned in the text. These relations may be binary (i.e. between two objects), or n-ary (more than two related objects). For example, detect, in a corpus of documents, that François Holland is the current president of the France, will be translated into a relation type *President* between the entity *François Holland* and the entity *France*. The detection of n-ary relations corresponds to what is called *record extraction*, where we identify a network of relations between entities. The extraction of events is a special case of this task.

Event extraction (EE)

Event extraction is another important task of information extraction. This task can be seen as a special form of n-ary relation extraction where an *action* is linked to other entities such as a date, place, participants, etc. This definition can vary according to the theoretical points of view and the application. The detection of events increasingly interest companies in many domains for its applications in economic and strategic intelligence. We present in Section 2.1.2 an overview of the techniques which are used for the detection and extraction of events from text.

2.1.1 Events

The concept of event is at the center of our work, since we consider events as the triggers of lived experiences. All experiences are related to "something that happened", which is a rough definition of event. For this reason, we are going to dedicate this subsection to the definition of events, a notion that has been widely used in Natural Language Processing with significant variance in its meaning.

Events are elusive entities; as the authors of [Welty and Aroyo (2012)] argue, even human annotators do not agree on what is an event and what is its boundary in terms of the extension of its participants, temporal and geospatial extent, etc. More aspects of events appear when trying to recognize or extract them from text: polarity of speaker's judgment on events, negation, modality, relations (temporal, causal, declarative, etc.) to other events, etc.

[5] http://programmableweb.com/api/zemanta
[6] http://nerd.eurecom.fr/
[7] https://gate.d5.mpi-inf.mpg.de/webaida/
[8] http://wit.istc.cnr.it/stlab-tools/fred
[9] http://stanbol.apache.org

For example, the text:

The Black Hand might not have decided to barbarously assassinate Franz Ferdinand after he arrived in Sarajevo on June 28th, 1914.

expresses three events (*decide, assassinate, arrive*), with *Black Hand* being a participant in two of them (*decide, assassinate*), *Franz Ferdinand* in the third (*arrive*), a temporal extent for the third (*June 28th, 1914*), and a relative temporal extent for the other two (given the third's extent and the past tense suffixes in the first and third), a geo-spatial extent (*Sarajevo*), a judgment with negative polarity on the second event (*barbarously*), a negation (*not*) over the modality (*might*) modifying the first event, and an explicit temporal relation between the second and third event (*after*).

Etymologically, "event" is a polysemic word that comes from the latin "venire" (become). The dictionary defines events as a thing that happens or something to which attach significance. In addition, the definition of event has received fundamental attention across academic fields, from philosophy [Casati and Varzi (2014), Davidson (1993)] to cognitive psychology [Zacks and Tversky (2001)].

In philosophy, events are properties, specifically properties of moments or intervals of time [Montague (1969)]:

> Then event of the sun's rising will be the property of being a moment at which the sun rises, and events in general will form a certain class of properties of moments [or intervals] of times. (1969: 149-150)

However, [Chisholm (1970)] show event as a state of affairs that is not time-bound and that therefore is such both it and its negation may occur. This definition is based on an assumption of events to states of affairs (i.e. entities that can be the objects of propositional attitudes) rather that to properties.

> A proposition could be defined as any state of affairs which i necessarily such that either it or its negation does not occur.... An event is any contingent state of affairs which is not a proposition and which implies change. (1970: 20)

For Davidson, events are "things that happen at some points in time" [Davidson (1993)]. By contrast, [Higginbotham et al. (2000)] found that categorizing events would be helpful to define them. They propose two categories of events *universal* and *particular*. Universal events were defined as "things that can recur or be instantiated at different places and time". However, particular events are "things that occur at a specific place and time".

Some philosophers, as in [Casati and Varzi (2014)], show that an event is often defined as an abstract concept, or defined within the context of a very specific domain (e.g. textual news, time series, social media).

In physics, Einstein in his Theory of Relativity used the term event to denote the fundamental entity of observed physical reality - a single point in the time-space continuum. In addition, an event denotes a phenomenon considered as localized and instantaneous, occurring at a point and at a certain moment[10].

In journalistic terms, an event denotes a topical fact, which may be predictable or unpredictable: Natural disaster, sporting outcome, scientific outcome, political outcome, election result, etc.

The notion of events has been also used in the linguistics literatures. WordNet[11] [Miller and Fellbaum (1998)], a large lexical database of English, defines events as "something that happens at a given place and time". In addition, linguists that have worked on the semantic structure of events in text suggest other definitions containing time and situations. [Chung and Timberlake (1985)] found that an event can be defined in terms of three components: a predicate; an interval of time on which the predicate occurs; and a situation under which the predicate occurs.

[Krieg-Planque (2009)] gives a simple definition of the event: "An event is an occurrence perceived as meaningful within certain framework". The term "occurrence" involves the notion of temporality that determines a "before" and an "after" this occurrence. The frame refers to a given system of expectation which determines whether the occurrence acquires (or not) its remarkability and, consequently, is promoted (or not) to the rank of event.

In the area of information retrieval, the notion of events has been used in the task of information extraction (e.g. entities, values, temporal expressions, relation, and events). Some evaluation frameworks (e.g. Automatic Content Extraction (ACE), Message Understanding Conference (MUC), TimeML, etc.), were proposed definitions for events to facilitate the task of event extraction.

The Automatic Content Extraction[12] (ACE) evaluation [Doddington et al. (2004)], for instance, defines an event as "a specific occurrence involving participants" and "something that happens". An event is identified via an event trigger and an event extent. The event trigger is the word that most clearly expresses the event's occurrence, and the extent, which indicates the scope of the event, is defined as the sentence in which the event trigger is mentioned.

These definitions make the implicit assumption that events should have one or more participants. Yet, not all events have a clearly defined set of participants, thus limiting its practical use. In addition, these definitions also do not explain how to address events with ambiguous semantic scope (e.g. the May 25, 2011 Japan earthquake). However, instead of defining all possible events, the ACE program a set of event types (e.g. Conflict) and subtypes (e.g. Attack) to be extracted from various text sources (e.g. Newswire, Blogs, Conversation, Transcripts) and provides a set of corresponding predefined templates along with their pre-defined attributes (time, place, participants, etc.).

[10] http://www.larousse.fr/encyclopedie/nom-commun-nom/événement/50167
[11] https://wordnet.princeton.edu/
[12] http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf

A template of the "Attack" event subtypes applied to the sentence "Rebel suicide bombers launched a simultaneous attack on China Bay air force base in Trincomalee, 240 km (150 miles) northeast of Colombo..." is presented in Table 2.1.

Attribute	**Description**	**Example**
Attacker	The attacking / instigating agent	Rebel suicide bombers
Target	The target of attack	China Bay air force
Means	The thing used in the attack	Rebel suicide bombers
Time	When the attack takes place	-
Place	Where the attack takes place	Trincomalee, 240 km (150 miles) northeast of Colombo

Table 2.1: Attack event template and sample extracted attributes

The ACE definitions are particular, but restricted to a small class of events. Focusing on a restricted class of events is often useful to eliminate ambiguity and enables precise annotations for evaluation purposes. In addition, this definition only applies to supervised detection task, where the classes of events that should be detected are known a priori. A disadvantage is that events such as Festivals and Concerts cannot be represented since there are no corresponding templates.

In MUC (Message Understanding Conference) [Chinchor (1998)], an event is related to an event category. More precisely, the category of event is associated with a template, which gathers the relevant information concerning the event. The participants in MUC had to create a template whenever they found an instance of a given event category in a document.

The associated information with the events in the templates generally takes the form of named entities. Therefore, the notion of event is materialized by a relation which can be either between named entities, or carried by a verb.

The ISO-TimeML model [Pustejovsky et al. (2010)] is a rich specification language for event and temporal expressions in natural language processing text. Originally, TimeML [Pustejovsky et al. (2003a)] has been developed in order to improve the performance of Question Answering systems.

Events in TimeML are taken to be situations that *occur* or *happen*. They can be punctual (e.g. John reached the summit) or last for a period of time (e.g. John walked up a mountain). TimeML focused on events expressed by tensed or untensed verbs, nominalisation, adjectives, predictive clauses, or prepositional phrases. In addition, TimeML also consider as events those predicates describing *states* or *circumstances* in which something obtains or holds true [Saurii et al. (2005)].

TimeML does not identify event participants, but the event tag and its attributes have been designed to interface with Named Entity taggers in a straightforward manner.

In Topic Detection and Tracking (TDT), [Allan (2002)], event was initially defined as "some unique thing that happens at some point in time" [Allan et al. (1998a)]. According

to this definition, a defined time period must be associated with the event. Therefore, this definition was further extended to include locations as well [YANG et al. (1999)], defining an event as "something that happens at a specific time and place".

The scope of an event, according to these definitions is open and may be interpreted in a several ways. For instance, under these definitions, The World Trade Center attacks that took place on September 11, 2001 is an event. However, textual news document that reports on this event might mention the subsequent collapse of the World Trade Center towers, which is an acceptable event according to the definition. Here, it is unclear whether the events in our examples should be considered as separate events or one collective event.

To address the ambiguity of the initial TDT definition, an amended definition was proposed [Allan (2002)], stating that an event is "a specific thing that happens at a specific time and place along with all necessary preconditions and unavoidable consequences". This new definition considers our previous examples of the World Trade Center attacks and subsequent collapse of the World Trade Center towers to be a single event. This definition opens a number of questions as to what are the necessary preconditions and unavoidable consequences for certain events [Makkonen (2003)]. For certain events (e.g., the terror attack in Paris 2015), some of the preconditions and unavoidable consequences are unknown or subject to discuss.

Overall, the TDT definitions of an event introduce some useful ideas and make some clarification regarding event boundaries, but they do not cover all possible types of events.

2.1.2 Event Extraction

Once we know what an event is, then the problem is to find and extract them. In Section 2.1.1, we presented four models for defining events (i.e. ACE, MUC, TimeML, and TDT) for the purpose of event extraction from text. In this section, we show some approaches which use these definitions in order to extract events. Then, we describe a number of representative approaches to event extraction from several sources as news, social media, blogs, and forums.

For ACE (Automatic Content Extraction) [Doddington et al. (2004)], several systems attempted to extract ACE events [Ahn (2006), Grishman et al. (2005), Hardy et al. (2006), Liao and Grishman (2010)]

In [Ahn (2006)], the authors presented a simple, modular approach to event extraction. They divided the event extraction task into different subtasks, including identification of event anchors and identification of event arguments, and then used machine learning methods to optimize and evaluate the results for each subtask.

An event anchor is a word that best captures the core meaning of an event. Therefore, the event anchors identification task was treated as as a word classification task where the task is to classify every term in a document with a label defined by 34 event types. Lexical, WordNet [Miller and Fellbaum (1998)], and Dependency features were used in this task in

order to train MegaM[13], a maximum entropy learner [Daumé III (2004)], and TiMBL[14], a memory-based learner, [Daelemans et al. (2004)] classifiers.

Event argument identification consists of determining entities, times, and values that are associated with each event. This task was treated as a single multi-class identification task and with training a separate multi-class classifier for each event type. To train the classifiers, a set of features such as POS tag, event type, entity mention, and dependency path were used.

The evaluation of these tasks was carried out using ACE 2005 training corpus, which contain 599 documents. 539 documents were used as training set and 60 documents as test set. For event anchor, the authors show that using the maximum entropy classifier and then the TiMBL classifier gives the best results. However, the two classifiers have the same performance for the argument identification task.

Several MUC (Message Understanding Conferences) campaigns are interested in filling predefined templates with a number of attributes from newspapers. As in information extraction, the literature of the domains offers both works based on symbolic approaches and statistical approaches.

The first symbolic approach is described in [Aone and Ramos-Santacruz (2000)] where the authors are interested in relations and events, as described in *MUC-7* [Chinchor (1998)]. They developed the system REES (Relation and Event Extraction System), which was evaluated by extracting 100 relation and event types, 61 of which are events, from a news source. The authors developed ontologies of the relation and events to be extracted for political, financial, business military, and life-related domains. Events are extracted with their participants, e.g. "who did what to whom when and where?" For examples, for a BUYING event, REES extracts the buyer, the artifact, the seller, and the time and the location of the BUYING event.

The system consists of three main parts: (1) A tagging module which consists of three modules: **NameTagger:** to recognize the names of people, organizations, places, and artifact. **NPTagger:** to recognize non-recursive Base Noun Phrase (BNP) [Ramshaw and Marcus (1995)], and then to detect complex NPs for the four main semantic types of NPs, i.e., Person, Organization, Location, and Artifact. **EventTagger:** to recognize events using lexico-syntatic patterns. These patterns tag events in the presence of at least one argument specified in the lexical entry for a predicate. Therefore, a lexicon entry is used for each event-denoting word, generally a verb. New types of events can be extracted by just adding new verb entries to the lexicon without creating new patterns. (2) A rule-based co-reference resolution module that resolves definite noun phrases of Organization, Person, and Location types, and singular person pronouns: he and she. (3) A template generation module which used declarative rules to generate and merge templates automatically achieve portability. MUC style data were used for training and testing. The system achieved a 0.70 F-Measure over 26 event types.

[13] http://www.isi.edu/~hdaume/megam/

[14] http://ilk.uvt.nl/timbl/

In terms of statistical approaches, [Chieu et al. (2003)] develops the system ALICE (Automated Learning-based Information Content Extraction) in order to extract events using statistical learning. They evaluated four classification algorithms (Maximum Entropy [Ratnaparkhi (1998)], SVM [Cortes and Vapnik (1995)], Naive Bayes [Duda et al. (1973)], and Decision Tree [Quinlan (2014)]) using the Weka package[15] on the MUC-4 data test. The objective of MUC-4 task is to extract information on terrorist events occurring in Latin American countries from free text documents. For this task, authors firstly extract manually templates paired with their corresponding documents that contain terrorist events for training.

A set of features has been used to train the classifiers as verbs, nouns, head word, Named Entity, and coreference features. The best results are obtained using Maximum Entropy classifier and this approach achieves accuracy competitive to the best of the MUC-4 systems.

For TimeML event types extraction, the work that has mainly drawing our attention are those of [Saurí et al. (2005)] and [Bethard and Martin (2006)].

[Saurí et al. (2005)] developed the EVITA (Events In Text Analyzer) event recognition tool. EVITA is used for event recognition and extraction in newswire text. Events are identified using lexical analysis, context analysis of verbs, lexical lookup of adjectival events, and machine learning to determine whether an ambiguous noun is used in an event sense. The TimeML annotation schema was used to identify time reference and events.

Preprocessing of the data was carried out using the Alembic Workbench tagger[16] for part of speech tagging, lemmatization, and chunking [Day et al. (1997)]. A shallow parser is used to retrieve event referring expression conveyed by three part of speech categories: verbes, nouns and adjectives. Events denoted by verbs are identified by lexical look-up and contextual parsing of the verbal chunk. Noun-based events are identified using WordNet[17] [Miller and Fellbaum (1998)] and disambiguation with Bayesian classifier trained on SemCor[18] texts. Adjectives are tagged as events when they appear as the head of a predicative complement as such in TimeBank1.2[19].

In addition to identifying events, EVITA also identifies certain grammatical features associated with them such as polarity, modality, tense and aspect. EVITA achieves a performance ratio of 0.80 F-measure.

[Bethard and Martin (2006)] show that the event identification task can be formulated as a classification task. They developed the system STEP that is capable to identify events in the purpose of question answering with a precision of 0.82 and recall of 0.71 on the TimeBank data. For each word in a document, they assign a label which indicates whether the word is inside or outside of an event. They use the standard B-I-O of the

[15] http://www.cs.waikato.ac.nz/ml/weka/
[16] http://www.timeml.org/terqas/alembic/AWB-overview.html
[17] https://wordnet.princeton.edu/
[18] http://web.eecs.umich.edu/~mihalcea/downloads.html#semcor
[19] http://www.timeml.org/

word chunking task that augments each class label with an indicator of whether the given word is Beginning, Inside, or Outside of a chunk [Ramshaw and Marcus (1995)].

In order to recognize whether a word is part of an event or not, they use a set of syntactic and semantic features (Affix features, Morphological features, Word class features, Temporal features, Negation features, WordNet hypernym features) as input to YamCha[20][Kudo and Matsumoto (2001)], a general-purpose chunker, and TinySVM[21] support vector machine implementation. Finally, each word in the document is classified as either beginning (B), inside (I) or outside (O) an event.

In addition to identifying events, STEP also identify the semantic class of an event with a precision of 0.67 and a recall of 0.51. [Bethard and Martin (2006)] show that identifying the semantic class of an event requires features that encode more of the semantic context of the words.

Event extraction from news articles had got the largest portion among the approach which attempt to study the event extraction task [Allan et al. (1998b), Chambers and Jurafsky (2011), Gabrilovich et al. (2004), Ploeger et al. (2013), Tanev et al. (2008)].

[Tanev et al. (2008)] employed clustering techniques for real-time event extraction from online news. They focused especially on violence and disaster events. Firstly, they collected the news articles using EMM system [Best et al. (2005)], which regularly checks for updates of headline across multiple sites. Next, the articles are grouped into clusters including documents on one topic. For each cluster, the system tries to extract the main event by analyzing all documents in the cluster.

A text processing task was applied into each document in each cluster in order to produce a more abstract representation of the documents. This task includes the following steps: tokenization, sentence splitting, named-entity recognition (e.g., people, numbers, locations).

The event extraction task was performed by creating extraction patterns using a machine learning and knowledge based techniques. Therefore, they firstly annotated a small corpus with event information, e.g., date, place, actors, affected dead, etc. Then, they learned automatically patterns for each event-specific semantic role and manually check, modify, filter out low quality patterns. Finally, the extracted patterns were matched against the first sentence and the title of each article from the cluster.

The evaluation has been carried out on 368 English language news cluster, where 29 violent events are described. The results show that the system was able to recognize 27 events of 29 where 0.93 of coverage.

[Ploeger et al. (2013)] introduce an automatic activist events extraction method from various news sources using NLP tools. They consider everything that *happens* as an event, which may have actors, location, and occurs at a point in time. Therefore, each verb in a sentence represents an event, since verbs report actions, occurrences, and state of affairs.

[20] http://chasen.org/~taku/software/yamcha/
[21] http://chasen.org/~taku/software/TinySVM/

A preprocessing task was applied to the text in order to extract events. Firstly, the text is split into sentences and words using Stanford's sentence splitter and word tokenizer[22]. Then, verbs are spotted using Stanford's part-of-speech tagger [Toutanova et al. (2003)]. Actors, places, dates, and times are recognized using Stanford's named entity recognizer [Finkel et al. (2005)].

In this research, the authors only consider named entities and timestamps grammatically dependent on a specific event to be part of that event. Therefore, Stanford's dependency parser [Klein and Manning (2003)] was used to resolve the dependency problem. Then, a date normalization task was applied to transform any relative timestamps (e.g. "Last Tuesday") into full dates (e.g. "23-06-2013"). In addition, a named entity disambiguation was realized using TextRazor's[23] API.

In order to represent extracted events, they use the Simple Event Model (SEM) [van Hage et al. (2011)] which uses a graph model defined using the Resource Description Framework Schema language (RDFS) and the Web Ontology Language (OWL). A corpus of 45 documents concerning arctic oil exploration activism was used in the evaluation. A gold standard was created by asking three domain experts to annotate every article with events, actors, places, and times. The system were able to extract 1829 events with 0.71 precision 0.58 recall and 0.64 F-Measure.

Since social networking sites such as Facebook[24] and Twitter[25] have become an important complementary source of such information, event extraction from these sites has been also studied.

In [Ritter et al. (2012)], authors presented an open domain event extraction within Twitter "TwiCal". They employ an NLP based approach to find named entities and event phrases from tweets. The extracted events are categorized into types based on latent variable model.

Event phrases could be verbs, nouns, and adjectives. Therefore, in order to extract events, [Ritter et al. (2012)] annotate 1000 tweets with event phrases, following annotation guidelines similar to those developed for the event tags in Timebank [Pustejovsky et al. (2003b)]. A contextual, dictionary, and orthographic features were used with features based on Twitter-tuned POS tagger [Ritter et al. (2011)] and event dictionary terms gathered from WordNet [Miller and Fellbaum (1998)] by [Saurí et al. (2005)] to build an event classifier "TwiCal-EVENT", which was able to identify events with 0.64 F-measure using 4-fold cross validation over the 1000 manually annotated tweets.

[Di Eugenio et al. (2013)] present experiments on detecting life-event of a user from twitter. They define life events as a personal information concerning marriage, birth of a child, graduation, or losing or getting a job. In their work, they are interested in detecting two life events: *marriage* and *employment*. A list of keywords was used to collect tweets.

[22] nlp.stanford.edu/software/tokenizer.shtml
[23] http://www.textrazor.com/
[24] http://www.facebook.com
[25] http://www.twitter.com

For *employment*, they used four keywords: *new job, laid off, interview,* and *job offer*. However, For *marriage*, they select the top three relevant keywords as ranked by using TF-IDF on multiple documents gathered by mining domain specific websites (brides.com; weddingstylemagazine.com; weddings-magazine.com; theknot.com; insideweddings.com). The three keywords turned out to be: *engaged, married, wedding*. Using these keywords, 4395 tweets were collected and used in the experiments.

In order to detect life events, a machine learning approach was used. Therefore, 2250 tweets were manually annotated with two or three classes: for example for marriage tweets, the class *YES-Tweeter* was assigned to tweets where the tweeter is getting married. The class *YES-Other* was assigned to tweets where somebody else is getting married. The class *NO* was assigned to the rest. The inter agreement between annotators was acceptable on marriage ($K = 0.72$) and excellent on employment ($k = 0.88$).

A several features were used with several machine learning techniques to build the classifier. The results show that unigram model, bag-of-word, using Complement Naive Bayes (CNB) [Rennie et al. (2003)] and Support Vector Machine (SVM) [Cortes and Vapnik (1995)] perform the highest accuracy. A bigram model improves slightly the results.

A semantic role labeling model was also used. This model consists of adding roles to the semantic arguments associated with the predicate. e.g. for *marry* Arg0 is the *causer*, whereas Arg1 and Arg2 represent the two married. The results on this model show that there was no improvement as concerns Employment and a few improvements for Marriage.

[Dickinson et al. (2015)] also study the automatic identification of prominent life events in Twitter. They focused on five life events: *having children, beginning school, marriage, parent's death,* and *falling in love*. A set of 1 million English written tweets was extracted per life event using a list of keywords. For example, for getting married, *wedding, marriage,*and *church* was used. WordNet [Miller and Fellbaum (1998)] was also used in order to find related terms. In addition, they also used OnlineSlangDictionnary[26] to extract slang phrases. The different tenses for each verb were also extracted using Verbix[27].

A machine learning approach was tested to automatically identify life events. This requires the development of a set of annotated training examples. Therefore, a corpus of 14k tweets was annotated using the Crowdflower[28] platform[29]. The dataset with two questions was uploaded to Crowdflower. The first question: ***Is this tweet related to a particular topic theme?***. The second one: ***Is this tweet about an important life event?***. Intersecting the two sets of answers where both are "yes" allows to obtain tweets about selected life events. By the end, 2241 tweets, which were written about an event and their target theme, were generated.

[26] `http://onlinesslangdictionary.com/thesaurus/`

[27] `http://www.verbix.com`

[28] `http://www.crowdflower.com/`

[29] Crowdflower is an online crowd sourcing annotating platform, where uploaded datasets are accompanied by questions for the crowd to make judgements on.

To achieve the goal of this work, [Dickinson et al. (2015)] train a machine learning classifier using a set of content, semantic, and user features. They used three classifiers J48, Naive Bayes (NB) [Duda et al. (1973)], and Support Vector Machines (SVM) [Cortes and Vapnik (1995)] and obtained results between 0.84 and 0.92 F1-measure.

Twitter has been also used to extract sport events. [Van Oorschot et al. (2012)] extract game events (e.g. goals, fouls) from tweets about football match to automatically generate match summaries. Events were detected and classified using a machine learning approach. They collected tweets about 61 Dutch premier league soccer matches. For each match, they extracted the event time, and then classify the events using a pre-defined list of event types.

Event times was detected by investigating three different peak detection methods. The first method checks, for each minute, if it is a local maximum of a window of two neighboring minutes. If so, they select this minute as being a peak. The second method looks at the difference in levels between different minutes and decides a minute is a peak when its change in volume compared to the previous minute is higher than a certain threshold. The third method uses the intensity threshold measure for picking peaks and also apply baseline correction to the tweet volume per minute signal. The results show that on average in 10.80 of the minutes of each games an event happens.

A machine learning approach was employed in order to classify the types of events. 5 types of events was used: *Goal scored, own goal scored, yellow card, red card,* and *player substitution*. Different features such as word frequency, information gain, and gain ratio are used to train a SVM classifier. The experimentation was carried out with three sets of game minutes: All minutes, Peak minutes, and event minutes (the minutes where an event takes place). The result shows that event minutes give the best results on event classification with 0.82 F-Measure.

Other works were studied the event extraction task from sports data based on web casting and broadcast video [Tan et al. (2000), Xu et al. (2006, 2004), Zhang et al. (2007)].

In [Zhang et al. (2007)], authors present a multi-modal framework for semantic event extraction from basketball games. The framework is based on web-casting text analysis. Web-casting text is the description of the game progress. This description contains the important events during the game with other information such as time, players, teams, actions, etc.

The objectives of this research are to firstly cluster the descriptions into different groups corresponding to certain events and then to extract keywords from the descriptions in each group for event detection. Therefore, each description was considered as one document, and each document was transferred into term-document vector. Cosine distance between vectors was applied in order to cluster descriptions into different group. Since the number of semantic events for the basketball game is limited, semantic event for each group was detected by using a pre-defined list of basketball events [Xu et al. (2006)] such as *Shot, Jumper, Layup, Foul, etc.*. Each semantic event features one or several keywords related to

this event (e.g. the *Shot* event features the *shot, header* keywords). Therefore, by detecting these keywords, the relevant event can be recognized.

All the description of the game was then clustered into 9 groups corresponding to 9 semantic events. The terms in each group were ranked using *tf*idf*, where *tf* is the term occurrence frequency in each group and *idf* is the inverse term occurrence frequency in the entire document corpus. The authors view that the first rank words can be selected as keywords for each group. Therefore, the text events can be detected by finding the item of description which contain the keywords and analyzing context information in the description.

Health social forums contain concentrated discussion about treatments for a particular disease. Therefore, there has been an increased interest in analyses of their content. One of these interests is the extraction of adverse drug events since analyzing patient reports of these events may add value to the current practice of Pharmacovigilance by providing new perspectives for understanding drug effectiveness and side effects timely [Benton et al. (2011)].

For this purpose, several studies have adopted co-occurrence analysis approaches to extract adverse drug event from patient forums [Bian et al. (2012), Leaman et al. (2010), Liu and Chen (2015)].

For instance, [Liu and Chen (2015)] exploited patient forums to extract adverse drug reactions. They developed a framework which consists of medical entity extraction for recognizing patient mentions of drug and events, adverse drug event extraction using a statistical learning technique.

Authors, firstly, collected 184.874 posting contributed by the American Diabetes Association (ADA) online community[30]. Then, a preprocessing task, which consist of text cleaning and sentence segmentation, was realized. Then, they used a statistical learning technique namely, Transductive SVM [Joachims (1999)] to extract relationships between drugs and their adverse reactions using syntactic features. Therefore, syntactic dependency tree is generated using Stanford Parser, and the shortest path is detected to extract relevant syntactic dependency features to be used for TSVM learning.

The system achieved poor results with bag of words features, 0.60 with syntactic dependency features and 0.69 when performing semantic filtering consisting on negative expression removal.

By the end, events were also used to build a commonsense knowledge Database from the web. [Hung et al. (2010)] elaborate on a framework that can be employed for mining the web for event-based commonsense knowledge by using lexico-syntactic patterns matching and semantic role labeling. A large number of raw sentences that possibly contain target knowledge is collected through Web search engines. Web queries are formulated based on a set of lexico-syntactic patterns.

[30] `http://community.diabetes.org`

For each raw sentence, the semantic roles were identified. A semantic role is the relationships that syntactic arguments have with verbs. The PropBank annotation scheme [Palmer et al. (2005)] was used in order to identify six arguments which are associated with a verb. After labeling the semantic rules, knowledge is extracted and stored in a database.

In order to keep the final knowledge item reliable, [Hung et al. (2010)] proposed a strategy which allows to prune knowledge items with a high probability of questionable semantic roles. The evaluation results showed that the proposed approach could automatically accumulate commonsense knowledge efficiently, with an accuracy rate close to 0.98.

The proposed approaches in the literature extract events from news articles, Twitter, or some annotated text. Due to the difference in structure, these works are not suitable for extracting events from user reviews. For example, tweets typically include a single event while the user reviews include a sequence of events, which make our task different from that discussed in [Di Eugenio et al. (2013), Ritter et al. (2012)] and [Dickinson et al. (2015)].

In addition, several approaches mentioned above use a predefined list of potentially interesting events [Di Eugenio et al. (2013), Zhang et al. (2007)], and [Van Oorschot et al. (2012)]. Unlike these approaches, we aim to extract all mentioned events in the review. We do not use a predefined list of events, but we performed a deep semantic parsing of text in order to obtain a semantic knowledge graph representation of it.

According to [McClosky et al. (2011)], parsing is an attractive approach for extracting events and allows to extract events with their arguments dependently. Therefore, the authors proposed an approach for supervised event extraction by taking the tree of event-argument relations and using it directly as the representation in a reranking dependency parser. This approach performs the event extraction task in three steps: (1) anchor recognition to identify and label event anchors, (2) event parsing to form candidate event structures by linking entities and event anchors. (3) event reranking in order to select the best candidate event structure.

In order to evaluate this approach, [McClosky et al. (2011)] have used the bio-molecular event corpus from the BioNLP'9 shared task [Kim et al. (2009)], which addresses the biomolecular event extraction task. They obtained a F-Measure of 53.5% in the development set, which contains 150 biomedical abstracts (1809 events), and 48.6% in the test set, which contain 260 abstracts (3182 events). These results show that parsing approaches are useful for the event extraction task and give competitive results.

In this PhD thesis, we aim to extract authentic lived experiences from user reviews (See chapter 5). We assumed that events could be the main elements in lived experience contents. Therefore, we used the extracted events from the reviews to firstly identify reviews which contain lived experience contents. Then, we used these events with their participants to extract authentic lived experiences and represent these experiences as event sub-graphs.

In the following section, we show some definition of lived experience from several domains and describe some approaches for defining lived experience.

2.1.3 Lived Experience Extraction

The concept of lived experience involves what an experience is like for a person and how the person recognizes and interprets the self-experienced experience. In distinguishing between phenomenology and hermeneutics, it can be said that phenomenology concerns the lived experience and is a descriptive way of seeing how things appear to a specific person, and hermeneutics concern the person's interpretation of his or her lived experience involving a specific phenomenon [Björk et al. (2005)].

The lived experience extraction task is not well studied yet. Lived experiences have been studied mainly in the context of anthropological, historical, and health studies. For example, [Halldórsdóttir and Hamrin (1996), Karian et al. (1998)], and [Pascal (2010)] explored the lived experience of having cancer, as perceived by people who have been diagnosed and treated for cancer, in order to understand this complex phenomenon. In these studies, data were collected through interviews with people who were in the recovery phase of cancer. The finding can help the effort to understand and support cancer patients on their illness trajectory.

Lived experience was also used to study other diseases. [Temple-Smith et al. (2004)], and [Treloar and Rhodes (2009)] focused on the live experience of hepatitis C[31]. They searched to explore how people who injected drugs perceives hepatitis C.

Several research in anthropology have studied the lived experience. For example, [Dzurec (2000), Hart and Grace (2000), Parse (2003), Stuifbergen and Rogers (1997)] have studied the lived experience of fatigue.

[Parse (2003)], for example, used parse research method [Parse (1987, 1998, 2001)] to discover the meaning of feeling very tired. Therefore 10 women were asked to answer the question: *What is the structure of the lived experience of feeling very tired ?*. The results of this study indicate that the lived experience of feeling very tired devitalizing languor arising with engaging endeavors amid pulsating moments of repose-revive.

The lived experience of human-becoming was studied by [Kagan (2004)]. The purpose of this study is to discover the structure of the lived experience of feeling listened to[32]. The parse [Parse (1998, 2001)] research method was chosen to guide the study and to discover the structure of lived experiences of feeling listened to. Therefore, 10 person, living in Chicago, were asked about their experiences feeling listened to. The results of this study is that lived experience of feeling listened to is an unreserved affirmation amid potential irreverence arising with the liberating contentment of benevolent affiliations.

[31] Hepatitis C virus infection (HCV) is a blood-borne virus with injecting drug use being the primary route of transmission in most countries.

[32] A phenomenon of health and quality of life related to the human-to-human relationship.

In [Edmonds (2010)], authors explored the lived experience of nursing students who study abroad. They identified the benefits and impediments that could be used to produce future research and to develop existing study abroad programs. In order to collect the data, the question *What is the lived experience of nursing students who study abroad?* was asked to students who completed a nursing study in England or Dominica between 2006 and 2008. The results show that there are vast benefits of study abroad programs of nursing including, increased personal growth, awareness of diverse cultures, adapting despite an unfamiliar environment, and increasing self-efficacy.

[Connell (2003)] searched to understand the lived experience of daughters and their elderly mothers in a caregiving situation. Phenomenology[33] method was used to explore the day-to-day lived experience for adult daughters caring for their mothers and for the mothers receiving the care. The data were collected by conducting interviews with five mother-daughter dyads.

Each care-receiving mother was asked *What it is the experience like for you to be cared for by your daughter?*. In addition, each adult caregiving daughter was asked *What is the experience like for you in caring for your mother?*. All interviews were transcribed by the researcher and the text was manually analyzed.

The results of this study show that mothers and daughters share the same joys and worries, from different perspectives. Mothers worried about falling and injuring themselves and daughters worried about making right decisions that affected their mothers' health.

All the mentioned study manually identified lived experience for such domains. The authors did interviews with users who have the experience in the domain. Then, the interviews were transcribed and the text was manually analyzed in order to detect and identify the lived experiences. In addition, the identification of lived experiences in the touristic domain still little touched. Researchers in human science, sociology, or philosophy do not attempt to study the lived experiences of users who visit a city or try a product.

To our knowledge, the only studies devoted specifically to lived experiences from the information extraction perspective concern *personal story identification* [Gordon and Swanson (2009), Gordon (2008)], and *experience mining* [Inui et al. (2008)].

In [Gordon and Swanson (2009), Gordon (2008)], authors identified personal stories in weblog entries using machine learning techniques. They defined personal stories as textual discourse that describes a specific series of causally related events in the past, spanning a period of time of minutes, hours, or days, where the author or a close associate is among the participants. Given this definition, the authors expected to more frequently see first person pronouns (i.e. I, me, my) along with a greater proportion of past tense verbs.

In their work, the story identification task has been treated as a binary classification task. The authors developed an automated story classifier using supervised machine learn-

[33] Phenomenology is a way to investigate subjective phenomena, and is based on the belief that essential truth about reality are grounded in everyday experience [Spielgelberg (1975), Van Manen (1984, 2015)]

ing techniques in order to identify weblog posts that can be characterized as personal story.

For this aim, a corpus of 5002 weblog entries was manually annotated. Each weblog entry was assigned by *story* or *non-story* label. Blog entries would be labeled as *story* if the annotator judged that the content of a blog post contains story content, even if some non-story text was included. The label *non-story* was assigned to weblog entries which do not contain story content. Accordingly, the label *story* was assigned to 240 of these entries (4.8%).

In order to build the classifier, 4252 of the annotated weblog entries were used as a training set. A several variation of n-gram features (e.g. unigram and bigram) were investigated to train a Support Vector Machine learning algorithm (SVM) [Cortes and Vapnik (1995)]. A 10-fold cross validation technique [Devijver and Kittler (1982)] was applied and the best results were obtained using unigram feature sets (Precision = 66%, Recall = 48%, F-measure = 55%). Table 2.2 show a list of the top features indicative of stories and non-stories, which are used in the classification task and achieved the best results.

Story Features	**Non-Story Features**
went	will
send	1
took	/
back	years
i	blog
had	has
evening	team
down	many
comments	can
friend	are
was	love
art	being
got	use
did	before
headed	:

Table 2.2: The top 15 features for each class in [Gordon (2008)] system

By the end of this work, [Gordon and Swanson (2009), Gordon (2008)] were able to create a large-scale comprehensive corpus of personal stories found in English-language weblogs. They applied their classifier to the ICWSM 2009 Spinn3r Dataset [Burton et al. (2009)][34] and identified a million weblog entries that contain personal stories.

[34] a collection of ten millions of weblog entries written between August 1 and October 1,2008.

Experience mining [Inui et al. (2008)] substantially agrees with personal story extraction in the kind of data and extraction criteria, involving factuality, direction involvement of the author as the experiencer, and the kind of event to be considered.

In their work, the authors developed a language processing technology for automatic extraction of personal experiences from English and Japanese weblog posts. Therefore, they firstly collected personal experiences concerning consumer products (automobiles, cellular phones, etc.), touristic services, and then store them in a huge database. Each experience was represented as a piece of structured information comprising such slots as topic (*What the experience is about?*), experiencer (*the author of the text*), event expression (*What event is experienced?*), event type (*the semantic type of the experienced event*), and factuality (*the temporal and modal status of the event*).

[Inui et al. (2008)] decomposed this task into four subtasks. The first subtask concerns the extraction of the mentioned events from the text. For this purpose, the authors built a typological lexicon of expression of experiences. They considered three categories of experience: (1) Sentiment experiences: predicative expression of Emotion (*enjoy, disappointed*), Evaluation (*tasty, inconvenient*), and Reputation (*popular, criticised*), where each expression has a sentiment orientation (i.e. positive or negative). (2) Happening experiences: predicative expression referring to a non-volitional event or state which is related to the use of a topic object and has a sentiment orientation (*pass an exam, get slim, broken, released, get used to,etc.*). (3) Action experience: predicative referring to experiencers' volitional actions related to the use of a topic object and do not necessarily involve sentiment orientation.

Expression of emotion, evaluation, and reputation have been collected in existing sentiment lexicons such as SentiWordNet [Esuli and Sebastiani (2006)] for English and Kobayashi's sentiment lexicon [Kobayashi et al. (2007)] as well as Higashiyama's sentiment lexicon [Higashiyama et al. (2008)] for Japanese. For the action expression, WordNet-like lexicon has been employed. Happening expression has been collected by employing lexicons such as those automatically acquired by [Takamura et al. (2005)].

The second subtask was Entity-event relation extraction which consist of identifying the relation instances between an event and its subject. The third subtask concern the identification of the experiencer of each experience. This task was manipulated as an extension of identifying opinion holders [Kobayashi et al. (2007)].

The last subtask was factuality analysis. For each mention, the authors identified the modal status of the event entity referred to in the event mention. Therefore, they annotated each event mention in a given text with a triplet <*Event-time, Modality, Modality-time*>. The *Event-time* represents the tense, aspect and polarity status of the event. The *Modality* specifies the author's mental or communicative attitude toward the event in question. They defined 9 modality classes *(Affirm, Infer, Doubt,Hear, Intend, Ask, Recommend, Hypothesize, Other)*. The *Modality-time* slot describes the tense, aspect, polarity of the modality.

The factuality analysis was automated by creating a manually annotated corpus and exploiting the machine learning techniques. Two annotators were asked to annotate the event mentions with factuality triplet. 2646 sentences, including 4417 event mentions were annotated. A set of bag-of-word features with part-of-speech tags and lexemes were used to train two machine learning models. The first model consists of using SVM-HMM algorithm [Tsochantaridis et al. (2005)] to train an *Event-time* model and SVM-Multiclass package [Tsochantaridis et al. (2005)] to train a *Modality* model, which took care of the *Modality* slot independently of the *Event-time* slots. The second model was the Factorial CRFs (Conditional Random Fields) model which was employed using the GRMM toolkit[35] [Sutton (2006)].

For the evaluation, a three fold cross validation technique [Devijver and Kittler (1982)] was applied on the annotated corpus. The results show that the Factorial CRF-based model performs better than the SVM model for all the slots. This shows the importance of considering the inter-dependency between neighbored labels in the task of factuality analysis.

This work was also supported by an application system[36], which stores $50M$ experiences instances extracted from $150M$ Japanese blog posts with semantic indices. Given one or more topic objects specified by a user, the system provides the user experiences related to the topic.

The main objective of this PhD thesis is to extract authentic lived experience from user reviews based on the semantic contents of reviews (See chapter 5). For this purpose, we operationally defined a lived user experience as an event mentioned in a review, where the author is among the participants. Our approach consists of three steps: (1) identify reviews containing lived experience contents. This step is close to the work of [Gordon and Swanson (2009), Gordon (2008)]. However, we used semantic features such as personal events, and their participants to identify the relevant reviews. (2) extract lived experience sentences from the reviews which contain these contents. (3) represent the extracted lived experiences as event sub-graphs.

2.2 Semantic Web

The web has become an inexhaustible source of information. This ranges from simple textual documents to multimedia content. The volume of these data increases exponentially from year to year. Because of this large amount of data, we quickly become overwhelmed by the amount of documents. Although many tools, such as search engines and content aggregators, give us access to the information, this remains insufficient, especially in the era of the deluge of data. In addition, in traditional WWW, the data is unstructured and hard to reuse when cross domain. Therefore, it is necessary to develop techniques for

[35] `http://mallet.cs.umass.edu`
[36] `http://minna.naist.jp/`

interacting with these data. These interactions must enable us to enrich the current Web with innovative and high-potential features.

For this purpose, the W3C[37] is drawn by creating the Semantic Web [Berners-Lee et al. (2001)]. This is to provide technologies that allow the transition from a static Web to a Web with data that can be interpreted by both humans and machines. The objective of this Web is to link data from different sites in order to provide a structure which allows machines to communicate, exchange, and interpret data and enables users to find, share, and combine information more easily. This initiative is also known as *Linked Data, Linked Open data* or *Linking Open Data.*

Figure 2.1 presents the current status of the LOD[38] in the form of a graph where the nodes correspond to knowledge bases respecting the principles of the Web Semantic and the arcs represent the links between these bases. Among the knowledge bases, the semantic base *DBPedia* provides a large portion of the content of Wikipedia and incorporate links to other knowledge bases such as *GeoNames, W3C, Foaf-Profiles, etc.* The relations between the knowledge bases (in the form of RDF triples, Section 2.2.2) allow Web applications to provide better service to their user by exploiting additional knowledge from other knowledge bases.

In summary, the Semantic Web could be defined as a system that enables machines to understand and respond to complex human requests based on their meaning. Such an "understanding" requires that the relevant information sources to be semantically structured. According to the W3C, "The Semantic Web provides a common framework that allows data too be shared and reused across application, enterprise, and community boundaries."

This version of the Web is only an evolution of the current Web. HTML, CSS, and HTTP will always be used, but should lead to a revolution of its. To achieve this goal, a set of languages is proposed in order to automatically represent and manipulate the data on the Web. In this section, we will look at these different languages. We start with the RDF data model (Section 2.2.2). Then, we will present the SPARQL query language (Section 2.2.3).

2.2.1 Semantic Web Architecture

In this section, we introduce a commonly global architecture of Semantic Web [Horrocks et al. (2005);Harth et al. (2011)]. The Semantic Web defined by Tim Berners-Lee is structured in layers as shown in figure 2.2. This figure presents the technologies in superimposed layers, from the most detailed element to the most abstract one[39].

In this architecture, we identify three main parts:

[37] World Wide Web Consortium,http://www.w3.org/

[38] Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

[39] http://www.w3.org/2007/03/layerCake.png

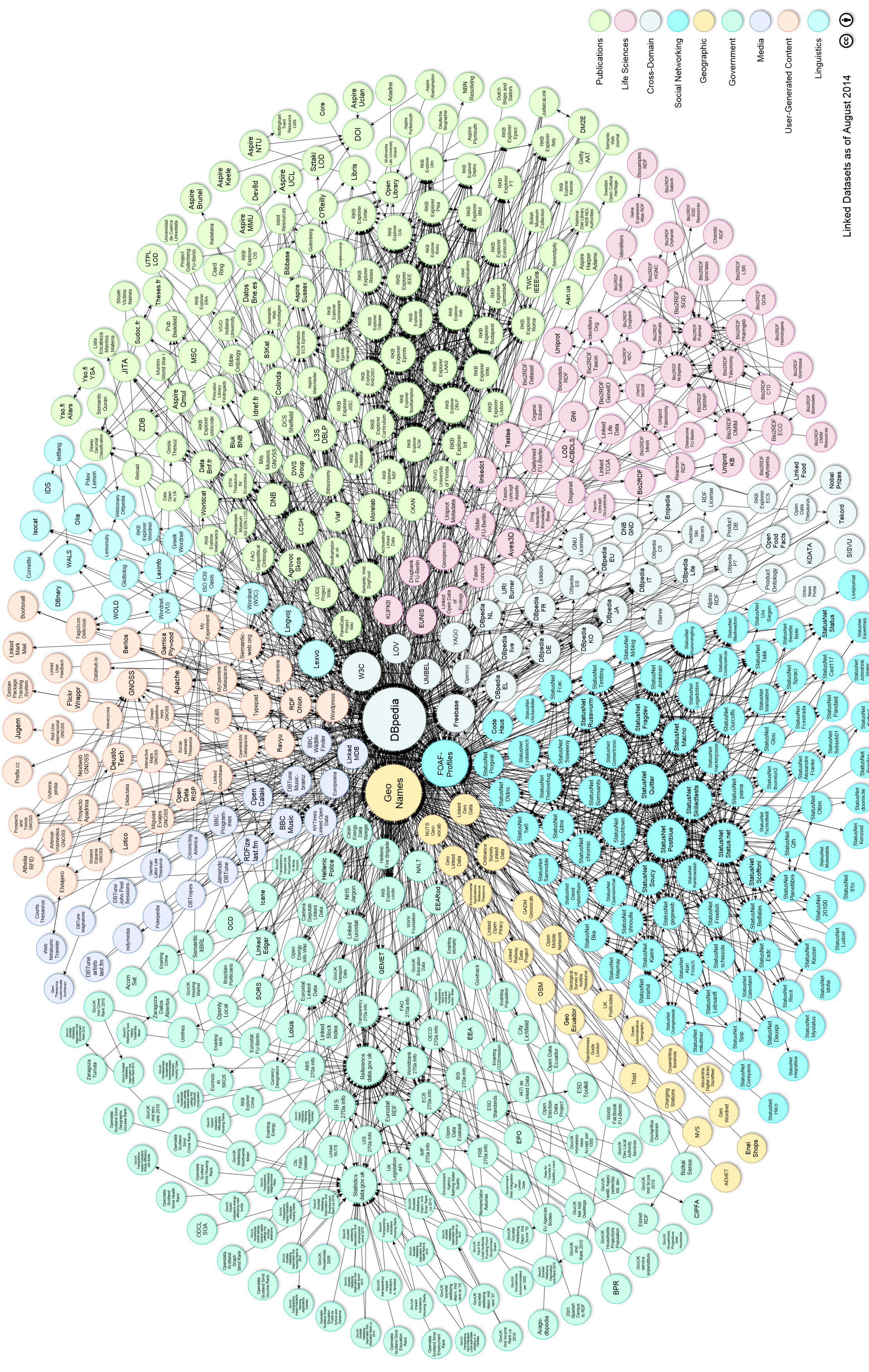


Figure 2.1: Linked Open Data

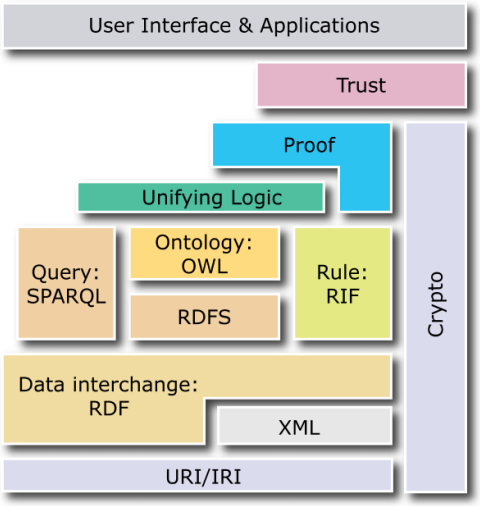


Figure 2.2: Semantic Web Stack

- The protocols URI/IRI: refer to protocols which identify resources on the Web. Each resource should be uniquely identified. It gives access to the description of a given entity.
- Languages:
  - RDF/XML: are simple languages for expressing data models, which refer to objects and their relationship.
  - SPARQL/RDFS/OWL/RIF: refers to the modeling and interrogation languages of semantic data.
- Software components: this part allows the implementation of applications which are based on the protocols defined above. These applications rely on reasoning rules that allow to infer new knowledge.

  In the following sections, we will focus on the lower layers of the architecture, namely the RDF language and the SPARQL query language.

2.2.2 Resource Description Framework (RDF)

In the Semantic Web, data are represented in Resource Description Framework (RDF) format [Klyne and Carroll (2006)]. RDF is a data description language, recommended by the W3C, allowing to represent information about resources in a graph format. The resource can be anything, including documents, people, physical objects, and abstract concepts. RDF is designed to represent structured metadata of WWW resource, such as the title, author, time, location.

W3C proposed the first specification of RDF in 1999[40]. This specification has been following by another in 2004[41]. More recently, a new version denoted RDF1.1 was published

[40] http://www.w3.org/TR/1990/REC-rdf-syntax-19990222/
[41] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

in 2014[42]. The aim is to propose a data model simpler than other proposed models as XML, in order to make the interaction and the diffusion of data on the Web more easy. RDF allows to make statements about resources. A statement consists of three elements, namely triples, and has the following structure:

<Subject> <Predicate> <Object>

The subject refers to a resource; the predicate is the relationship between the subject and the object; the object refers to a resource or an attribute describing the subject. A resource is identified by a URI (Unique Resource Identifier) [Masinter et al. (2005)], which may be, for convenience, shortened by a *namespace*, or an IRI (Internationalized Resource Identifier). The notion of IRI is a generalization of URI, allowing non-ASCII characters to be used in the IRI character string. URI can appear in all three positions of triple and should not contain special characters as (“<”, “>”, “!”, “\\”, “|”, “{”, and “}”).

There are several syntax to represent RDF data:

- RDF/XML [Beckett and McBride (2004)] : is the first created representation. It provides an XML syntax for RDF data. Triples are represented using XML tags. In addition, triples are specified within an XML element *rdf:RDF*. URI is identified by *rdf:about* attribute. Figure 2.3 gives an overview of RDF/XML representation for a small description for the resource “Syria”. In this example, we have one subject: [*"http://www.example.org/Location/Syria"*], two predicates: [*hasCapital, hasPopulation*], and two objects [*"http://www.example.org/Location/Damas", 23000000*>]

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ns="http://www.example.org/vocabulary/"
    <rdf:Description rdf:about="http://www.example.org/Location/Syria">
        <ns:hasCapital   rdf:resource="http://www.example.org/Location/Damas">
        <ns:hasPopulation> 23000000 </ns:hasPopulation>
    </rdf:Description>
</rdf:RDF>
```

Figure 2.3: RDF/XML Description for Syria

The *xmlns:rdf* namespace, specifies that elements with the *rdf* prefix are from the namespace *"http://www.w3.org/1999/02/22-rdf-syntax-ns#"*. The *xmlns:ns* namespace, specifies that the elements with the *ns* prefix are from the namespace *"http://www.example.org/vocbulary/"*. The <*rdf:Description*> element identifies a resource with the *rdf:about* attribute and contains elements that describe the resource.

- N-Triples [Carothers and Seaborne (2014)] : provides a simple line-based, plain-text way for serializing RDF data. This type of representation allows to present each

[42] http://www.w3.org/TR/rdf11-concepts/

triple in one line. In addition, each triple must end by a ".". The URI should be in "<>". Figure 2.4 shows the representation of the source "Syria" using N-Triples.

```
<"http://www.example.org/Location/Syria"> <"http://www.example.org/vocabulary/hasCapital"> <"http://www.example.org/Location/Damas"> .
<"http://www.example.org/Location/Syria"> <"http://www.example.org/vocabulary/hasPopulation"> "23000000" .
```

Figure 2.4: RDF Description for Syria using N-Triple

- Turtle/Notation3 [Beckett et al. (2008)] : is an extension of N-Triples. It allows a more concise representation for RDF data. For example, it use "," to indicate the presence of several pairs <subject,predicate> for the same object or ";" to indicate the presence of several predicates for the same subject. In figure 2.5, we show the representation Turtle for our example.

```
@prefix rdf: <"http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ns: <"http://www.example.org/vocabulary/> .
<"http://www.example.org/Location/Syria">  rdf:type ns:Location ;
                                ns:hasCapital  <http://www.example.org/Location/Damas> ;
                                ns:hasPopulation "23000000" .
```

Figure 2.5: RDF Description for Syria using Turtle

- RDF/JSON [Davis et al. (2013)] : This representation allows to serialize such a set of RDF triples as a series of nested data structures. In general, a triple (subject S, Predicate P, object O) is serialized as {"S" : { "P" : [*O*]}}. The object of the triple O is represented as a further JSON object with several keys such as *value*, and *type*. Type: is the type of the object (uri, literal,...). The value is the URI of the object, its lexical value or a blank node label depending on whether the object is a uri, literal or b-node. In figure 2.6, An example of two triples that share the same subject, but have differing objects and predicates.

```
{"http://www.example.org/Location/Syria" : { "http://www.example.org/vocabulary/hasCapital" :
                                                    [{"value": "http://www.example.org/Location/Damas",
                                                      "type": "uri"}],
                                             "http://www.example.org/vocabulary/hasPopulation" :
                                                    [{"value": "23000000",
                                                      "type": "literal"}]
                                           }
}
```

Figure 2.6: RDF Description for Syria using RDF/JSON

2.2.3 SPARQL Query Language

Since the RDF model became a W3C recommendation, it was necessary to create a query language for the RDF data. Several languages for querying RDF data have been proposed and implemented (e.g. TRIPLE [Sintek and Decker (2001)], RQL [Karvounarakis et al. (2002)], RDQL [Seaborne (2004)], SeRQL [Broekstra and Kampman (2006)], SPARQL [Prud'Hommeaux et al. (2008))]. A comparison of these languages is available [Haase et al. (2004) ; Angles and Gutierrez (2005) ; Hutt (2005)]. These languages must allow to query an RDF graph, to make a simple and complex selection, and to do an insertion or an update.

The SPARQL language is becoming the reference language for querying RDF datasets. On January 2008, $SPARQL1.1$ became an official W3C Recommendation [Prud'Hommeaux et al. (2008)] for querying semantic graphs, and $SPARQL1.1$ in March, 2013 [Harris et al. (2013)].

SPARQL can be used to express queries across diverse data source, whether the data is stored as RDF or viewed as RDF via middleware. SPARQL supports multiple matches (multiple selected variables), aggregation, subqueries, negation, constraining values in the form of Boolean-valued expressions, and results construction. The results of SPARQL queries can be result sets or RDF graphs [Harris et al. (2013)].

SPARQL defines a query language with a SQL-like style, where a simple query is based on query patterns, and query processing consists of binding of variables to generate pattern solutions (graph pattern matching). The structure of a SPARQL query [Domingue et al. (2011)] is shown in Figure 2.7.

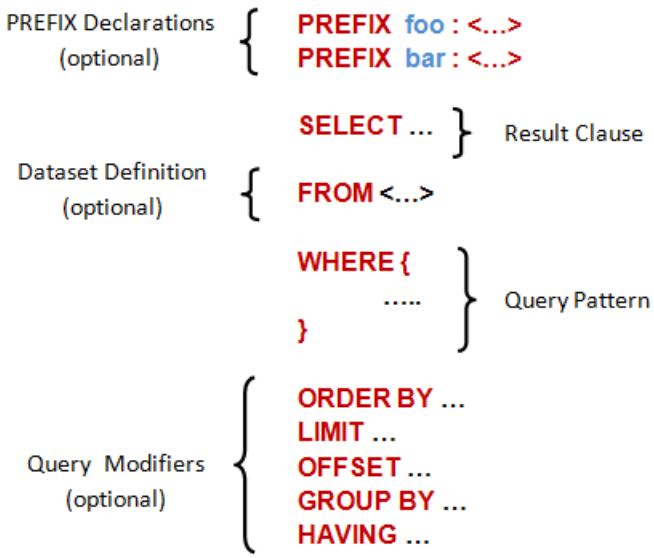


Figure 2.7: A SPARQL query structure

A SPARQL query can be divided into five sections (see [Harris et al. (2013)] for more details):

- PREFIX Declarations: They introduce the prefixes to be used to substitute the different namespaces. Indeed, the use of namespaces allows to abbreviate the URIs. This provides more clarity for reading a query triplet. The declaration of prefixes is optional. The syntax to declare a prefix is : *PREFIX prefix_name:<local_namespace>*. For example, the namespaces of RDF, RDFs, and OWL vocabularies:

  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
  PREFIX owl: <http://www.w3.org/2002/07/owl#>

- Result Clause: identifying what information to return from the query. A SPARQL query can take four forms:

  - *SELECT* form: Provides answers in a tabular form about the value of the requested variable(s). It like a SQL query executed against a relational database.
  - *ASK* form: To request the existence of a pattern in the RDF graph. The results is a boolean value (true, false).
  - *CONSTRUCT* form: It is similar to the *SELECT* form, but it allows to construct a new sub-graph from the specified constraints.
  - *DESCRIBE* form : Returns a single result RDF sub-graph containing RDF data about the requested resource.

  As in SQL, SPARQL query results may contain duplicates. Therefore, it is possible to ensure the uniqueness of the result with the use of *DISTINCT* and *REDUCED*. The modifier *DISTINCT* enforces that no duplicates are included in the query results. The modifier *REDUCED* permits the elimination of duplicates.

- Dataset Definition: defines the RDF graph(s) against which the query is executed. This part is optional. When nothing is specified, the default graph is selected.

- Query Pattern: specifying what to query for in the underlying dataset. This part is indicated by the *WHERE* clause which indicates the core of a SPARQL query. It is the declaration of the triplets, which correspond with the possible triplets of the graph. These triple patterns are used to select the triples composing the result.

- Query Modifiers: This part also optional. It allows to define modifiers which will act on the result of the request. These modifiers are *ORDER BY, HAVING, GROUP BY, LIMIT* and *OFFSET*. As in SQL, the clause *ORDER BY* orders the results set, the *GROUP BY* partition results into groups, the *HAVING* filter aggregated results, the *LIMIT* and *OFFSET* allow getting results in chunks.

SPARQL queries combine basic graph patterns into compound patterns with composition, optional or alternative parts. For instance, the *FILTER* clause adds constraints on

the variables. It uses boolean conditions to filter out unwanted query results. However, the *OPTIONAL* clause allows to define optionals conditions in the query. It tries to match a graph pattern, but does not fail the whole query if the optional match fails. In addition, SPARQL provides a means of combining graph patterns so that one of several alternative graph patterns may match. Pattern alternatives are syntactically specified with the *UNION* keyword.

The following SPARQL query shows an example querying the names, the emails of peoples. The *OPTIONAL* clause allow to return the age of people if it exists in the dataset. The *FILTER* condition indicates that If a solution has an age variable, then it must be greater than 25.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <http://example.org/schema/>

SELECT ?name ?mbox ?age
WHERE {
        ?x foaf:name ?name .
        ?x foaf:mbox ?mbox }

        OPTIONAL { ?x ex:age ?age . FILTER ( ?age > 25)}
        ORDER BY ?name
}
```

2.3 Sentiment Analysis

Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes [Liu (2010)]. Since early 2000s, sentiment analysis has grown to be one of the most active research areas in NLP. Because of the rise of the social Web, both the research and the industry are interested in automatic processing of opinions in text. In this section, we describe the common tasks of sentiment analysis and opinion mining. The plan is similar to the one proposed by [Liu (2010)]. We provide a definition of each task, common issues and main approaches.

2.3.1 Polarity classification

Polarity classification is probably the most studied subproblem of sentiment analysis. In general, this task is to determine whether a text expresses a positive or a negative attitude of its author towards the topic of the text.

The existing approaches to polarity classification fall into two large categories [Pang and Lee (2008)]:

Lexicon Based Approaches

A lexicon based approach uses some sort of an affective lexicon to derive the polarity of the examined text. These lexicons are usually composed of terms and the corresponding scores that represent the polarity of the terms. This approach uses string matching

techniques between texts and the annotated lexicon, and calculates the average score of the matched words.

The most common publicly available sentiment lexicons are WordNet-Affect [Strapparava et al. (2004)] and SentiWordNet [Baccianella et al. (2010)], which are the extensions of WordNet [Miller and Fellbaum (1998)]. WordNet-Affect is part of the WordNet Domains project[43] which assigns topical labels to WordNet synsets such as *sports, politics, medicine*. SentiWordNet was constructed by automatic annotation of WordNet synsets with three numeric scores representing positiveness (*Pos*), negativeness (*Neg*), and objectivity (*Obj*) by taking the advantage of graph-based model of the WordNet. SentiWordNet is evaluated by [Baccianella et al. (2010)] on a manually annotated dataset, and authors report that it can be used for sentiment analysis tasks.

In addition, we can cite the lexicon SenticNet [Cambria et al. (2010)], which is a knowledge-based extension of the above-mentioned lexicons. SenticNet is also evaluated by [Cambria et al. (2010)] on a manually annotated dataset, and the authors show that this lexicon can be used for sentiment analysis.

Statistical Based Approaches

A common approach for polarity classification is to train a supervised machine learning classifier over a set of given samples. The task is considered as a text classification problem. The most commonly used features are n-gram and POS tags, and the most commonly used classifiers are Support Vector Machine (SVM) [Cortes and Vapnik (1995)] and Naive Bayes (NB) [Pang and Lee (2008)].

In the work of [Pang et al. (2002)] on polarity classification task, authors use three machine learning algorithms (Maximum Entropy [Ratnaparkhi (1998)], NB, and SVM) and a set of features (e.g. uni-grams, bi-grams, POS tags) in order to classify movie reviews. The data are collected from the Internet Movie Database (IMDb)[44]. The authors reported that SVM algorithm outperforms the other algorithms, and a simple setup using uni-grams features with binary weights yielded the highest accuracy. The IMDb dataset was later used by other researchers thus producing comparable results ([Whitelaw et al. (2005)], [Matsumoto et al. (2005)]).

In [Pang and Lee (2004)], the authors proposed to use a 2-stage classification. The first stage classifies text into subjective and objective, and filters out the objective portion, while keeping the subjective portion for the second stage of classification, which classify the subjective texts into positive and negative. The idea is to prevent the polarity classifier from considering irrelevant texts. The authors showed that discarding objective texts improves the accuracy of polarity classification from 82.8% to 86.4%.

The authors in [Aly and Atiya (2013)] collected book reviews written in Arabic language from Goodreaders[45] social network. The corpus consists of about 63K book reviews. Each

[43] http://wndomains.fbk.eu
[44] http://www.imdb.com/reviews/index.html
[45] www.goodreads.com

review has a rating of 1 to 5 stars. The authors considered the reviews with rating 4 or 5 stars as positive and the reviews with 1 and 2 stars as negative. The authors used this corpus for polarity classification and rate prediction. Therefore, two machine learning classifiers were investigated on 1-gram, 2-gram, and 3-gram features. The results show that SVM classifier outperforms NB and the 3-gram are the best features.

Some researchers combine lexicon and statistical based methods to enrich the extracted features to improve the result of polarity classification [Dang et al. (2010)], [Hamouda and Rohaim (2011)]. The idea is to extract sentiment and some linguistic features, such as n-grams and POS tag, from the text and to add the sentiment score from the lexicon as a feature value. Then, classifiers are trained on these combined features to improve the accuracy of the classifiers.

In this thesis, we use the polarity classification task in order to correlate open rating system with the mentioned events in the text (Chapter 4), and also with the mentioned lived experience events in the text (Chapter 6, Section 6.1). Our work is different from [Aly and Atiya (2013)], who use n-gram features to classify the polarity of book reviews which are written in Arabic language. In our work, we use semantic features, such as events, quality, participant of events to classify the polarity of hotel reviews which are written in English language.

2.3.2 Subjectivity analysis

One of the common tasks of sentiment analysis is the subjectivity analysis. This task consists of detecting the presence of subjective sentences in the text. It determines what is subjective opinion (e.g. I like iPhone.), and what is objective one (e.g. Iphone is an Apple product). An *objective sentence* presents some factual information about the world, while a *subjective sentence* expresses some personal feelings, views, or beliefs [Liu (2012)].

Generally, subjectivity analysis is considered a binary classification problem: for a given text the system should return *true* if it contains subjective sentences and *false* otherwise. The common approach is to use a machine learning classifier trained on two sets of texts representing positive and negative samples. For example, the work of [Wiebe et al. (1999)] performed subjectivity classification using the Naive Bayes classifier with a set of binary features, e.g., the presence in the sentence of a pronoun, an adjective, a cardinal number, a modal other than *will* and adverb other than *not*. The results show that using these features, the classifier achieves an average accuracy 81.5 which is more than 20% higher than the baseline accuracy.

[Pang and Lee (2004)] augmented the polarity classification framework [Pang et al. (2002)] with an additional preprocessing step where sentences from an analyzed text are being classified as subjective or objective. The authors translated subjectivity classification into a graph-partitioning problem and used the min-cut max-flow theorem to solve it. The sentences labelled as "subjective" are extracted and passed to a general polarity classifier

(n-grams model with SVM). They reported a statistically significant improvement of the classification accuracy from 82.8 to 86.4.

[Benamara et al. (2011)] performed subjectivity classification with four classes, *S, OO, O,* and *SN*, where *S* means subjective and evaluative (their sentiment can be positive or negative), *OO* means positive or negative opinion implied in an objective, *O* means objective with no opinion, and *SN* mean subjective but non-evaluative (no positive or negative sentiment). The result of the classification showed that a subjective sentence may not be evaluative (with positive or negative sentiment) and an objective sentence can imply sentiment too.

[Abdul-Mageed et al. (2011)] developed a manually annotated text in Arabic language. Two human annotators were asked to annotate newspaper documents with objective, subjective-positive, subjective-negative labels. In their work, the author investigated 2-stage sentiment classification task on the corpus, where they first classify the text into subjective and objective, then the subjective text is classified into positive and negative.

In our work, we viewed the lived experience identification (Chapter 5) as a subjectivity classification task. However, our work consists of identifying user reviews which contain lived experience contents, a specific type of subjective text. Therefore, a corpus of user reviews was manually annotated by two classes indicating whether reviews include lived experience contents or not. Then, we employed a SVM classifier using semantic features such as lived experience events and their participants to perform the lived experience identification. This task is followed by an additional task consisting of extracting the lived experience contents from the reviews by integrating the Web Semantic and the natural language processing (NLP) techniques.

2.3.3 Aspect-based Sentiment Analysis (ABSA)

Although classifying opinionated texts is useful in many cases, it is often insufficient for many other applications because they do not identify opinion targets or assign sentiments to such targets. A positive opinion document about the entity does not mean that the author has positive opinions about all aspects of the entity. Evenly, a negative opinionated document does not mean that the author dislikes everything.

In a typical opinionated document, the author writes both positive and negative aspects of the entity, although the general sentiment on the entity may be positive or negative. Polarity or subjectivity classification does not provide such details. Therefore, to obtain these details, we need to discover the aspects and determine whether the sentiment is positive or negative on each aspect. This led us to discover a new type of research in sentiment analysis called *Aspect-based Sentiment Analysis (ABSA)* or *Feature-based Sentiment Analysis*

In ABSA, the opinion target is decomposed into entity (product, service, person, event, etc.) and its aspects (price, size, quality, location, etc.). Therefore, the goal of ABSA is to

associate the relevant sentiment traces to the right and explicit spots in the text with the correct polarity. Consequently, the ABSA task consists of two sub-tasks: (1) extracting aspects from the opinion text. Then, (2) for each extracted aspect, determine the associated sentiment (positive, negative, neutral).

[Hu and Liu (2004)] on their work of mining and summarizing customer reviews followed this approach. They identify aspects (features) that customers have expressed their opinions on using data mining algorithms. They considered that aspects are usually nouns or noun phrases in reviews sentences. In order to identify them, they used NLProcesser linguistic parser[46] to parse reviews and produce the part-of-speech tag for each word (whether the word is a noun, verb, adjective, etc). The occurrence frequencies for each aspect are counted, and only the frequent ones are kept. The next step of this work consists of identifying review sentences which contain one ore aspects and determining the sentiment for each sentence (positive or negative). For this purpose, authors firstly identified opinion word in each sentence containing one or more aspect. To facilitate this task, they only used adjectives as opinion word. Then, the sentiment of each opinion word was identified using WordNet [Miller et al. (1990)]. Thereafter, the orientation of the identified review sentences was predicted using opinion words and their sentiments.

[Long et al. (2010)] propose an approach to select review that talk about an aspect or service by using information distance of the review on the aspect. Therefore, they extracted aspects (nouns) based on frequency and information distance. Their method first finds the core aspect words using the frequency-based method. Other related words to an aspect have been also found using the information distance based on Google and proposed by [Cilibrasi and Vitanyi (2007)]. For example, for the aspect *price*, it may be find "$" and "dollars". All these words are then used to select reviews which discuss a particular aspect most.

Our work on Aspect-based sentiment analysis in Section 6.2 is close to the work of [Hu and Liu (2004)] on aspect extraction and sentiment classification. However, we used a machine reader to perform a deep semantic parsing of text in order to extract review aspects and their associated opinion words.

The main objective of the work of [Hu and Liu (2004)] is to summarize user reviews based on review aspects, opinion words, and the orientation of review sentences. In the following section (Section 2.3.4), we show in details this type of user reviews summarization.

2.3.4 Opinion Summarization

Automatic summarization is a well known problem in text analysis. According to [Radev et al. (2002)], a summary can be loosely defined as a text that is produced from one or more texts, that conveys important information in the original text(s), that is no longer

[46] http://www.infogistics.com/textanalysis.html

than half of the original text(s). The main idea of text summarization is to provide a shorter version of a text preserving its main idea.

[Radev et al. (2002)] and [Ganesan et al. (2010)] distinguish between two types of summarization:

- Extractive Summarization: This type of summary can be generated by identifying important material in the text. The summary can be created by reusing portions (words, sentences, etc.) of the input text verbatim.
- Abstractive summarization: This type is known to be difficult because it require regenerating the extracted content. The summary are created by reformulating the extracted information in novel term, fusion the process of combining extracted portions, and compression the process of squeezing out unimportant material.

These approaches have been studied in the literature. For instance, [Witbrock and Mittal (1999)] use statistical approach to choose important words and phrases and their syntactic context from news article and then order the word into sentences using a bigram language model. However, [Jing and McKeown (2000)] develop an automatic system to perform the cut-and-paste process used by humans to reduce the gap between automatically generated summaries and human-written abstracts. They extract sentences from article and transform its into the corresponding summary sentences in its human-written abstract.

Opinion summarization can be seen as a form of text summarization. Its main goal is to extract opinions from the text and present them in a shorter form. This task is critically needed to help users better digest the large amounts of opinions expressed on the web.

Opinion summarization has been studied by [Ganesan et al. (2010)]. Authors propose a summarization framework, Opinosis, that uses graphs to produce abstractive summaries of highly redundant opinions by using shallow NLP, leveraging mostly the word order in the text and its inherent redundancies. The results show that *Opinosis* summaries have better agreement with human summaries compared to the baseline extractive method. 60% of the generated sentences are no different from human composed sentences.

Researchers have also studied opinion summarization in the tradition fashion, e.g., producing a short textual summary based on multiple reviews or even a single review [Ku et al. (2006), Seki et al. (2006), Stoyanov and Cardie (2006)]. This text-based summary gives the reader a quick overview of what people think about a product or service. However, it may be suitable for human reading, but not suitable for analytical purpose because they are often only qualitative but not quantitative.

A common form of opinion summary is based on aspects, and is called *aspect-based opinion summary* (or *feature-based opinion summary*) [Hu and Liu (2004), Liu et al. (2005)]. In their work, authors mine and summarize customer reviews of a product by identifying features that customers have expressed their opinions on, identifying review sentences which contain one or more features, identifying the opinion orientation of each sentence, and then producing the summary. The proposed summary by [Hu and Liu (2004)] look

like that in Figure 2.8. In the figure, *picture quality* and *size* are the product features. 253 user reviews expressed positive opinions about the picture quality, and 6 expressed negative opinion. <individual review sentences> is a link pointing to the sentences and/or the whole reviews that give the opinions.

*Digital_camera_*1:
Feature: **picture quality**
Positive: 253
<individual review sentences>
Negative: 6
<individual review sentences>
Feature: **size**
Positive: 134
<individual review sentences>
Negative: 10
<individual review sentences>
...

Figure 2.8: An example summary [Hu and Liu (2004)]

This form of opinion summarization has two main characteristics: (1) it captures the essence of opinion (entities and their aspects) and sentiments about them. (2) it is quantitative, which means that it gives the number or percent of people who hold positive or negative opinions about the entities and aspects.

In our work for user review summarization in Section 6.2, we are interested in abstractive summary. Our objective is to generate an abstractive summary from user reviews based on a machine reader and sentiment analysis dictionaries. Our work is close to the work of [Hu and Liu (2004)] on aspect extraction and sentiment orientation. However, we extract review aspects and their opinion words by employing a deep semantic parsing of text. We used three sentiment lexicons (SentiWordNet [Baccianella et al. (2010)], AFINN[47], and [Liu (2012)] to detect the polarity of the opinion words. In addition, we proposed to regroup similar aspects using WordNet:Similarity [Pedersen et al. (2004)] to generate the abstractive summary. Furthermore, our summary representation is different from the representation model proposed by [Hu and Liu (2004)] (See Figure 2.8). Our summary consists of a set of quadruple (Aspect, Attribute, Frequency, Polarity) indicating the extracted aspects, their adjectives, their frequencies, and the associated sentiment.

2.3.5 Opinion Spam Detection

Public opinion do not only serve as feedback for product manufacturers (service providers, political parties, etc.), but they also influence other people's decision when choosing a product or a service. Hence, many parties are interested in publishing positive opinions about themselves and negative opinions about their competitors. Since the Web provides

[47] https://github.com/abromberg/sentiment_analysis/tree/master/AFINN

many ways to express opinion without any means of verification of their trustworthiness, a new research direction, called identification of opinion spam, has been created. The objective of this research is to detect such spamming activities to ensure that the opinions on the Web are a trusted source of valuable information.

According to [Jindal and Liu (2008)], there are three types of spam reviews:

- **Type 1 (untruthful reviews):** These are fake reviews that are written not based on the user experiences of using products or services, but are written with hidden motives. They often give undeserving positive reviews to some target entities (products or services) in order to promote entities and/or give unjust or malicious negative reviews to some other entities in order to damage their reputations.
- **Type 2 (reviews on brand only):** These reviews do not comment on the products or services that they are supposed to review, but only comment the brand, the manufacturers, or the seller of the products. These reviews are considered as spam since they are not targeted at the specific products and are often biased.
- **Type 3 (untruthful reviews):** These are non-reviews. They have two main subtypes: (1) advertisements and (2) other irrelevant reviews containing no opinions (e.g., question, answers, and random text).

The main objective of opinion spam detection in review context is to identify every fake review, fake reviewer, and fake reviewer group. This task can be formulated, in general, as a classification problem with two classes, *spam* and *non-spam*. This requires the annotation of a set of reviews with the two labels mentioned above. However, manually labeling the dataset for learning is very hard, if not impossible. Therefore, researchers tried to find training examples for detecting possible spam reviews.

In the work of [Jindal and Liu (2008)], authors first re-framed the problem as one of trying to recognize duplicate reviews, since a priori it is hard to see why posting repeats of reviews is justified. Therefore, they applied heuristic to extract untruthful reviews from Amazon[48]. To collect such reviews, authors performed an analysis of 5.8 million reviews from 2.14 million reviewers and 6.7 million products to find authors posted similar repetitive comments about products which were considered as opinion spam.

To build a gold standard, [Ott et al. (2011)] composed a collection of fake reviews about hotel to be used in addition to a set of filtered reviews collected from TripAdvisor[49] which were considered as truthful. A part of the constructed gold standard was given to three volunteers to perform an analysis of human performance at identifying deceptive opinions. These results were compared to performance of machine learning approach that used n-grams and features produced by Linguistic Inquiry and Word Count (LIWC)[50] software

[48] `http://www.amazon.com`

[49] `http://tripadvisor.com`

[50] a text analysis software program which calculates the degree to which people use different categories of words in texts `http://www.liwc.net`

developed by [Pennebaker et al. (2007)]. Their experiments showed that text classification performed the best using only unigram and bigrams based on the 50/50 spam and non-spam class distribution. In addition, the comparison revealed that even a simple n-grams based classifier performs better than human experts at classifying reviews into deceptive and truthful. Moreover, human biased towards the truthful decision (i.e. labeling most of the reviews as truthful), while a classifier based on a combination of n-grams and LIWC features yielded up to 89.9 of F-Measure estimated by 5-fold cross validation.

Chapter 3

FRED as an Event Extraction System

Contents

3.1 Introduction 41
3.2 FRED: A machine reader tool 42
3.2.1 From Natural Language into DRT Languages 42
3.2.2 From DRS Form to RDF/OWL ontologies 43
3.2.3 Other FRED Components 44
3.3 FRED as Event Extraction Tool 45
3.4 FRED Quality & Importance 49
3.5 Conclusion 52

In this chapter, we present FRED, a machine reader tool that automatically generates RDF/OWL ontologies and linked data from natural language text. Then, we present how we employ this tool in order to extract events from text.

3.1 Introduction

Extracting, logically representing, and connecting elements from a sentence is crucial to create semantic applications that are event-aware. In addition, it's important to disambiguate as much as possible the entities and concepts expressed, in order to make the extracted model *linked*, and to exploit the full power of the Semantic Web and Linked Data.

Machine readers have been introduced by [Etzioni et al. (2006)] as tools for text understanding. They combine different text analysis layers (Part-Of-Speech tagging, syntactic analysis, disambiguation, named entity recognition, event detection) to produce a rich semantic representation of the text.

FRED[1] [Gangemi et al. (2016),Presutti et al. (2012)] is a tool that automatically generates RDF/OWL ontologies and linked data from mutlilingual natural language text based on Combinatory Categorial Grammar [Steedman (2000)], Discourse Representation Theory [Kamp (1981)], Linguistic Frame Semantics [Fillmore (1982)], and Ontology Design

[1] http://wit.istc.cnr.it/stlab-tools/fred

Patterns [Gangemi and Presutti (2009)]. It is a machine reader for the Semantic Web. It is able to produce such a formal knowledge representation, specifically for the Semantic Web. This feature adds to the paradigm of *machine reading* the ability to interpret the generated graphs by machines, according to a shared formal semantics.

3.2 FRED: A machine reader tool

FRED is a *machine reader* [Etzioni et al. (2006)] for the Semantic Web. It extracts knowledge (named entities, senses, taxonomies, relations, events) from text, resolves it onto the Web of Data, adds data from background knowledge, and represents all that in RDF and OWL.

FRED performs the generation of the semantic graphs in two main steps: *(1)* Transform natural language text into a logical form. It performs a deep semantic parsing of text and extracts complex relations between the elements of the text using Boxer [Bos (2008)]. *(2)* Transform the generated logical form into RDF/OWL graphs. For this step, FRED defines and uses a set of translation and heuristic rules which allows to achieve this transformation. Each of this step will be more explained in in the following.

3.2.1 From Natural Language into DRT Languages

The first step of FRED consists of transforming the input text into Discourse Representation Theory (DRT) form [Kamp (1981)]. DRT is a formal theory of meaning originally described in [Kamp (1981)], and is equivalent to first-order logic (FOL). DRT uses an explicit semantic structured language called Discourse Representation Structure (DRS), a standard representation corresponding to natural language sentences. DRT provides an event-based, Neo-Davidsonian [Kamp (1981)] (based on reified n-ary relations just as frames are) model to represent natural language.

In order to produce DRS output, FRED performs deep parsing of natural language text and extracts complex relations using Boxer [Bos (2008)], an implementation of compositional semantics of language. Boxer[2] is open-source software that performs deep parsing of natural language using Combinatory Categorial Grammar (CCG) parse tree [Steedman (2000)]. The output of Boxer is event-based, verb-centric, semantic representations of natural language complying with DRT semantics. These representations are expressed in the form of DRS using both VerbNet [Schuler (2005)][3], and FrameNet [Baker et al. (1998)] for frame labelling and semantic role labelling, i.e. the representation of event types and the relations between events and their participants.

In fact, VerbNet [Schuler (2005)] allows to identify the roles involved in a sentence, and FrameNet [Baker et al. (1998)][4] uses these roles to detect a corresponding frame. The

[2] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer

[3] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

[4] FrameNet is a lexical resource that collects linguistic frames, each described with its semantic roles, called frame elements, and lexical units (the words evoking a frame)

detection of these semantic frames [Coppola et al. (2009)] allows to recognize the complex relation in natural language text. However, the generated FOL modeling style with boxer is not always compatible with Semantic Web and Linked Data design. Therefore, [Presutti et al. (2012)] adopted Boxer for tackling the frame detection task by integrating it with a resource that provides the most completed mapping between VerbNet and FrameNet. Figure 3.1 shows the output of Boxer for the sentence *"People love movies"*.

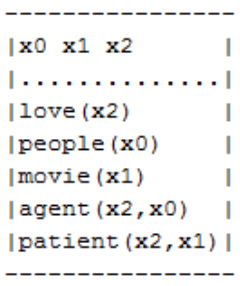


Figure 3.1: Boxer output for the sentence: *People love movies.*

As shown in this figure, the box is divided into two sections, the top section contains the discourse referents *x0, x1, x2*; the bottom section contains the predicates that constrain their interpretation: *x2* is an event described by the term *love*, which have two arguments, an agent *x0*, of type *people* and a patient *x1*, of type *movie*.

3.2.2 From DRS Form to RDF/OWL ontologies

The second step of FRED consists of transforming the logical output of Boxer with frames into RDF/OWL ontologies. This task has been achieved using a mapping model and a set of heuristics that follow good practices of OWL ontologies and RDF data design. For this task, [Presutti et al. (2012)] define two types of transformation rules:

- Translation rules: indicate the rules which define the global transformations from DRS constructs to OWL constructs. Table 3.1 shows the main translation rules which defined by [Presutti et al. (2012)] and used to transform DRS to OWL. It indicates DRT constructs, their syntax in Boxer, their corresponding FOL, and their corresponding OWL/RDF construct. In addition, Table 3.2 presents the most frequently built-in predicates used in Boxer with their associated Semantic Web entity.

DRT construct	Boxer syntax	FOL construct	OWL construct
Predicate	pred(x)	Unary predicate Φ	rdf:type
Relation	rel-name(x,y)	Binary relation	owl:ObjectProperty
Eq Rel	eq(x,y)	Identity	owl:sameAs
Named Entity	named(<var>, <name>, <type>)	Unary predicate Φ	owl:NamedIndividual
Discourse Referent	(<var>)	Quantified Variable	(generated) owl:NamedIndividual
DRS	<drs> with event E	Proposition P with predicate Φ_E	owl:NamedIndividual
Negated DRS	not(<drs>)	Negated Proposition $\neg$P	G_P with $NotE$ owl:disjointWith E

Table 3.1: The main translation rules from DRS to OWL [Presutti et al. (2012)].

Boxer built-in type	Label	Semantic Web entity
Per	Person	foaf:Person
Org	Organisation	foaf:Organisation
Loc	Location	dbpedia:Place
Tim	Time	to:Interval
Ttl	Title	dul:Role
Event	Event	dul:Event
Eq	Equal to	owl:sameAs

Table 3.2: Boxer built-in types and relations [Presutti et al. (2012)].

Based on the mappings reported in Table 3.1 and Table 3.2, the tool FRED generates the RDF graph presented in Figure 3.2 for the sentence *"People love movies"*. Events are modelled as subtypes of the class *dul:Event*. The referent *x0* becomes *fred:people_1*, as well as alignment of predicates to existing semantic web technologies, e.g. with *owl:equivalentClass*. The event type *fred:Love* is aligned to *vn.data:Love_31020100*, and therefore, roles are labeled as *vn.role:Experiencer* and *vn.role:Theme*.

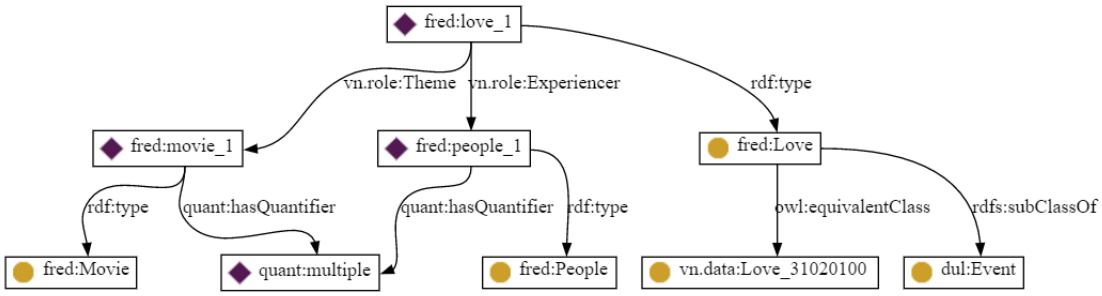


Figure 3.2: FRED output for the sentence: *People love movies.*

- Heuristic rules: indicate the rules which define local transformations that deal with adapting the results of Boxer heuristics to the needs of a Semantic Web ontology.

  FRED implements many heuristic rules such as, heuristic rule based on co-reference, heuristic rule to do with naming, heuristic rule that creates individuals with a generating name to existentially quantified variables that are not resolved as named entities, and other heuristic rules to do with the generation of terminology associated with the definition of appropriate classes or properties.

3.2.3 Other FRED Components

Besides Boxer, the FRED tool leverages multiple natural language processing (NLP) components by integrating their outputs into a unified result, which is formalised as an RDF/OWL graph. The developers of FRED found that Boxer's *pronoun* CRR capabilities are limited, Therefore, they integrated CoreNLP[5] as an additional component for this

[5] http://nlp.standford.edu/software/corenlp.shtml

specific task. In addition, FRED exploits external services to enrich RDF/OWL results with Named Entity Resolution (NER) and Word-Sense Disambiguation (WSD). For NER, FRED uses Stanbol enhancers[6]. Stanbol allows to indicate any number of datasets to be used as sources for entity recognition and resolution, hence providing great flexibility and customizability with respect to the entities that are of interest for a FRED user. In addition, FRED also integrates TAGME [d'Aquin et al. (2008)], which use Wikipedia content as context to disambiguate named entities.

For WSD, which is used for producing alignments with classes from external ontologies, the tool FRED reuses the UKB tool [Agirre and Soroa (2009)][7], by aligning domain classes to WordNet synsets, and the synsets to DOLCE+DnS foundational ontology[8] and WordNet lexnames ("super-senses")[9].

3.3 FRED as Event Extraction Tool

As we have shown above, FRED takes as input the Discourse representation Structures (DRS) produced by Boxer. The output of Boxer is event-based, verb-centric, semantic representations of natural language complying with the Discourse Representation Theory (DRT) semantics. Therefore, FRED is event-centric and it natively supports event extraction.

FRED is available as a RESTful API and as a web application. In its current form, it relies upon several NLP components: Boxer[10] for the extraction of the basic logical form of text and for disambiguation of events to VerbNet[11], UKB[12] or IMS[13] or BabelNet API[14] for word sense disambiguation, and Apache Stanbol[15] for named entity resolution.

FRED contains several functionalities for event extraction, which can be summarized according to typical subtasks:

- Event identity: FRED focuses on events expressed by *verbs*, *propositions*, *common nouns*, and *named entities* (typically proper nouns).
- Event classification: FRED uses Linked Data-oriented induction of types for the identified events, reusing e.g. VerbNet, WordNet[16], DBpedia[17], schema.org, and DOLCE[18] as reference ontologies.

[6] http://incubator.apache.org/stanbol/
[7] http://ixa2.si.ehu.es/ukb/
[8] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[9] http://wordnet.princeton.edu/man/lexnames.5WN.html
[10] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer
[11] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html?
[12] http://ixa2.si.ehu.es/ukb/
[13] http://www.comp.nus.edu.sg/~nlp/sw/
[14] http://lcl.uniroma1.it/babelnet/
[15] http://stanbol.apache.org
[16] http://wordnet.princeton.edu
[17] http://dbpedia.org
[18] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

- Event unity: FRED applies *semantic role labeling* [Moschitti et al. (2008)] to verbs and propositions in order to detect event boundaries, and *frame detection* [Coppola et al. (2009)] for resolving roles against a shared event ontology. In other words, FRED makes use of both VerbNet [Schuler (2005)] and FrameNet [Baker et al. (1998)] for representing event types and the relation between events and their participating individuals.

- Event modifiers: FRED extracts *logical negation*, *basic modalities*, and *adverbial qualities*, applied to verbs and propositions, which can then be used as event judgment indicators.

- Event relations: FRED relates events via the role structure of verbs and propositions, and extracts *tense relations* between them.

The following sentence is used as a lead example for showing FRED's functionalities:

The Renaissance was a cultural movement that spanned in Italy from the 14th to the 17th century. Some sources report that the Renaissance might have been started by Greek scholars from Constantinople.

The figure 3.3 show the FRED's output diagram for the mentioned sentence above. This diagram depict a subset of the generated triples, which cover the core semantics of the text. In the diagram from this figure, the following events are recognized, extracted, classified, and aligned to WordNet, VerbNet, and/or DOLCE: `Renaissance` (classified as a `Movement`, and aligned to the WordNet `Motion` synset, and to the DOLCE `Event` class), `span_1`, `report_1`, and `start_1` (classified as occurrences of the `Span`, `Report` and `Start` frames respectively, and aligned to VerbNet).

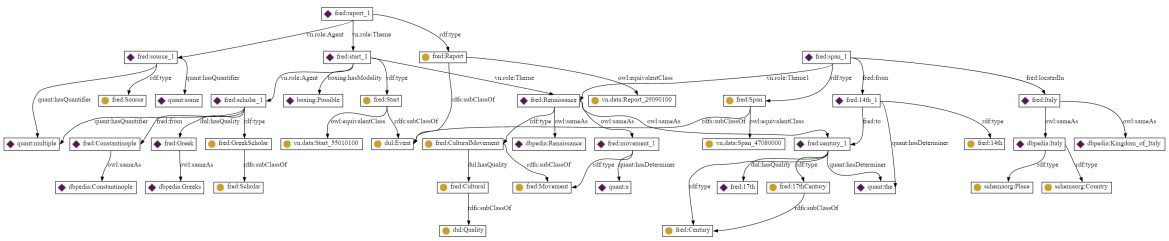


Figure 3.3: A FRED graph depicting the core subset of triples representing event-related knowledge.

Furthermore, the events have participants (e.g. `Italy`, `scholar_1`, `source_1`, etc., also classified and linked appropriately) through some roles labelled with properties derived from VerbNet(e.g. `vn.role:Agent, vn.role:Theme`), or from the lexicon used in the sentence (e.g. `fred:from`). In one case, a modal modifier (`Possible`) to the event `start_1` is added.

In order to well understand, we show by the following a subset of the generated triples by FRED for the events which are detected from the mentioned sentences above.

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"

fred:span_1
        fred:from            fred:14th_1 ;
        rdf:type             fred:Span ;
        vn.role:Theme1       fred:Renaissance ;
        fred:locatedIn       fred:Italy .

fred:report_1
        vn.role:Theme        fred:start_1 ;
        vn.role:Agent        fred:source_1 ;
        rdf:type             fred:Report .

fred:start_1
        rdf:type             fred:Start ;
        vn.role:Agent        fred:scholar_1 ;
        vn.role:Theme        fred:Renaissance ;
        boxing:hasModality   boxing:Possible ;

fred:Span
        rdfs:subClassOf      DUL:Event ;
        owl:equivalentClass  vn.data:Span_47080000 .

fred:Report
        rdfs:subClassOf      DUL:Event ;
        owl:equivalentClass  vn.data:Report_29090100 .

fred:Start
        rdfs:subClassOf      DUL:Event ;
        owl:equivalentClass  vn.data:Start_55010100 .
```

The text are represented as an RDF/OWL n-ary relation modelling the identified events, with their arguments modelled as typed individuals, and the semantic roles modelled as object properties.

From these triples, the events `span_1`, `report_1`, and `start_1` are detected as a subtype of the `Span`, `Report`, and `Start` event types respectively. These event types are modelled as subtype of the class *DUL:Event*, using the ontology *DOLCE*, and are aligned to `vn.data:Span_47080000`, `vn.data:Report_29090100`, and `vn.data:Start_55010100` respectively, defined in VerbNet [Schuler (2005)]. The relations between these event types and their arguments are modelled as object properties according to the semantic roles that are recognized using VerbNet, e.g. `vn.role:Agent` and `vn.role:Theme`. In case additional roles are detected but not recognized, FRED creates new (role) object properties and labels them by reusing the appropriate text from the input, e.g. `fred:from`, `fred:locatedIn`.

FRED also represents modality in its unified OWL/RDF graph, by identifying the corresponding patterns in Boxer output. Modality in FRED can be of two types: `boxing:Necessary` which corresponds to forms such as (*will, should, must, etc.*) and `boxing:Possible` for forms such as (*may, might, etc.*). Both are individuals of the

nominal class `boxing:Modality`. In our example, the event `start_1` has the modality `boxing:Possible` which was indicated using the property `boxing:hasModality` and formalise the *"might have been started"* fragment.

In addition, some relations between events are detected: `report_1 vn.role:Theme start_1`, and `span_1 before report_1`.

Beside, negation could be detected by FRED using the corresponding patterns of Boxer. FRED annotates the identified event with the information that its truth value is false using the property `boxing:hasTruthValue`. Figure 3.4 shows the output diagram of FRED for the following sentence (See Chapter 2, Section 2.1.1):

The Black Hand might not have decided to barbarously assassinate Franz Ferdinand after he arrived in Sarajevo on June 28th, 1914.

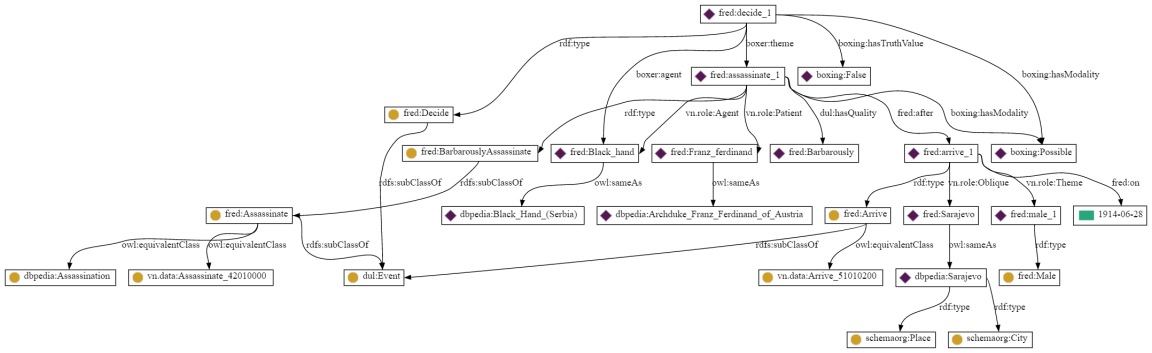


Figure 3.4: A diagram showing the FRED graph for the *Black Hand* sentence.

In this diagram, the event `decide_1` has a truth value "False" which indicates a negation (not) modifying this event.

Finally, FRED represents such modifiers (i.e. adjectives) as qualities of the modified event by means of the DOLCE property `dul:hasQuality`. Our example in the Figure 3.4, indicates that the event `assassinate_1` has a quality `Barbarously`, which was indicated by the predicate `dul:hasQuality`.

By the follows, we show the generated RDF triples by FRED for two events `decide_1`, and `assassinate_1`, respectively.

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

fred:decide_1
        boxing:hasTruthValue     boxing:False ;
        boxer:theme              fred:assassinate_1 ;
        boxer:agent              fred:Black_hand ;
        boxing:hasModality       boxing:Possible ;
        rdf:type                 fred:Decide .
```

```
fred:assassinate_1
        boxing:hasModality       boxing:Possible ;
        rdf:type                 fred:BarbarouslyAssassinate ;
        vn.role:Agent            fred:Black_hand ;
        fred:after               fred:arrive_1 ;
        dul:hasQuality           fred:Barbarously ;
        vn.role:Patient          fred:Franz_ferdinand .

fred:Decide
        rdfs:subClassOf          DUL:Event .

fred:BarbarouslyAssassinate
        rdfs:subClassOf          fred:Assassinate .

fred:Assassinate
        rdfs:subClassOf          DUL:Event .
```

The triples given as output by FRED are more than those visualized, for example they include text spans and their reference to the semantic annotations, through the Earmark [Peroni et al. (2011)], the NLP Interchange Format (NIF) [Hellmann et al. (2013)], and semiotics.owl[19] vocabularies. FRED provides annotations that link text fragments to their corresponding graph elements. Text fragment are represented as offsets (e.g. fred:offset_30_37_decided).

FRED will be demoed as an event extractor by showing event-intensive sentences, and examples of views that focus on relevant event knowledge. RDF models can be morphed to concentrate on specific features. For example, Figure 3.5 semantically summarizes the model from the *Black Hand* sentence by only showing events with their relations, and their main participant, obtained by means of the following SPARQL query:

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>
PREFIX boxer: <http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl#>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
CONSTRUCT {?e :agent ?x . ?e ?r ?e1}
WHERE {
 {{?e a boxing:Situation} UNION {?e a ?class . ?class rdfs:subClassOf+ dul:Event}}
 ?e ?p ?x
 FILTER (?p = vnrole:Agent || ?p = boxer:agent || ?p = vnrole:Experiencer || ?p = vnrole:Actor
     || ?p = vnrole:Actor1 || ?p = vnrole:Actor2 || ?p = vnrole:Theme)
 FILTER NOT EXISTS {?e vnrole:Theme ?x . ?e vnrole:Agent ?y
     FILTER (?x != ?y)}
 OPTIONAL {{{?e ?r ?e1} UNION {?e ?s ?z . ?z ?t ?e1}} {{?e1 a boxing:Situation} UNION
    {?e1 a ?class1 . ?class1 rdfs:subClassOf+ dul:Event}} FILTER (?e != ?e1)}}
```

3.4 FRED Quality & Importance

The importance of FRED comes from its ability to perform specific knowledge extraction tasks such as event detection [Hogenboom et al. (2011)], named entity recognition [Nadeau and Sekine (2007)] and resolution [Bhattacharya and Getoor (2007)], taxonomy induction

[19] http://www.ontologydesignpatterns.org/cp/owl/semiotics.owl

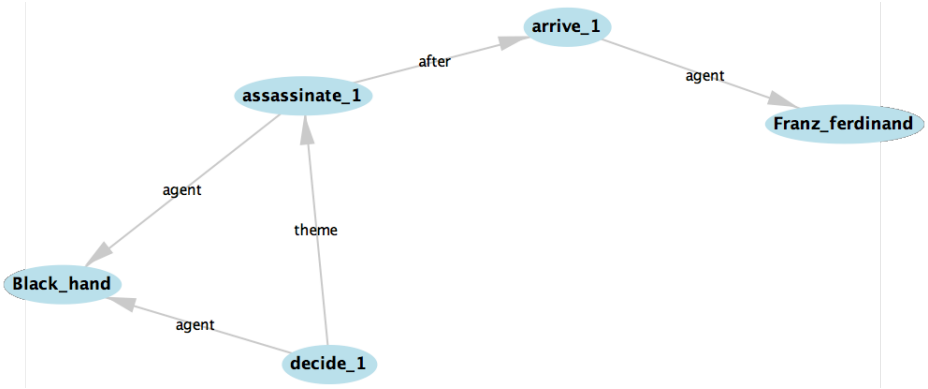


Figure 3.5: A summarized FRED graph showing only event relations and agentive participants for the Black Hand sentence.

[Ponzetto and Strube (2011)], sense tagging [Ciaramita and Altun (2006)] and disambiguation [Navigli (2009)], terminology extraction [Hartmann et al. (2012)], relation extraction [Ciaramita et al. (2005)], [Banko et al. (2008)], semantic role labeling [Moschitti et al. (2008)], and frame detection [Coppola et al. (2009)].

FRED has been compared with other information extraction tools used for Semantic Web tasks [Gangemi (2013)]. The author defines a number of basic semantic tasks by providing a correspondence between NLP tasks and semantic web terminology. Table 3.3 reports the list of tasks with a brief explanation of such correspondences: each NLP task is informally associated with a corresponding OWL-based semantics by indicating the type of triples that may be produced starting from its output. The study compares fifteen tools (including FRED) by assessing their coverage of, and performance on, the listed tasks against a gold standard of 524 triples produced from a news text. The result shows that FRED has the largest coverage of tasks and best accuracy performance for some of them.

Task	NLP terms	Semantic Web triples
TopE	Topic Extraction	dc:subject
NER	Named Entity recognition	owl:NamedIndividual
NEReS	Named Entity resolution	owl:sameAs
TE	Terminology extraction	owl:Class || owl:ObjectProperty || owl:DatatypeProperty
TReS	Terminology resolution	owl:equivalentTo || rdfs:subClassOf || rdfs:subPropertyOf
Senses	Sense tagging	rdf:type
Tax	Taxonomy induction	rdfs:subClassOf
RE	Relation Extraction	owl:ObjectProperty || owl:DatatypeProperty
Events	Event detection and SRL	<Event> rdf:type <Event.type> . <Event> $<semrole_i>$ $<Entity_j>$
Roles and Frames	Frame detection	<Event.type> rdfs:subClassOf <Frame>

Table 3.3: Summary of basic semantic tasks [Gangemi (2013)].

Table 3.4 shows the results of the evaluation. It reports a value for each semantic task indicating the accuracy performance for each of them. The "−" sign indicates that task not computed (i.e. task not addresses) while the "+" sign indicates that the task computed (addressed). See [Gangemi (2013)] for a detailed description of each tool and also for details about performance measures.

[20] http://www.mpi-inf.mpg.de/yago-naga/aida/

Tools	TopE	NER	NEReS	TE	TReS	Senses	Tax	RE	Events	Roles and Frames
AIDA[20]	–	.89	**.80**	–	–	.64	–	–	–	–
Alchemy[21]	**.52**	.89	–	.20	–	.64	–	.30	–	–
Apache Stanbol[22]	–	.77	.25	–	–	.50	–	–	–	–
CiceroLite[23]	–	.89	.75	.21	.07	.64	–	.25	.18	.22
DB Spotlight[24]	–	.79	.55	–	–	.42	–	–	–	–
FOX[25]	–	.86	.75	.33	**.65**	.57	–	–	–	–
FRED	–	.84	.60	**.90**	.07	.48	+	**.82**	**.87**	**.69**
NERD[26]	–	.88	.60	–	–	.69	–	–	–	–
Open Calais[27]	.48	.82	–	–	–	.57	–	–	.04	–
PoolParty KD[28]	.28	–	–	–	–	–	–	–	–	–
ReVerb[29]	–	–	–	–	–	–	–	.27	–	–
Semiosearch[30]	–	–	.60	–	.46	–	–	–	–	–
Wikimeta[31]	–	.86	.75	.04	.07	**.80**	–	–	–	–
Zemanta[32]	–	**.93**	–	–	–	.27	–	–	–	–

Table 3.4: Summary of evaluation results for basic tasks indicating accuracy values in the interval [0,1] with 1 expressing the best possible accuracy [Gangemi (2013)].

In addition, FRED has been also compared with Semafor [Das et al. (2010)] for the frame detection task. The results show that FRED is much faster than Semafor [Presutti et al. (2012)] (See Figure 3.6). Table 3.5 summarizes the accuracy performance of the two tools. The results show that FRED and Semafor have comparable precision, while the value of recall is lower for FRED.

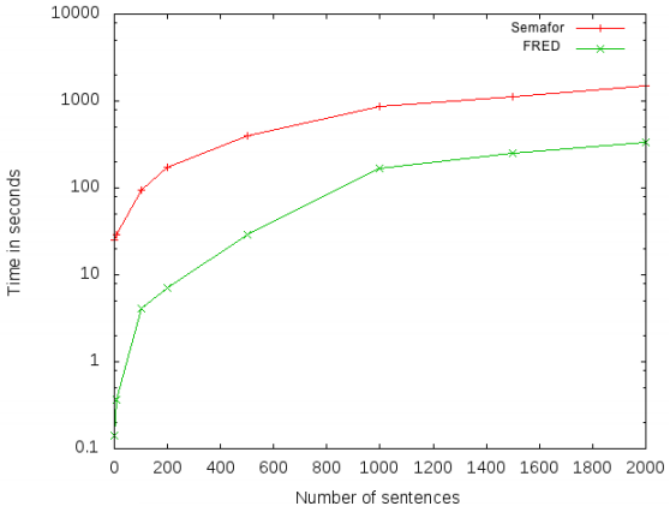


Figure 3.6: Time to provide answers in function of the number of sentences per document as reported in [Presutti et al. (2012)].

[21] http://www.alchemyapi.com/api/demo.html
[22] http://dev.iks-project.eu:8081/enhancer
[23] http://demo.languagecomputer.com/cicerolite
[24] http://dbpedia-spotlight.github.io/demo/
[25] http://aksw.org/Projects/FOX.html
[26] http://nerd.eurecom.fr/
[27] http://viewer.opencalais.com/
[28] https://drupal.poolparty.biz/
[29] http://reverb.cs.washington.edu/
[30] http://wit.istc.cnr.it/stlab-tools/wikifier
[31] http://www.wikimeta.com/wapi/semtag.pl
[32] http://www.zemanta.com/blog/

Tool	Precision	Recall	F-Score
FRED	75.320	57.519	65.227
Semafor	75.325	74.797	75.060

Table 3.5: The performance comparison between FRED and Semafor on Frame detection task [Presutti et al. (2012)].

Besides, FRED is available as a web application, as a RESTful service, as well as Python API (*fredlib*[33]), which relies on the k∼ore REST services, and allows to query FRED with a user-specified corpus, also enabling the manipulation of the resulting graph. Furthermore, FRED offers a several output results such as graphical graph, as shown in Figure 3.7, RDF/XML, Turtle, RDF/JSON, etc. (See Section 2.2.2 for more details about the syntax of these representations).

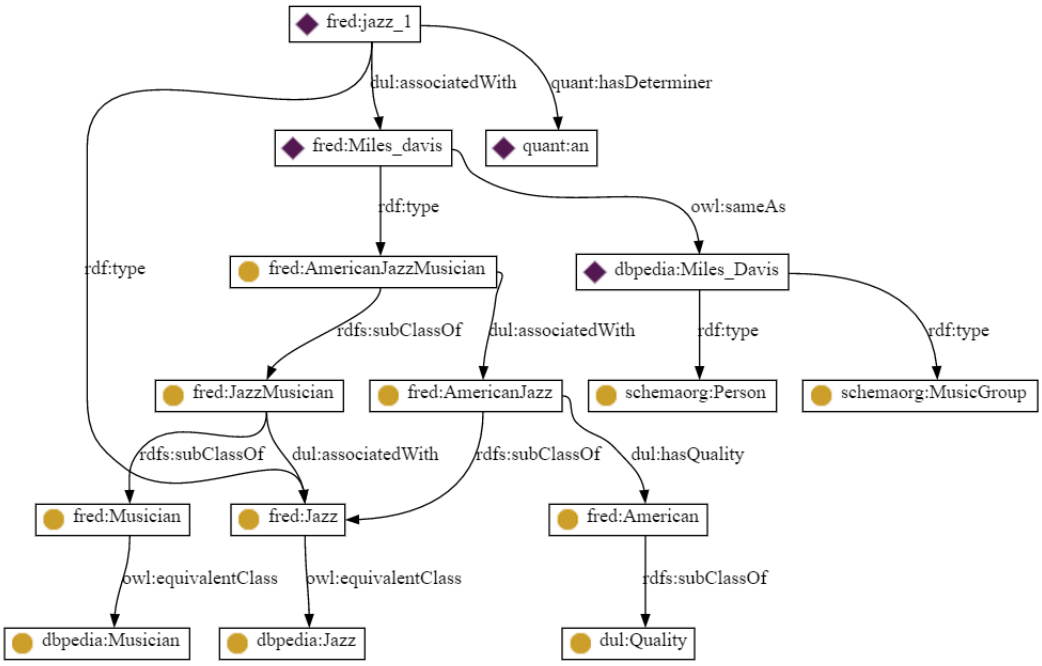


Figure 3.7: The graphical output of FRED for the sentence: *Miles Davis was an american jazz musician.*.

3.5 Conclusion

In this chapter, we presented the machine reader tool FRED. It is a tool to automatically transforms the input text into RDF/OWL knowledge graph representation of the text. FRED is event-centric, therefore, it supports event extraction task. It also supports multi event detection sub-tasks such as event classification, event modifiers detection, and event participants detection. Therefore, it is an intermediate component for event extraction and

[33] http://wit.istc.cnr.it/stlab-tools/fred/fredlib

representation, which can be augmented with background knowledge, and whose graphs can be combined e.g. in time series for historical tasks.

In the following chapters, we use FRED as a semantic middleware in order to perform a deep semantic parsing of user reviews and represent them as RDF/OWL knowledge graphs. These semantic graphs could be then queried in order to extract such information such as events, named entities, relations, etc. Our choice of FRED is based on its capabilities to detect events in the open domain. In addition, in a recent landscape analysis of knowledge extraction tools [Gangemi (2013)], FRED has got 0.73 precision, 0.93 recall, and 0.87 accuracy, largely better than the other tools attempting event extraction.

The work presented in this chapter has resulted in the following publication:

- Aldo Gangemi, Ehab Hassan, Valentina Presutti, and Diego Reforgiato. Fred as an event extraction tool. In Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2013) at 12th International Semantic Web Conference, ISWC 2013, Zurich, Switzerland, page 14, 2013.

Chapter 4

From events to Lived Experiences

Contents

4.1 Introduction 55
4.2 Our Approach 57
4.2.1 User Review 58
4.2.2 Event Extraction 58
4.2.3 Classification Methods 66
4.3 Experiments 68
4.3.1 DataSet 69
4.3.2 Development of the Review Classifier 69
4.3.3 Evaluation 73
4.3.4 Error Analysis 75
4.3.5 σ Estimation 76
4.3.6 BaseLine 76
4.3.7 Application to ESWC2014 challenge 78
4.4 Conclusion 79

In this chapter we present the work carried out on event extraction from user reviews. Our objective was to verify if events, as extracted by FRED, may be used as clues to predict the polarity of a score assigned by a user. We measured the correlation between user review contents and the rankings which are attributed by the user. In Section 4.1, we provide some background on open rating systems. In Section 4.2, we present our approach to extract events and build the classifiers. In Section 4.3, we report the results about our experiments in correlating events with ratings.

4.1 Introduction

With the explosion of Web 2.0, platforms such as blogs, discussion forums, peer-to-peer networks and various other types of social media allow people to express themselves and interact with others. They can post reviews of products and services in merchant websites and express their opinions through these platforms. Online reviews are often the primary factor in a customer's decision to purchase a product or service, and are a valuable source of information that can be used to determine public opinion on these products or services.

Open rating systems tackle the problem of setting ratings on a large and rapidly growing body of content by using an open system for the expression of these ratings, i.e., anyone can publish ratings [Guha (2003)]. They allow to synthetically grasp the opinion of the crowds with reference to specific entities: products, services, statements of ideas, etc. This concept has proven highly successful and is currently employed at several Web sites such as Amazon[1], Booking.com[2], Epinions[3], iTunes[4], TripAdvisor[5], and many other sites.

When an opinion is given by both synthetic ranking and a review, one may wonder e.g. (as shown in Figure 4.1):

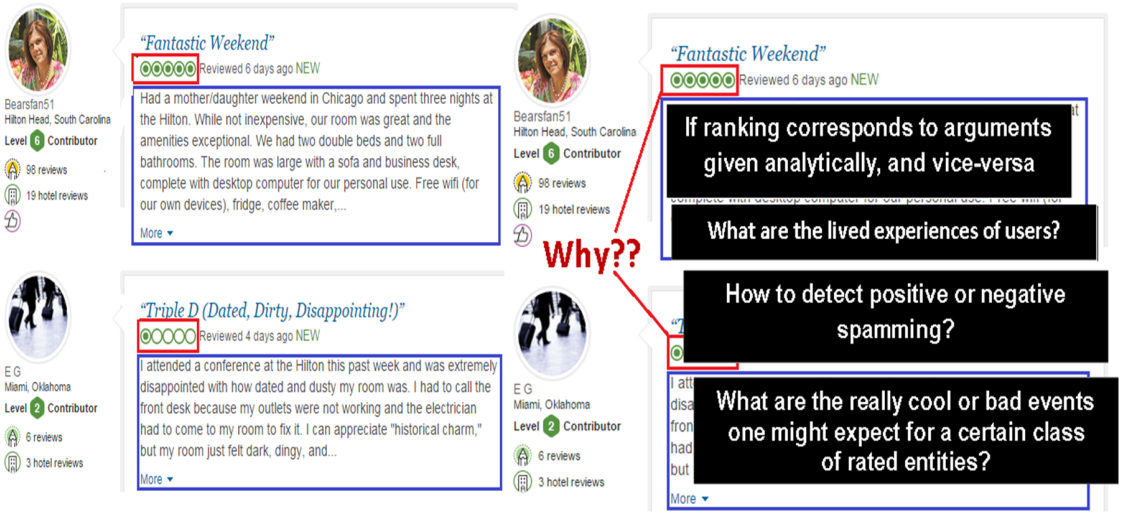


Figure 4.1: Proposed user questions when decision making process

- if ranking corresponds to arguments given analytically, and viceversa;
- if similar rankings from different people are comparable analytically;
- what is, in summary, the analytical reason for the ranking;
- how to detect positive or negative spamming.
- what is the lived experiences of users.
- what are the really cool or bad events one might expect for a certain class of rated entities.

[1] www.amazon.com
[2] www.booking.com
[3] www.epinions.com
[4] www.apple.com/itunes
[5] http://www.tripadvisor.com

In our research, we assume that reviews content should have an influence on the rating score. Therefore, we intended to study the correlation between events which could be extracted from user reviews and the rating scores given by users. We aimed to build an approach addressing the first and the last task, i.e. if arguments given in reviews correlate with open rating, and if it is possible to extract relevant event dictionaries from user reviews and use these dictionaries to determine the polarity of reviews.

We formulate these tasks as a binary text classification task. We explored machine learning techniques to build a classifier which allows to classify two types of reviews (positive and negatives) using event features.

To extract events from user reviews, we use FRED to perform a deep semantic parsing of text which, allow us to obtain a RDF Linked-Data-ready graph representation of the text. Those events will be used afterward to construct an event dictionary allowing to discriminate review types.

4.2 Our Approach

Our approach for the polarity classification task can be shown in Figure in 4.2. This approach consists of four main steps:

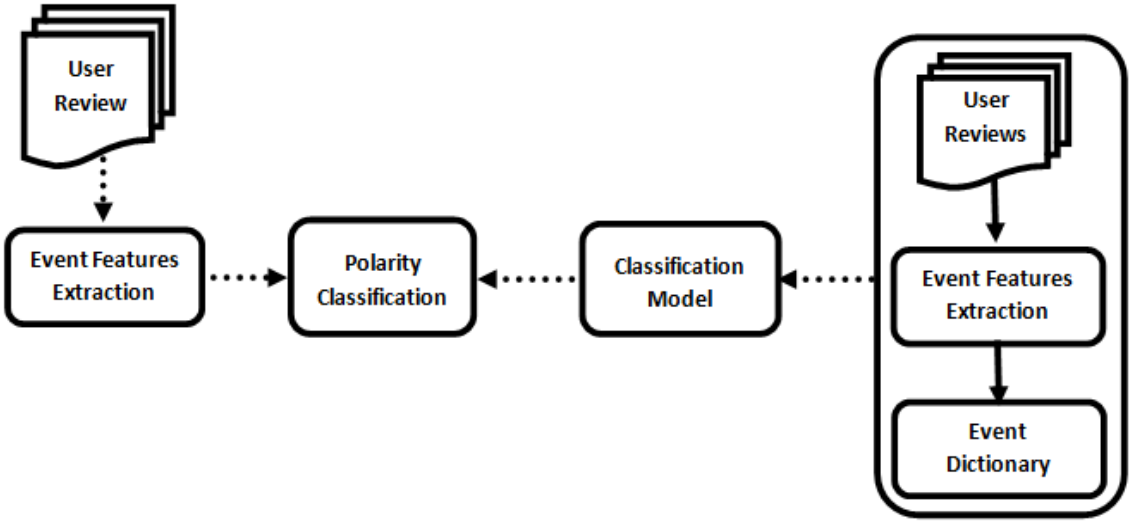


Figure 4.2: Overview of the proposed approach

- Collect a positive and negative set of user reviews. We considered the reviews with rating 5 stars as positive and the reviews with 1 stars as negative. This set will be divided into both training set to build the classifier and test set to evaluate the polarity classification task.
- Extract events with their associated features from both training and test set by performing a deep semantic parsing of text.

- From the training set, build an event dictionary D_E for each class of reviews.
- Use the events in the dictionary as features to build machine learning classifiers.

In the following sections, we show more details about each of these steps.

4.2.1 User Review

A review is an evaluation of a publication, product, service, company, or other object or idea[6]. A user review is a review conducted by a computer user and used on merchant websites to give customers an opportunity to rate and comment on products or services they have purchased. Often, the company will include a URL on printed literature or e-mail marketing to invite customers to review their service after a transaction has been completed. Other consumers can read these reviews when making a purchase decision.

4.2.2 Event Extraction

Extracting, logically representing, and connecting elements from a sentence is crucial to create semantic applications that are event-aware. In addition, it's important to disambiguate as much as possible the entities and concepts expressed, in order to make the extracted model *linked*, and to exploit the full power of the Semantic Web and Linked Data.

Event extraction (EE) can be broadly defined as the creation of specific knowledge concerning facts and situations referred to in some content and/or data: texts, images, video, databases, sensors, etc. One of the interesting task in this proposal is to search and detect events, which appear in user reviews. We focus on events expressed by verbs, propositions, common nouns, and named entities (typically proper nouns).

In order to extract events from user reviews, we performed a deep semantic parsing of reviews and detecting the most appropriate linguistic frames capturing complex relations expressed in the reviews. The deep semantic parsing allows to obtain a RDF/OWL knowledge graph representation of the text. We employed a deep variety of machine reading [Etzioni et al. (2006)], as implemented in the FRED tool[7].

As anticipated in Section 3.2, FRED [Gangemi et al. (2016), Presutti et al. (2012)] is a tool to automatically transform knowledge extracted from text into RDF and OWL, i.e. it is a *machine reader* [Etzioni et al. (2006)] for the Semantic Web. FRED extracts knowledge (named entities, senses, taxonomies, relations, events) from text, resolves it onto the Web of Data, adds data from background knowledge, and represents all that in RDF and OWL. In addition, It is event-centric, therefore it natively supports event extraction (See Chapter 3).

[6] https://en.wikipedia.org/wiki/Review
[7] http://wit.istc.cnr.it/stlab-tools/fred

FRED is available as a RESTful API and as a web application. In its current form, it relies upon several NLP components: Boxer[8] for the extraction of the basic logical form of text and for disambiguation of events to VerbNet, UKB[9] or IMS[10] or BabelNet API[11] for word sense disambiguation, and Apache Stanbol[12] for named entity resolution.

FRED contains several functionalities for event extraction including *Event identity, Event classification* (using VerbNet[13], WordNet[14], DBpedia[15], schema.org, and DOLCE[16] as reference ontologies.), *Event unity* (using VerbNet [Schuler (2005)] and FrameNet [Baker et al. (1998)] to detect event arguments), *Event modifiers* (negation, quality, modality), and *Event relations*. The following segments of a user review is used as a lead example for showing FRED's functionalities:

"*When I asked for refund he was very rude, slamming things down on the counter and swearing. He finally refunded me after 5 minutes of arguing.*"

In the diagram from Figure 4.3, the following events are recognized, extracted, classified, and aligned to WordNet, VerbNet, and/or DOLCE, `ask_1`, `refund_1`, `slam_1`, and `refund_2` (classified as occurrences of the `Ask`, `Slam` and `Refund` frames respectively, and aligned to VerbNet).

The output produced by FRED is a semantic graph which composed of many RDF triples. For example, the triples which concern the event type `Refund` are:

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

fred:refund_1
        rdf:type          fred:Refund .

fred:Refund
        rdfs:subClassOf   DUL:Event .
```

These triples indicate that the event `refund_1` has the type `Refund`, and the last one was classified as an *Event* using the ontology *DOLCE*.

The semantic knowledge graph given as output by FRED could be queried, using a SPARQL query, to extract the required information. In our work, we want to extract mentioned events in the text. Therefore, we applied the following SPARQL query to the semantic graph of FRED using the SPARQL endpoint Corese[17].

[8] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer
[9] http://ixa2.si.ehu.es/ukb/
[10] http://www.comp.nus.edu.sg/~nlp/sw/
[11] http://lcl.uniroma1.it/babelnet/
[12] http://stanbol.apache.org
[13] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html?
[14] http://wordnet.princeton.edu
[15] http://dbpedia.org
[16] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[17] Corese is a Semantic Web Factory (triple store & SPARQL endpoint) implementing RDF, RDFS, SPAQRL 1.1 Query & Update. www.wimmics.inria.fr/corese

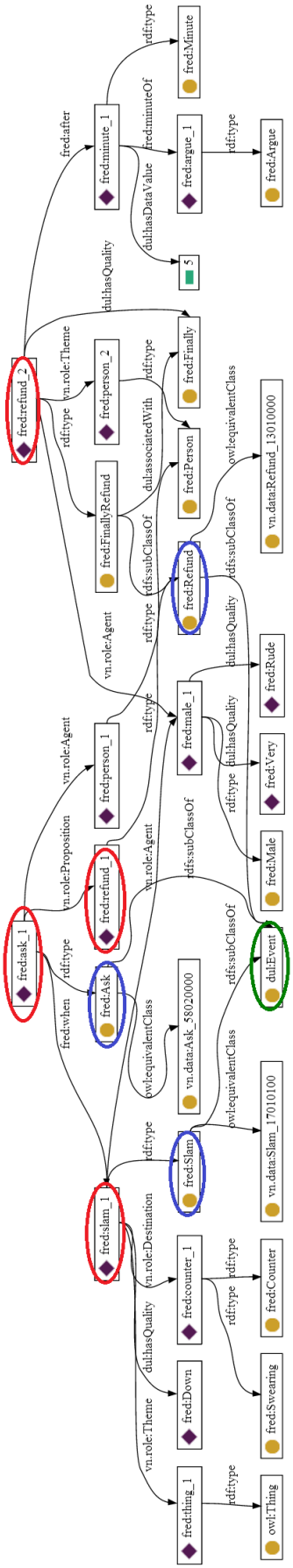


Figure 4.3: The output of FRED for the example text: *"When I asked for refund he was very rude, slamming things down on the counter and swearing. He finally refunded me after 5 minutes of arguing"*

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
PREFIX d0: <http://www.ontologydesignpatterns.org/ont/d0.owl#>
PREFIX schemaorg: <http://schema.org/>
PREFIX fred: <http://www.ontologydesignpatterns.org/ont/fred/domain.owl#>
SELECE distinct ?e ?et
WHERE {
{

{?e rdf:type/rdfs:SubClassOf* dul:Event}
UNION {?e rdf:type/rdfs:SubClassOf* schemaorg:Event}
UNION {?e rdf:type/rdfs:SubClassOf* d0:Event}
}
OPTIONAL {?e rdf:type/rdfs:SubClassOf* ?et Filter (?et != dul:Event)}
OPTIONAL {?e rdf:type ?et}
}
```

This query allows to extract all mentioned events in the text. ?e is the event occurrence in the text and ?et is the type of the occurrence e. The event type et could be modelled as subtype of the class `dul:Event`, `schemaorg:Event`, or `d0:Event`. Therefore, we queried all the event occurrences which their types are sub-classes of the semantic class *Event* in the ontologies `DOLCE`, `schema.org`, and `d0`. From our example, we are able to extract three event types {*Ask, Refund,* and *Slam*}.

4.2.2.1 Event Negation Extraction

Negation is important for user judgment as well as for polarity classification. In our work, we assumed that negation can be useful to classify the two classes of reviews and it can enrich our dictionary with additional events. The FRED tool can capture the logical negation which could be associated with an event in the text using the property `boxing:hasTruthValue` where *boxing* is the prefix which refers to the OWL vocabulary which is created by FRED in order to represent the negation. For example, the sentence *"I did not eat in the hotel restaurant"* expresses an event type *(Eat)* with a negation *(not)*. The RDF/XML triples of FRED for this sentence are:

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

fred:eat_1
        boxing:hasTruthValue     boxing:False ;
        rdf:type                 fred:Eat .

fred:Eat
        rdfs:subClassOf          DUL:Event .
```

These triples indicate that the event `eat_1` have a truth value `(False)` and the event type of this event occurrence is `Eat` which was classified as an *Event* using the ontology `DOLCE`.

To extract events with their negations, if they exist, from user reviews, we modified the previous SPARQL query to become as the following:

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>
PREFIX boxer: <http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl#>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
PREFIX d0: <http://www.ontologydesignpatterns.org/ont/d0.owl#>
PREFIX schemaorg: <http://schema.org/>
PREFIX fred: <http://www.ontologydesignpatterns.org/ont/fred/domain.owl#>
SELECE distinct ?e ?neg ?et
WHERE {
{

{?e rdf:type/rdfs:SubClassOf* dul:Event}
UNION {?e rdf:type/rdfs:SubClassOf* schemaorg:Event}
UNION {?e rdf:type/rdfs:SubClassOf* d0:Event}
}
OPTIONAL {?e boxing:hasTruthValue ?neg}
OPTIONAL {? rdf:type/rdfs:SubClassOf* ?et Filter (?et != dul:Event)}
OPTIONAL {?e rdf:type ?et}
}
```

?*neg* indicates the optional negation which could be associated with an event. Using this query, we extracted the event *Not-Eat*. Notice that we added the prefix *boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>* to be able to extract the negation which should have the predicate `boxing:hasTruthValue`.

4.2.2.2 Event Quality Extraction

Event quality is the adverbial word which could be associated with an event. It is a robust determinant of the sentiment which could be associated with events. In addition, it is very important element for user decision about products or services.

In our work, we assumed that event qualities can upgrade the results of our classification task. Therefore, we extracted these elements and tested their effects on the results of the polarity classification task. Extracting these elements from the semantic graph of FRED is simple, whereas FRED represents such modifiers (i.e. adjectives) as qualities of the modified term by means of the DOLCE property `dul:hasQuality`. Our example in the Figure 4.3, indicates that the event `slam_1` has a quality `Down`, which was indicated by the predicate `dul:hasQuality`.

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

fred:slam_1
        rdf:type         fred:Slam ;
        dul:hasQuality   fred:Down .

fred:Slam
        rdfs:subClassOf  DUL:Event .
```

To extract the qualities from the text, we modified our SPARQL to be as the following:

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>
PREFIX boxer: <http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl#>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
PREFIX d0: <http://www.ontologydesignpatterns.org/ont/d0.owl#>
PREFIX schemaorg: <http://schema.org/>
PREFIX fred: <http://www.ontologydesignpatterns.org/ont/fred/domain.owl#>
SELECE distinct ?e ?neg ?qlt ?et
WHERE {
{

{?e rdf:type/rdfs:SubClassOf* dul:Event}
UNION {?e rdf:type/rdfs:SubClassOf* schemaorg:Event}
UNION {?e rdf:type/rdfs:SubClassOf* d0:Event}
}
OPTIONAL {?e boxing:hasTruthValue ?neg}
OPTIONAL { ?e dul:hasQuality ?qlt}
OPTIONAL {? rdf:type/rdfs:SubClassOf* ?et Filter (?et != dul:Event)}
OPTIONAL {?e rdf:type ?et}
}
```

?*qlt* in this query indicates the quality modifier which could be associated with an event and have the property `dul:hasQuality`.

4.2.2.3 Event Participants Extraction

Event participants are the semantic arguments associated with this event. Each event argument plays a semantic role (e.g. Agent, Patient, Oblique, Theme, etc.). A semantic role is the underlying relationship that a participant has with the main event in a clause. *Agent*[18] is the semantic role of a person or thing who is the doer of an event [Larson (1984), Longacre (2013)], *Patient* is a semantic role of a person or thing that is affected by an event [Larson (1984), Longacre (2013)], *Oblique* is the grammatical relation possessed by all "objects of preposition" in English. *Theme* is a participant which is characterised as changing or moved by an event.

In this work, we assumed that these arguments can improve our classifiers. Therefore, we extracted them from user reviews and then used them to build a dictionary of events and event participants, which will be used to classify the polarity of user reviews.

One of the FRED's functionalities that it applies *semantic role labeling* [Moschitti et al. (2008)] to verbs and propositions in order to detect event participants, and *frame detection* [Coppola et al. (2009)] for resolving roles against a shared event ontology. In other words, FRED uses VerbNet[19] [Schuler (2005)] and FrameNet [Baker et al. (1998)] to represent event types and the relation between events and their participants. In our example in Figure 4.3, the event `refund_2` is detected as a subtype of the class `FinallyRefund`. The last one is detected as a subtype of the event `Refund`. The event type `Refund` is modelled

[18] An agent is usually the grammatical subject of the verb in an active clause.

[19] `http://verbs.colorado.edu/~mpalmer/projects/verbnet.html`

as subtype of the class dul:Event and is aligned to vn.data:Refund_13010000, defined in VerbNet. The relation between fred:FinallyRefund and its arguments are modelled as object properties according to the semantic roles that are recognized, e.g. vn.role:Agent and vn.role:Theme. However, fred:after, fred:minuteOf are additional roles that are detected but not recognized. Therefore, FRED creates and labels them by reusing the appropriate text from the input.

In the following, we show the output of FRED which concerns the event refund_2 and its participants.

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"


fred:refund_2
        fred:after              fred:minute_1 ;
        vn.role:Agent           fred:male_1 ;
        rdf:type                fred:FinallyRefund ;
        vn.role:Theme           fred:person_2 .

fred:minute_1
        :hasQuantifier          quantifiers:multiple ;
        rdf:type                fred:Minute ;
        fred:minuteOf           fred:argue_1 ;
        dul:hasDataValue        5 ;

fred:FinallyRefund
        rdfs:subClassOf         fred:Refund .

fred:Refund
        owl:equivalentClass     vn.data:Refund_13010000 ;
        rdfs:subClassOf         DUL:Event .
```

We can notice from these triples that the participants of the event refund_2 are {minute_1, male_1, person_2}. In addition, we can distinguish between two types of event participants: *(1)* **Direct participants:** which are the arguments which connect to event directly, i.e. the direct objects of the events (e.g. the mentioned participants above). *(2)* **Indirect participants:** which represent the participants in an event participant, i.e. the direct objects of the event participants. In our example, we can notice three indirect participants {multiple, argue_1, 5} which are the participants in the direct event participant minute_1.

In this work, we are interested in event participants for the first, second and third degree. Therefore, we extracted them using the following SPQRQL query:

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>
PREFIX boxer: <http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl#>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
PREFIX d0: <http://www.ontologydesignpatterns.org/ont/d0.owl#>
PREFIX schemaorg: <http://schema.org/>
PREFIX fred: <http://www.ontologydesignpatterns.org/ont/fred/domain.owl#>
PREFIX pos: <http://www.ontologydesignpatterns.org/ont/fred/pos.owl#>
```

```
PREFIX x: <http://www.essepuntato.it/2008/12/earmark#>
PREFIX y: <http://ontologydesignpatterns.org/cp/owl/semiotics.owl#>

Select ?e ?neg ?qlt ?et ?x ?y ?z ?xx ?yy ?zz ?xxx ?yyy ?zzz

WHERE {
 {?e rdf:type/rdfs:subClassOf* dul:Event}
 UNION {?e rdf:type/rdfs:subClassOf* schemaorg:Event}
 UNION {?e rdf:type/rdfs:subClassOf* d0:Event}
 OPTIONAL {?e ?agent ?x
  FILTER (?agent = vnrole:Agent || ?agent = boxer:agent
   || ?agent = vnrole:Experiencer || ?agent = vnrole:Actor
   || ?agent = vnrole:Actor1 || ?agent = vnrole:Actor2
   || ?agent = vnrole:Cause || ?agent = vnrole:Theme
   || ?agent = vnrole:Topic || ?agent = vnrole:Beneficiary
   || ?agent = vnrole:Actor || ?agent = vnrole:Cause)
     OPTIONAL {?x ?aspect1 ?xx
       FILTER (?aspect1 != rdf:type&& ?aspect1 != pos:boxerpos)
     OPTIONAL {?xx ?aspect11 ?xxx  FILTER (?aspect11 != rdf:type
     && ?aspect11 != pos:boxerpos)}
    }}
 OPTIONAL {?e ?atheme ?x
  FILTER (?atheme = vnrole:Theme || ?atheme = vnrole:Theme1
   || ?atheme = vnrole:Theme2 || ?atheme = boxer:theme)
    MINUS {?e ?agent1 ?a
     FILTER (?agent1 = vnrole:Agent || ?agent1 = boxer:agent
      || ?agent1 = vnrole:Experiencer
      || ?agent1 = vnrole:Actor || ?agent1 = vnrole:Actor1
      || ?agent1 = vnrole:Actor2 || ?agent1 = vnrole:Cause) FILTER (?x != ?a)}}
 OPTIONAL {?e ?patient ?y
  FILTER (?patient = vnrole:Patient || ?patient = vnrole:Patient1
     || ?patient = vnrole:Patient2)
   OPTIONAL {?y ?aspect3 ?yy
     FILTER (?aspect3 != rdf:type && ?aspect3 != pos:boxerpos)
      OPTIONAL {?yy ?aspect33 ?yyy
        FILTER (?aspect33 != rdf:type && ?aspect33 != pos:boxerpos)}
    }}
 OPTIONAL {?e ?ptheme ?y
  FILTER (?ptheme != ?atheme && (?ptheme = vnrole:Theme
   || ?ptheme = vnrole:Theme1 || ?ptheme = vnrole:Theme2
   || ?ptheme = boxer:theme))
   {?e ?agent1 ?a
    FILTER (?agent1 = vnrole:Agent || ?agent1 = boxer:agent
     || ?agent1 = vnrole:Experiencer || ?agent1= vnrole:Actor
     || ?agent1= vnrole:Actor1 || ?agent1= vnrole:Actor2
     || ?agent1 = vnrole:Cause) FILTER (?y != ?a)}}
 OPTIONAL {?e ?oblique ?z
  FILTER (?oblique != vnrole:Topic &&  ?oblique != vnrole:Beneficiary
   && ?oblique != vnrole:Patient && ?oblique != vnrole:Patient1
   && ?oblique != vnrole:Patient2 && ?oblique != vnrole:Theme
   && ?oblique != vnrole:Agent && ?oblique != boxer:agent
   && ?oblique != vnrole:Experiencer && ?oblique != vnrole:Actor
   && ?oblique != vnrole:Actor2 && ?oblique != vnrole:Cause
   && ?oblique != boxer:theme && ?oblique != vnrole:Theme1
   && ?oblique != vnrole:Theme2 && ?oblique != rdf:type)
OPTIONAL {?z ?aspect4 ?zz
  FILTER (?aspect4 != rdf:type && ?aspect4 != pos:boxerpos)
   OPTIONAL {?zz ?aspect44 ?zzz
     FILTER (?aspect44 != rdf:type && ?aspect44 != pos:boxerpos)}}}
  OPTIONAL {?e boxing:hasTruthValue ?neg}
  OPTIONAL {?e dul:hasQuality ?qlt}
  OPTIONAL {? rdf:type/rdfs:SubClassOf* ?et Filter (?et != dul:Event)}
  OPTIONAL {?e rdf:type ?et}
  }
```

This query allows us to extract all mentioned events in the text with their negations, qualities, and participants which have a semantic role such as *Agent, Patient, Oblique, Theme*. $?x$, $?y$, and $?z$ indicate the direct participants of the event e (the participants of the first degree in the semantic graph). $?xx$, $?yy$, and $?zz$ represent the indirect participants (the participants of the second degree in the semantic graph). $?xxx$, $?yyy$, and $?zzz$ are the indirect participants (the participants of the third degree in the semantic graph).

4.2.3 Classification Methods

Generally, classification is defined for the situation when there are m objects, each one belonging to one of the n classes, and a classification task would be to assign the belonging class to a new given object. A common approach for solving text classification problem is by training a supervised machine learning classifier such as support vector machines (SVM) or a Naïve Bayes (NB) classifier. The idea is given a set of training examples with their associated labels, the training algorithm constructs a model which is able to predict the category of a new example.

Therefore, to perform supervised classification, we need to separate data into two types of data set:

- Training Set: in this set, we have the input data together with the correct expected class. This data are used to train the classifier.
- Test Set: represents the data that we use to apply our model. This data do not have any expected class, but we use the training set to predict the class of the test set. Generally, this set is used to test the performance of the classifier.

Generally, 70% of the data set is used as a training set to train the classifier, and 30% is used as a test set.

In our work, we choose to build our training model using two classifiers, Naïve Bayes and Support Vector machine, since they can perform well in text classification tasks [Rish (2001), Joachims (1998)].

Naïve Bayes (NB)

Naïve Bayes [Duda et al. (1973)] is a learning algorithm that is frequently employed to tackle text classification problem. It is considered to be one of the simplest among other supervised learners. They are rather easy to implement and computationally inexpensive. Yet, they provide quite good results [Rish (2001)] and often outperform more complex learners. NB are based on the probability model that was formulated by Thomas Bayes, called Bayes' theorem, which can be written in a simple word as follows:

$$\text{posterior probability} = \frac{\text{conditional probability . prior probability}}{\text{evidence}}$$

The general notation of the posterior probability can be written as:

$$P(C_i \mid F_1....F_n) = \frac{P(F_1....F_n \mid C_i)\, P(C_i)}{P(F_1....F_n)}$$

Where:

C_i	class
$F_1....F_n$	set of features

The conditional probabilities $P(F_1....F_n \mid C_i)$ can be calculated as follows:

$$P(F_1....F_n \mid C_i) = P(F_1 \mid C_i).P(F_2 \mid C_i)....P(F_n \mid C_i)$$

$P(F_1 \mid C_i)$ means: How likely is it to observe this particular feature F_1 given that it belongs to class C.

The prior probabilities $P(C_i)$ is the general probability of encountering a particular class. In the case of polarity classification, the priors could be formulated as:

P(Positive) = "the probability that any user review has a positive polarity".

P(Negative) = $1 - P(Positive)$.

The evidence $P(F)$ represents the probability of encountering a particular feature F independent from the class label.

The objective function in the Naïve Bayes probability is to maximize the posterior probability given the training data in order to formulate the decision rule.

Support Vector Machine (SVM)

Support vector machines (SVM) [Cortes and Vapnik (1995)] use an optimization technique called **quadratic programming**[20] to divide samples of two classes as best as possible. SVM classifier represents samples as points in a multidimensional space and then looks for a hyperplane that separates points of one class from the points of another class.

The optimal hyperplane algorithm introduced originally by Vapnik in 1963 was a linear classifier [Vladimir and Vapnik (1995)]. Nevertheless, in 1992, Boser, Guyon and Vapnik [Boser et al. (1992)] have proposed a way to build non-linear classifiers by using the *kernel trick*. The idea behind kernel methods is to map the input data points into a higher dimensional space, where the separating hyperplane must be found. In the literature, there is a large variety of kernels that can be used. Examples of such kernels are: Linear, Polynomial, Normalised Polynomial, Laplacian, Gaussian or Euclidean. The choice of a suitable kernel function K is crucial for the learning process and must be performed depending on the particular problem that has to be solved. While there exists an infinite number of such hyperplanes, an SVM chooses the one with the largest

[20] is the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables.

margins between the hyperplane and the points representing a class. These points are called **support vectors** (See Figure 4.4).

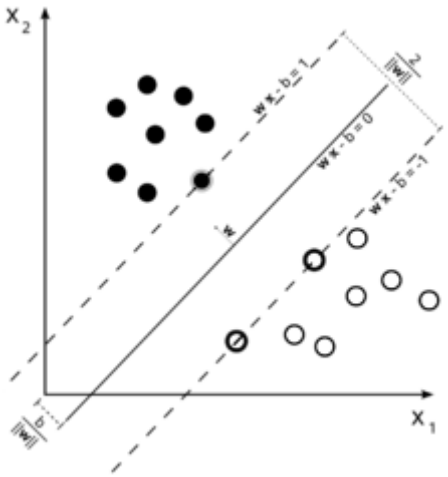


Figure 4.4

Cross Validation Technique:

The most popular manner to determine the optimal value of class C is through cross-validation. The training data is randomly partitioned into k folds having approximately equal size. Then, $k-1$ folds are used for learning and the remaining one for testing. The process is repeated for all the folds (k times) and an average of the performances is computed [Olson and Delen (2008)].

To produce a classifier's input, texts from the dataset are being represented as a features vector, where feature values encodes the presences of these features in the text. In our work, we use several semantic features, such as events, event qualities, and event participants to train two classifiers: Naïve Bayes and Support Vector Machine. The following section describes the dataset that we are used (Section 4.3.1) and reports the experimental setting (Section 4.3.2), and then turns to the results of our experiments (Section 4.3.3).

4.3 Experiments

Our experiment deals with TripAdvisor[21] hotel reviews, which contain both open rating and analytic text. The experiment tests if relevant event dictionaries can be extracted from these reviews, and then used as features to predict whether a review is positive or negative.

[21] http://www.tripadvisor.com/

4.3.1 DataSet

[Ott et al. (2011)] have recently created the first publicly available[22] dataset for deceptive opinion spam research containing 800 positive reviews (400 truthful reviews and 400 fake reviews) which have been assigned with 5-stars in the open system ranking and 800 reviews for the negative polarity (400 truthful reviews and 400 fake reviews) which have 1-star in the open system ranking. In this work, we were only interested in the truthful reviews which are collected from the 20 most popular Chicago hotels on TripAdvisor. We selected 600 reviews, 300 user reviews for the positive reviews (15 reviews for each hotel) and 300 user reviews for the negative ones. In our experimentation, 420 user reviews were used as a training set, 210 reviews for the positive class and 210 reviews for the negative one, and 180 user reviews were used to evaluate our classifier (90 for each class).

We include 50% positive reviews and 50% negative reviews in our training set in order to create a balanced dataset which allow to create a good event dictionary containing extracted events from the same number of reviews for the two classes.

We chose to not take into account any mixed review (that is, reviews having 3 stars in the system ranking) because they could contain mixed types of events (both positive and negative), while we had the objective to find discriminant events, or at least to have reviews in which most events show a correlation to the evaluation assigned by users.

rev.class	Training Set	Test Set	Total
Positive	210	90	300
Negative	210	90	300
Total	420	180	600

Table 4.1: Characteristics of datasets used in our experiment

4.3.2 Development of the Review Classifier

Using Only Events

Using our training set, we were able to recognize and extract 8388 events: 3106 for the positive reviews and 5282 for the negative ones. The number of aggregated events is 1867: 716 for the positive reviews and 1151 for the negative ones (Table 4.2).

rev.class	events	aggregated events
Positive	3106	716
Negative	5282	1151
Overall	8388	1867

Table 4.2: The number of extracted events from our training set for the two classes of review.

[22] Available by request at: `http://www.cs.cornell.edu/~myleott/op_spam`

The number of common event types which appear in the two classes, positive and negative, are 320. Therefore, our dictionary contains 1547 event types ($716 + 1151 - 320 = 1547$).

The resulting dictionary could not be used to discriminate between the two types of events since it contains common events which appear in the two classes of review. So, we decided to look at the frequencies of events and remove the events which are distributed in the two classes in a manner homogeneous. Therefore, we decided to look at event frequencies to indicate these events and remove them from the dictionary. We considered that an event is representative to a class if the probability $P(c|e) \geq \sigma$ where:
$c \in C = \{+,-\}$;
e: A generic event;
σ: A threshold that we determined empirically between 4 possibilities: {0.6, 0.7, 0.8, 0.9}. The best value of σ is 0.7 (See section 4.3.5).
For example, the event *Love* appeared 41 times in the positive reviews and 8 times in the negative ones. It can be very useful to discriminate the positive reviews.

$$P(+|\text{“}Love\text{”}) = \frac{41}{41+8} \simeq 0.84 > 0.7.$$
$$P(-|\text{“}Love\text{”}) = \frac{4}{4+41} \simeq 0.16.$$

However, the event *Cancel* appeared 11 times in the negative reviews and 3 times in the positive ones. This event type is helpful to discriminate the negative review because:

$$P(+|\text{“}Cancel\text{”}) = \frac{3}{3+11} \simeq 0.21.$$
$$P(-|\text{“}Cancel\text{”}) = \frac{11}{3+11} \simeq 0.79 > 0.7.$$

While, the event *Stay* frequented 321 times in the positive reviews and 262 times in the negative ones. This event appears almost identically in the two classes and should be deleted since it cannot be useful for our classification.

$$P(+|\text{“}Stay\text{”}) = \frac{321}{321+262} = 0.55 < 0.7.$$
$$P(-|\text{“}Stay\text{”}) = \frac{262}{321+262} = 0.45 < 0.7.$$

In the end, we removed 205 common events which do not help to discriminate the reviews types, and left 115 events among the events that exist in the two classes (23 events for the positive reviews and 92 for the negative ones): the result was a dictionary containing 1342 characterizing events: 396 uniques for the positive reviews, 23 common events discriminating the positive reviews, 831 uniques for the negative reviews, and 92 common events but important to discriminate the negative reviews.

In table 4.3 and table 4.4, we show the most frequent event types in the positive and negative dictionaries, respectively used to classify the polarity of reviews. Table 4.5 shows some of common event types that we are deleted from our dictionary.

Event Type	$C_+(e)$	$C_-(e)$	$P(+\|e)$
Walk	80	19	0.76
Love	41	8	0.84
Recommend	39	10	0.80
Enjoy	29	5	0.85
DefinitelyStay	20	3	0.87
HighlyRecommend	15	1	0.94
DefinitelyRecommend	10	0	1
ReallyEnjoy	8	0	1
Decorate	6	1	0.86
Welcome	5	1	0.83
Wish	5	1	0.83
Please	5	2	0.71
Not-Beat	4	0	1
Appreciate	4	0	1
Not-Believe	3	0	1
Not-Hear	3	1	0.75
RecentlyRenovate	3	1	0.75

Table 4.3: Most frequent event types used for classifying positive reviews

Add event qualities to the dictionary

As we mentioned above, we aimed to show the effects of event qualities on the results. Therefore, we extracted them from our training set and added them to the event dictionary. Using our training set, we were able to recognize and extract 9169 events and qualities: 3645 for the positive reviews and 5524 for the negative ones. The number of aggregated events and qualities is 2267: 915 for the positive reviews and 1352 for the negative ones (Table 4.6).

The number of common events and qualities are 451. Therefore, the new dictionary contains $915 + 1352 - 451 = 1816$ events and qualities. As we found when we built the event dictionary, some common events and qualities could not be used to discriminate between the two types of reviews. Therefore, we looked at their frequencies to indicate whether they can be representative to a class or not. As like event dictionary, we used the probability $P(c|e) \geq \sigma$ for this purpose.

According to the frequencies of these events and qualities, we removed 269 common events and qualities which do not help to discriminate the reviews types, and left 182 events and qualities among the events and qualities that exist in the two classes (43 events and qualities for the positive reviews and 139 for the negative ones): the result was a dictionary

Event Type	$C_+(e)$	$C_-(e)$	$P(-\|e)$
Call	16	93	0.85
Check	17	50	0.75
Find	18	48	0.73
Arrive	15	42	0.74
Give	14	41	0.75
Look	16	41	0.72
Ask	14	36	0.72
Request	4	34	0.89
Hear	7	28	0.80
Charge	8	27	0.77
Not-Work	2	25	0.93
Wait	6	25	0.81
Complain	4	14	0.78
Not-Want	0	13	1
NeverStay	0	11	1
Cancel	3	11	0.79
Not-Expect	2	7	0.78
Not-Care	1	5	0.83
Not-Feel	1	5	0.83
Not-Apologize	0	3	1

Table 4.4: Most frequent event types used for classifying negative reviews

Event Type	$C_+(e)$	$C_-(e)$
Stay	321	262
Have	175	252
Get	75	100
Go	42	83
Make	34	53
Take	25	56
Feel	23	20
Use	17	36
Not-Have	17	35
Come	16	37
Need	16	21
Think	15	25
Expect	13	21
Choose	10	13
Locate	13	7

Table 4.5: Some event types removed from the dictionary

containing 1547 characterizing events and qualities: 464 uniques for the positive reviews, 43 common events and qualities discriminating the positive reviews, 901 uniques for the

rev.class	events & qualities	aggregated events & qualities
Positive	3645	915
Negative	5524	1352
Overall	9169	2267

Table 4.6: The number of extracted events and qualities from our training set for the two classes of review.

negative reviews, and 139 common events and qualities, but important to discriminate the negative reviews.

Add event participants to the dictionary

We also built other dictionary involving events and their participants, which can be found in the training set. We extracted event participants for the first, second, and third degree. Using the training set, we extracted 37941 events and event participants: 14222 for the positive reviews and 23719 for the negative ones. The number of aggregated events and participants is 4835: 2019 for the positive reviews and 2816 for the negative ones (Table 4.7).

rev.class	events & participants	aggregated events & participants
Positive	14222	2019
Negative	23719	2816
Overall	37941	4835

Table 4.7: The number of extracted events and event participants from our training set for the two classes of review.

The number of events and event participants, which are found in the two classes of reviews 1014. Therefore, our new dictionary contains $4835 - 1014 = 3821$ features. As we mentioned before, some common features could not improve the results of the polarity classification task. So, we should delete them from the dictionary. Using the probability $P(c|e) \geq \sigma$, we removed 626 common events and event participants and kept 388 ones to obtain at the end a new dictionary of event and event participants containing 3197 features: 1005 uniques for the positive reviews, 121 common events and event participants discriminating the positive reviews, 1802 uniques for the negative reviews, and 267 common events and event participants which are important to discriminate the negative reviews.

4.3.3 Evaluation

Application to the training set:

The events, event qualities, and event participants were used as features for a multinomial Naïve Bayes and Support Victor Machine classifiers. Each review is represented by a feature vector where the $i-$th component value is the frequency of the $i-$th feature in the review, and the last component is the type of the review (positive or negative). We chose

the Weka[23] implementation for the classifier using cross-validation techniques to validate the model (in particular, we used 10-fold cross-validation). We use the 420 reviews of the 600 truthful reviews which are created by [Ott et al. (2011)] as our training set. Our approach is evaluated in terms of precision *(P)*, recall *(R)*, and F-measure *(F)*. The obtained results can be observed in Table 4.8 and Table 4.9, respectively. *(P)* measures the ratio of correctly classified reviews among the retrieved user reviews, while R measures the ratio of correctly classified reviews to all reviews for this class, and *(F)* is the weighted average of the precision and Recall. *(P)*, R, and *(F)* are computed as follows:

$$P = \frac{t_p}{t_p + f_p}$$

$$R = \frac{t_p}{t_p + f_n}$$

$$F = \frac{2 * P * R}{P + R}$$

Where t_p is true positive (i.e. the number of user reviews predicted positive that are actually positive), f_p is false positive (i.e. the number of user reviews predicted positive that are actually negative), and f_n is false negative (i.e. the number of user reviews predicted negative that are actually positive).

Application to the test set:

To validate the obtained results, we applied our classifier to the test set, which contain 180 user reviews were (90 positive reviews and 90 negative ones). Table 4.8 and Table 4.9 show the achieved results using SVM and NB classifiers.

		Training Set			Test Set		
Features	*Nb_Features*	P	R	F	P	R	F
All_Eve	1547	79.8%	79.8%	79.8%	78.3%	77.8%	78%
$Eve_\sigma = 0.7$	1342	82.6%	82.6%	82.6%	75.5%	73.3%	74.4%
All_Eve_Qlt	1816	79.5%	78.8%	79.1%	80.6%	80.6%	80.6%
(*Eve-Qlt*)$\sigma = 0.7$	1547	84%	84%	84%	80.6%	80.6%	80.6%
All_Eve_Part	3821	83.4%	83.1%	83.3%	**82.8**%	**82.8**%	**82.8**%
(*Eve-Part*)$\sigma = 0.7$	3197	**88.4**%	**88.1**%	**88.3**%	80.3%	79.4%	79.7%

Table 4.8: Overall results for review classification using NB method

These tables show the results of our experiments using six configurations: *All_Eve* configuration presents the results of using all extracted events, without deleting the common events, as features, $Eve_\sigma = 0.7$ presents the obtained results by deleting the events which cannot discriminate the user reviews (common events) and used the rest as features,

[23] http://www.cs.waikato.ac.nz/ml/weka/

		Training Set			Test Set		
Features	*Nb_Features*	P	R	F	P	R	F
All_Eve	1547	79.6%	78.3%	78.9%	76.2%	75%	75.6%
$Eve_{\sigma} = 0.7$	1342	81.5%	79.5%	80.5%	73.8%	71.7%	72.7%
All_Eve_Qlt	1816	76%	75%	75.5%	73.9%	72.2%	73%
(*Eve-Qlt*)$\sigma = 0.7$	1547	**82.6**%	**80.7**%	**81.6**%	**80.3**%	**77.8**%	**79**%
All_Eve_Part	3821	81.3%	80.2%	80.7%	77.5%	75%	76.2%
(*Eve-Part*)$\sigma = 0.7$	3197	81.4%	80.2%	80.8%	78.7%	76.1%	77.4%

Table 4.9: Overall results for review classification using SVM method

All_Eve_Qlt presents the achieved results by using all events with their qualities as features, and (*Eve-Qlt*)$\sigma = 0.7$ presents the achieved results by deleting the common events and qualities, and used the rest as features, *All_Eve_Part* indicates the results using all extracted events and their participants as features, (*Eve-Part*)$\sigma = 0.7$ presents the obtained results when we delete some common events and event participants.

As shown in these tables, NB outperforms SVM for all the used features. The best results on the training set for NB classifier were obtained using the configuration (*Eve-Part*)$\sigma = 0.7$. However, for the test set, *All_Eve_Part* achieves the best results using NB method. For SVM classifier, (*Eve-Qlt*)$\sigma = 0.7$ gives the best results for both training and test set. In addition, using all events as features have been particularly good for the two classifiers. Further deleting some common events, we improve the results by about 3% for NB classifier and 2% for SVM on the training set. Furthermore, adding event qualities to the event dictionary upgrade the results by about 2% for both NB and SVM classifiers on the training set and about 6% on the test set. Besides, adding participant features to the dictionary improve the results by about 4% on the training set using NB method. However, these features decrease the performance by about 1% on the training set using SVM method.

4.3.4 Error Analysis

The classification errors are due to discriminant events for a type reviews, which exist in reviews of the other type. For example, from a positive review, according to a user, we extracted the following events: {*Get*, *Help*, *Put*, *Take*, *Experience*}. The classification results indicate that this review is negative. By looking in our dictionary, we only find the event {*Put*} among the other specified events. This event discriminates the negative reviews in our dictionary, but it exists in a positive review. For this reason, we are motivated to study and understand the impact of events. The other events of this review have been removed from the dictionary because they are common events and do not help to discriminate the two types of reviews.

4.3.5 σ Estimation

To properly build and have a dictionary allowing to discriminate the two types of reviews, we considered that an event is representative with respect to a class if the probability $P(c|e) \geq \sigma$. To determine the best value of σ, we tested 4 possibilities {0.6, 0.7, 0.8, 0.9}. Table 4.10 and Table 4.11, respectively, show the results of our experiment using Naïve Bayes and Support vector machine methods with *10 fold cross-validation* technique in the training with the 4 values of σ.

σ	**Precision**	**Recall**	**F-Measure**
0.6	80.5%	80.4%	80.5%
0.7	**82.6%**	**82.6%**	**82.6%**
0.8	81.3%	81.4%	81.4%
0.9	80%	80%	80%

Table 4.10: Results for the *10-fold cross* experiments for several values of σ for our training set using NB method

σ	**Precision**	**Recall**	**F-Measure**
0.6	80%	79.9%	80%
0.7	**81.5%**	**79.5%**	**80.5%**
0.8	80.4%	80.4%	80.4%
0.9	80%	79.9%	80%

Table 4.11: Results for the *10-fold cross* experiments for several values of σ for our training set using SVM method

These tables indicate that when $\sigma = 0.7$, we obtain the best performance for both NB and SVM method. These results justify our choice of the value of σ as 0.7 when we built our event dictionary in Section 4.3.2.

4.3.6 BaseLine

To be able to meaningfully evaluate our model, we needed to establish a reasonable baseline. We compared our method against a simple baseline, which involves extracting just the verbs which exist in our training set (240 user reviews: 120 for the positive class and 120 for the negative one) and used them as features for a multinomial Naïve Bayes and Support Vector Machine classifiers to discriminate between the two types of reviews. We chose to compare our method to this model, since most of our event features are represented by verbs.

To extract those verbs, we used the semantic graph that is generated by FRED. FRED graphs contain some annotations such as the Part-Of-Speech (POS) for each term in the

text. The POS annotations have been represented using the ontology *pos.owl*[24], and they can be captured using the property *boxerpos*. For example, the output of FRED for the sentence *I did not eat in the hotel restaurant* contains a triple that indicates that the POS of the word *Eat* is a *v*:

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:j.6="http://www.ontologydesignpatterns.org/ont/fred/pos.owl#"

fred:eat_1
        rdf:type        fred:Eat .

fred:Eat
    j.6:boxerpos        pos.owl:v ;
    rdfs:subClassOf     DUL:Event .
```

In order to retrieve verbs from the FRED's semantic graph, we used the following query:

```
PREFIX  pos:<http://www.ontologydesignpatterns.org/ont/fred/pos.owl#>
SELECT distinct ?p
WHERE {

        {?p  pos:boxerpos pos:v}
      }
```

By following the same process that we are performed when we built the event dictionary, we are able to extract from our training set 6883 verbs: 2539 for the positive reviews and 4344 for the negative ones. The number of aggregated verbs is 1310: 504 for the positive reviews and 806 for the negative ones (Table 4.12).

rev.class	**verbs**	**aggregated verbs**
Positive	2539	504
Negative	4344	806
Overall	6883	1310

Table 4.12: The number of extracted verbs from our training set for the two classes of review.

The number of common verbs is 317. Therefore, the number of the dictionary which contain all the mentioned verbs in the training set is $1310 - 317 = 993$. However, when we use the probability condition $P(c|e) \geq \sigma$ to remove the verbs which are distributed in the two classes in a manner homogeneous, we obtain a verb dictionary containing 768 verbs: 187 uniques for the positive reviews, 29 common verbs discriminating the positive reviews, 489 uniques for the negative reviews, and 93 common verbs but important to discriminate

[24]This ontology contains the part-of-speech tags used in the Penn Treebank Project. It also contains three POS (v, a, n) used in Boxer as a simplification of the Penn ones

the negative reviews. We removed 195 common verbs from the dictionary and kept 122 common ones.

The two verb dictionaries are then used to build our classifiers. Table 4.13 and Table 4.14, respectively, show the results in both training and test set using NB and SVM methods for each verb dictionary.

		Training Set			Test Set		
Features	$Nb\_Features$	P	R	F	P	R	F
All_Verbs	993	74.2%	74%	74.1%	72.8%	73.1%	72.9%
$Verbs_{\sigma} = 0.7$	796	76%	76%	76%	70.3%	70.9%	70.6%

Table 4.13: Overall results for review classification using NB method and verb dictionary

		Training Set			Test Set		
Features	$Nb\_Features$	P	R	F	P	R	F
All_Verbs	993	73.8%	73.5%	73.6%	70.5%	69.7%	70.1%
$Verbs_{\sigma} = 0.7$	796	75.2%	74.7%	74.9%	68.8%	67.4%	68.1%

Table 4.14: Overall results for review classification using SVM method and verb dictionary

We can notice from these tables and the obtained results for event dictionaries in Table 4.8 and Table 4.8, that events perform better than verbs for the polarity classification. In other word, our results in this task of correlating the reviews to their rating are better than the baseline, we feel entitled to apply our method to the investigation about rating motivation. Notice that verbs alone can be hardly used for summarization, because verb vectors have no structure to be used for a semantic summarization task. Event graphs provided by FRED have already shown intuitively.

4.3.7 Application to ESWC2014 challenge

To be able to meaningfully evaluate our model, we compared our approach with the systems which participated in the Polarity Detection task, the elementary task in the ESWC-14 challenge on Concept Level Sentiment Analysis [Recupero and Cambria (2014)]. The reviews which are used in this task were extracted from the Blitzer dataset[25]. To build our classifier for this task, we extract events with their qualities from the training set which contain 8000 reviews (4000 positives and 4000 negatives). Then, we used them as features for a multinomial Naïve Bayes classifier.

Table 4.15 shows the results of our approach and the results of the top three participants in this challenge. The evaluation is carried out on the test, which is composed of 2429 sentences constructed in the same way and from the same sources as the Blitzer dataset. Our system achieved the best performance on Recall and the second best system in Precision and F-measure.

[25] http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

Participant	Precision	Recall	F-Measure	Final position
NCU	0.78	0.57	0.66	1
IBM	0.66	0.59	0.62	2
FBK	0.42	0.47	0.44	3
Our system	0.68	0.60	0.63	

Table 4.15: Results of Polarity Detection Task at ESWC2014

4.4 Conclusion

In this chapter, we presented an approach to detect and classify the polarity for customer reviews based on events. We performed a deep semantic parsing with the help of FRED to obtain a RDF/OWL knowledge graph representation of the text. The semantic graph was queried in order to extract event-based features and we used them in Naïve Bayes and Support Vector Machine classifiers to predict the classes of the reviews and study the correlation between the reviews and theirs rating. We were capable to build event-based dictionaries, which can help discriminate the two types of reviews. The results in the classification experiment show a certain dependence from the chosen training set. However, we were able to use events and their participants and qualities to correlate the reviews to their rating, confirming our initial hypothesis that events may have an influence on the rating scores given by users. We can conclude positively about our testing that if machine reading is accurate enough, it can be employed to investigate the correlation between synthetic and analytic judgments.

The work presented in this chapter has resulted in the following publication:

- Ehab Hassan, Davide Buscaldi, and Aldo Gangemi. Correlating open rating systems and event extraction from text. In the 22nd International Conference on Neural Information Processing, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, pages 367-375. Springer, 2015.

Chapter 5

Lived Experience Recognition

Contents

5.1 Introduction 82
5.2 Defining a Lived Experience 83
5.3 Extracting Lived Experiences 84
5.3.1 Event Extraction 84
5.3.2 Personal Events Identification 87
5.3.3 Identification of Reviews containing LEs 88
5.3.4 Event Filtering 92
5.3.5 Event participant extraction 95
5.3.6 Extraction of sentences containing lived experiences 98
5.3.7 Lived Experience Graph Representation 99
5.4 Experimental Evaluation 99
5.4.1 Dataset 99
5.4.2 Review identification evaluation 100
5.4.3 Lived experience extraction evaluation 104
5.5 LEE as a RESTful API 106
5.6 Conclusion 108

In this chapter, we propose an operational definition of lived experience and present our approach based on machine learning and machine reading to identify and extract lived experiences from user reviews. Given a set of customer reviews of a product or a service, the task involves three subtasks: (1) identifying reviews which contain lived experiences; (2) for each pertinent review, identifying and extracting the relevant events with their participants, which indicate lived experience contents; (3) representing lived experiences in each review as an event sub-graph containing relevant events and their participants. Finally, we also provide an API to get the extracted subgraphs as linked data.

This chapter is structured as follows: in Section 5.1 we recall the motivations behind our work of extracting lived experiences from user reviews. In Section 5.2, we discuss our operational notion of *lived experience*, and we show some explanatory examples. Section 5.3 presents the identification of reviews, which contain lived experiences, the extraction of lived experience contents, and the selection techniques. Section 5.4 describes the evaluation performed for review identification and lived experiences extraction tasks. Section 5.5

introduces a web application, called LEE (*Lived Experience Extraction*)[1] for the extraction of Lived Experience in the form of RDF graphs.

5.1 Introduction

The web has significantly changed how people express themselves and interact with others. Now they can post reviews of products and services in merchant websites, as well as they express their viewpoints and interact with others through blogs and forums. It is now well agreed that user generated content contains valuable information that can be used for real word applications (e-commerce, politics, finance, etc.). As e-commerce is becoming more and more popular, the number of user reviews for a specific product or service may be in hundreds or even thousands. Because of this, taking a decision about commercial offers from a large amount of data on the Web becomes difficult, and takes a lot of time.

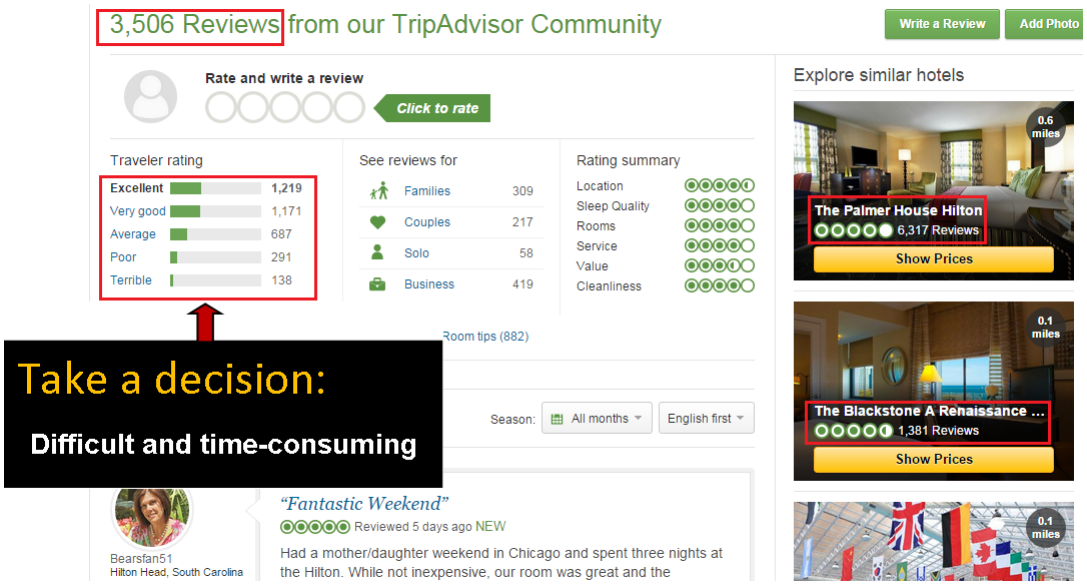


Figure 5.1: The number of user review on TripAdvisor for the Talbott Hotel

Automated solutions for this problem, either from recommender systems [Pazzani and Billsus (2007)], commonly exploiting collaborative or content-based filtering, or from user-based ranking systems, have known limitation including provenance assessment, spam detection, and genericity.

Reviews offer (often implicitly) suggestions or opinions on the basis of lived experiences. These reviews are very important in user decisions since they contain non-fictional narrative or stories that people tell about their experience with a product or service. Users can rely on these reviews to project themselves as a potential future consumer, compare their desires and requirements to those of other customers, and make a decision quickly. Lived experiences can give them specific and more interesting information than general opinions,

[1] https://lipn.univ-paris13.fr/ClientProj/client.jsp

and provide a larger palette of perspectives than traditional sentiment analysis, since experiences differ among users and hint at their own preferences and reasons for judgment. This would allow them also to avoid the loss of time during the making-decision process, and take a good decision since lived experiences are often decisive in customer selection.

5.2 Defining a Lived Experience

We operationally define a *lived user experience* (for the sake of our experiment) as an event mentioned in a review, where the author is among the participants. In addition, we separate lived experiences from generic user opinions, e.g. a situation involving the author giving an opinion about a service or facility is not considered as a lived experience. In other words, we want to detect events in which the author of a review is doing something together with anything associated with a product or service, separating this "quasi-objective" reporting of factuality from any judgment of it. This choice stands on the hypothesis that fake reviews tend to contain opinions that are not associated with actual events. Our hypothesis is supported by spam detection results on so-called *defaming spam* [Jindal and Liu (2008)]. As an example, let us consider the two following hotel reviews:

1. *The view from this hotel's rooms is quite stunning. And that's what make it very special, possibly better than the next door 4 star hotel and than many other hotels in Paris. The bedrooms interior decor is extremely nice. I asked for a room overlooking the pantheon and I got it. My deluxe room was number 32, and was tastefully decorated with a classic and beautiful Pierre Frey wallpaper, and an extra day bed. The bath had bathtub-shower combination and was separated from the toilet. If you book directly through the hotel, you'll get a voucher for a free-drink upon arrival. It was a bit cold at night at some point, maybe because it's March and the heating is not constantly on anymore. Each room has its own heating control, though. Strongly recommended.*

2. *our stay was absolutely perfect. its a cool hotel to look at, the design and feel is very trendy and hip. all the staff are terrific, especially the concierge staff-great info and attitude. our room was on the top floor, with great views. super comfy bed, and neat bathroom. fantastic, choose this with no hesitation!*

According to our definition, the first review contains three lived experiences represented by events where the user is among their participants (1) *I asked for a room overlooking the pantheon and I got it.* (2) *My deluxe room was number 32, and was tastefully decorated with a classic and beautiful Pierre Frey wallpaper, and an extra day bed.* (3) *If you book directly through the hotel, you'll get a voucher for a free-drink upon arrival.* The first lived experience has two event types *{Ask, Get}*, the second and the third have one event type *{Decorate}*, *{Get}*, respectively. All these events have the author as a participant *{I, My, You}*. On the contrary, the other sentences in this review do not represent lived experiences, as they do not contain events *(The bedrooms interior decor is extremely nice)*

or they include events, but those events do not have the author as a participant *(The bath had bathtub-shower combination and was separated from the toilet)*.

In the second review, the user writes his opinion in general without telling anything about his lived experiences *(all the staff are terrific, super comfy bed, ...)*. We do not notice any event or action (e.g. what, when, where, how), in which the user was participating. The sentence *"our room was on the top floor, with great views"* is not a lived experience sentence. It does not contain events. The verb "be" is a state verb which does not indicate an event. Following our operational definition, we discard this review from the set including lived experiences.

We expect that when choosing a service or product, the decision process assisted by lived experience extraction is quicker and more efficient, since it will be made based on the segments of relevant reviews that mostly stimulate *identification* in the reader. This is supported by results in [Duffy (2012)], which show that "*[sites such as] TripAdvisor [...] can involve an "apomediary effect" in which technological features and social identification combine in some circumstances to reduce information to a manageable level*".

Concerning representation, we adopt a neo-Davidsonian modeling of events [Kamp (1981)] as modeled in the OWL knowledge graphs extracted by the FRED tool (cf. Section 5.3).

5.3 Extracting Lived Experiences

Figure (5.2) is an architectural overview of our lived experiences extraction system. The input to the system is a user review. The output is the event-based lived experiences mentioned in the review.

The LEE system performs lived experiences extraction in three main steps: (1) indicating whether the review contains lived experiences contents or not; (2) if yes, identifying lived experience sentences; (3) representing the results as event sub-graphs. These steps are performed in multiple sub-steps.

Given the input, the system firstly finds event type features which are mentioned in a review. After that, the review is classified as a lived experience review if it contains one or more lived experience content, or not otherwise. For lived experience reviews, the system finds the pertinent events that indicate the main parts of lived experiences. In the last two steps, event participants are identified, and lived experience contents are extracted. Below, we discuss each of the sub-steps.

5.3.1 Event Extraction

As explained in Section 5.2, we consider events mentioned in user reviews, and have the review author among the participants, as a lived experience. Therefore, a lived experience should involve events and their participants, and the narrator should be among those

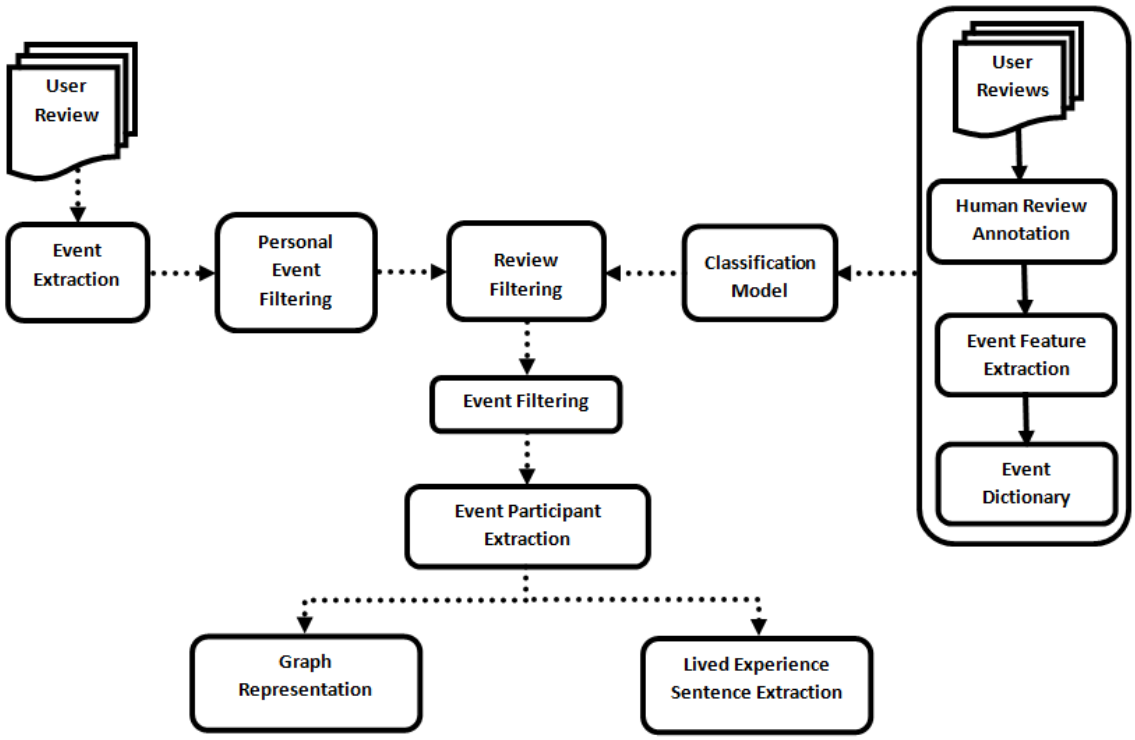


Figure 5.2: Event-based Lived Experience Extraction

participants. In this section, we describe our approach to extract events from customer reviews.

Event extraction (EE) can be broadly defined as the creation of specific knowledge concerning facts and situations referred to in some content and/or data: texts, images, video, databases, sensors, etc. In this research, we focus on events expressed by verbs, propositions, common nouns, and named entities (typically proper nouns).

In order to extract events from user reviews, we followed our approach in chapter 4, Section 4.2.2. We performed a deep semantic parsing of text which allow to obtain a RDF/OWL knowledge graph representation of the text. We employed a deep variety of machine reading [Etzioni et al. (2006)], as implemented in the FRED tool[2] [Gangemi et al. (2016), Presutti et al. (2012)]. FRED extracts knowledge graphs (formal representation of named entities, senses, taxonomies, relations, events, etc.) from text, resolves it onto Linked Open Data (DBpedia[3], schema.org, RDF versions of WordNet[4] [Miller and Fellbaum (1998)], FrameNet [Baker et al. (1998)], VerbNet[5] [Schuler (2005)], and adds data from background knowledge.

By applying the following SPARQL query to the semantic graph produced by FRED, we can extract event types with their main participants.

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
```

[2] http://wit.istc.cnr.it/stlab-tools/fred
[3] http://dbpedia.org
[4] http://wordnet.princeton.edu
[5] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html?

```
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>
PREFIX boxer: <http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl#>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
PREFIX d0: <http://www.ontologydesignpatterns.org/ont/d0.owl#>
PREFIX schemaorg: <http://schema.org/>
PREFIX fred: <http://www.ontologydesignpatterns.org/ont/fred/domain.owl#>
PREFIX pos: <http://www.ontologydesignpatterns.org/ont/fred/pos.owl#>
PREFIX x: <http://www.essepuntato.it/2008/12/earmark#>
PREFIX y: <http://ontologydesignpatterns.org/cp/owl/semiotics.owl#>

CONSTRUCT {
     ?e :agent ?x .  ?x ?aspect1 ?xx . ?xx ?aspect11 ?xxx .
     ?e :patient ?y . ?y ?aspect3 ?yy . ?yy ?aspect33 ?yyy .
     ?e ?oblique ?z .  ?z ?aspect4 ?zz.  ?zz ?aspect44 ?zzz .
     ?e ?r ?e1  .  ?e1 ?r1 ?e2 . ?e2 ?r2 ?e3 .
     ?e ?aspect ?mod . ?mod ?aspmod ?mod1 . ?mod1 ?aspmod1 ?mod2
         }
   WHERE{
       {?e rdf:type/rdfs:subClassOf* dul:Event}
       UNION {?e rdf:type/rdfs:subClassOf* schemaorg:Event}
       UNION {?e rdf:type/rdfs:subClassOf* d0:Event}
       OPTIONAL {?e ?agent ?x
          FILTER (?agent = vnrole:Agent || ?agent = boxer:agent
                 || ?agent = vnrole:Experiencer || ?agent =   vnrole:Actor
                 || ?agent = vnrole:Actor1 || ?agent = vnrole:Actor2
                 || ?agent = vnrole:Cause || ?agent = vnrole:Theme
                 || ?agent = vnrole:Topic || ?agent = vnrole:Beneficiary
                 || ?agent = vnrole:Actor || ?agent = vnrole:Cause)
          OPTIONAL {?x ?aspect1 ?xx
            FILTER (?aspect1 != rdf:type && ?aspect1 != pos:boxerpos)
              OPTIONAL {?xx ?aspect11 ?xxx  FILTER (?aspect11 != rdf:type
                      && ?aspect11 != pos:boxerpos)}}}
       OPTIONAL {?e ?atheme ?x
          FILTER (?atheme = vnrole:Theme || ?atheme = vnrole:Theme1
                 || ?atheme = vnrole:Theme2 || ?atheme = boxer:theme)
             MINUS {?e ?agent1 ?a
                FILTER (?agent1 = vnrole:Agent || ?agent1 = boxer:agent
                       || ?agent1 = vnrole:Experiencer || ?agent1 = vnrole:Actor
                       || ?agent1 = vnrole:Actor1 || ?agent1 = vnrole:Actor2
                       || ?agent1 = vnrole:Cause)
                  FILTER (?x != ?a)}}
       OPTIONAL {?e ?patient ?y
          FILTER (?patient = vnrole:Patient || ?patient = vnrole:Patient1
                 || ?patient = vnrole:Patient2)
             OPTIONAL {?y ?aspect3 ?yy
               FILTER (?aspect3 != rdf:type && ?aspect3 != pos:boxerpos)
                 OPTIONAL {?yy ?aspect33 ?yyy  FILTER (?aspect33 != rdf:type
                        && ?aspect33 != pos:boxerpos)}}}
       OPTIONAL {?e ?ptheme ?y
          FILTER (?ptheme != ?atheme && (?ptheme = vnrole:Theme || ?ptheme = vnrole:Theme1
          || ?ptheme = vnrole:Theme2 || ?ptheme = boxer:theme))
             {?e ?agent1 ?a
                 FILTER (?agent1 = vnrole:Agent || ?agent1 = boxer:agent
                        || ?agent1 = vnrole:Experiencer || ?agent1= vnrole:Actor
                        || ?agent1= vnrole:Actor1 || ?agent1= vnrole:Actor2
                || ?agent1 = vnrole:Cause)
           FILTER (?y != ?a)}}
       OPTIONAL {?e ?oblique ?z
          FILTER (?oblique != vnrole:Topic &&  ?oblique != vnrole:Beneficiary
                 && ?oblique != vnrole:Patient && ?oblique != vnrole:Patient1
                 && ?oblique != vnrole:Patient2 && ?oblique != vnrole:Theme
         && ?oblique != vnrole:Agent && ?oblique != boxer:agent
         && ?oblique != vnrole:Experiencer && ?oblique != vnrole:Actor
         && ?oblique != vnrole:Actor2 && ?oblique != vnrole:Cause
```

```
         && ?oblique != boxer:theme && ?oblique != vnrole:Theme1
         && ?oblique != vnrole:Theme2 && ?oblique != rdf:type)
             OPTIONAL {?z ?aspect4 ?zz
               FILTER (?aspect4 != rdf:type && ?aspect4 != pos:boxerpos)
                 OPTIONAL {?zz ?aspect44 ?zzz  FILTER (?aspect44 != rdf:type
                          && ?aspect44 != pos:boxerpos)}}}
       OPTIONAL {?e ?aspect ?mod
          FILTER (?aspect = owl:sameAs || ?aspect = dul:hasQuality
                 || ?aspect = boxing:hasModality || ?aspect = boxing:hasTruthValue)
             OPTIONAL {?mod ?aspmod ?mod1
               FILTER (?aspmod!= rdf:type && ?aspmod != pos:boxerpos)
                 OPTIONAL {?mod1 ?aspmod1 ?mod2
                   FILTER (?aspmod1!= rdf:type && ?aspmod1 != pos:boxerpos)}}}
OPTIONAL {{?e ?r ?e1} {
           {?e1 rdf:type/rdfs:subClassOf* dul:Event}} FILTER (?e != ?e1)
               OPTIONAL {?e1 ?r1 ?e2 FILTER (?r1 != rdf:type && ?r1!= pos:boxerpos)
                  OPTIONAL {?e2 ?r2 ?e3 FILTER (?r2 != rdf:type && ?r2 != pos:boxerpos)}}}
OPTIONAL {?e rdf:type ?et . ?et rdfs:subClassOf+ ?et1
  OPTIONAL {?et1 rdfs:subClassOf+ ?et2}}
}
```

This query allows to build a new event-based sub-graph from the semantic graph generated by FRED. This sub-graph contains all mentioned events in the text with their participants and modifiers. In this section, we used this query to only extract event types from the text. The participants and modifiers will be extracted and used in the following sections. From our first example in the introduction, we are able to extract eight event types *{Get, Overlook, Recommend, Ask, Decorate, Have, Make, Separate}*.

5.3.2 Personal Events Identification

According to our definition in Section 5.2, we assume that events, which have the first or the second person pronoun (i.e. *I, You, We, Me, Us, My, Mine, our, ours, Your, Yours, ...*) as a participant, are the most important elements in lived experiences. Therefore, we keep those events, and use them in the identification task (Section 5.3.3): such events can have the user as a direct participant, i.e. the event has the author as its experiencer. For example, the event type *"Ask"* in our first review in Section 5.2, have two direct participants, one of them is the author of the text, who is identified by the first person pronoun *"I"* in the text, and the semantic class *"Person"* in the semantic graph produced by FRED. The triples which concern this event type are:

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

fred:ask_1
        rdf:type                fred:Ask ;
        vn.role:Agent           fred:person_3 ;
        vn.role:proposition     fred:room_2 .

fred:Ask
        rdfs:subClassOf         DUL:Event .

fred:person_3
        rdf:type                Person .
```

The triples indicate that the event occurrence `ask_1` is detected as a subtype of the `Ask` event type. The last one is modelled using the ontology `DOLCE` as subtype of the class DUL:Event. In addition, this event has two direct participants: `person_3` and `room_2`. The participant `person_3` is detected as a subtype of the semantic class `Person`.

However, other events can have the user as an indirect participants, i.e. the event has the author as a participant in one of their direct participants. For example, the event type `decorate_1` in our first review in Section 5.2, have four direct participants. These participants do not involve the author of the review. However, as we can show in the following triples, the direct participant `room_1` has a participant `person_1` which indicates the first pronoun *"My"* and identifies the author of the text.

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

fred:decorate_1
        rdf:type                fred:Decorate ;
        vn.role:Theme           fred:wallpaper_1 ;
        vn.role:Theme           fred:bed_1 ;
        vn.role:Destination     fred:room_1 .

fred:Decorate
        rdfs:subClassOf         DUL:Event .

fred:room_1
        fred:roomOf             fred:person_1 ;
        ...

fred:person_1
        rdf:type                Person .
```

In order to identify personal event types, we verified the participants until the third degree. If an event have the author of the review as a direct participant (i.e. participant of the first degree), e.g. the event *Ask*, or indirect participant (i.e. participant of the second or the third degree), e.g. the event *Decorate*, we considered it as a personal event. For example, from our first review in Section 5.2, we retain 3 event types *{Get, Ask, Decorate}*, which contain the user as a participant, and eliminate 5 event types *{Overlook, Recommend, Have, Make, Separate}*, which do not contain the user among their participants.

These event types will be then used in the following step of our approach in order to classify the user review and indicate whether if it contain lived experience contents or not.

5.3.3 Identification of Reviews containing LEs

The next step of our approach consists of identifying reviews which contain authentic lived experience, according to our definition of it. This step is very important for the extraction of lived experiences. It allows to filter out reviews which do not contain lived experiences and keep reviews which contain this information in order to extract them in the next step. In other word, applying firstly the classification task on the input reviews allows to perform

the extraction process on the reviews, which certainly contain lived experience contents, and to ignore the irrelevant reviews.

We viewed lived experience identification as a binary text classification task. We explored the use of machine learning techniques for identifying pertinent reviews, which consist of lived experiences content. This required the development of a set of annotated training examples, where user reviews are assigned "LivedExperience" or "Non-LivedExperience" labels. We identify lived experiences at the review entry level, unlike previous work that treated the subjectivity classification task at the segment or sentence level. We believe that this approach is more useful for the extraction task.

5.3.3.1 Annotating User Reviews

We followed a traditional text classification approach, in which a corpus of user reviews was hand-annotated (LivedExperience / Non-LivedExperience) to be used as a training and testing set. User reviews would be labeled as "LivedExperience" if an annotator judges that the review contains at least one lived experience.

We did not use any crowdsourcing services such as Amazon's Mechanical Turk (AMT), since in the experiment described in [Gordon and Swanson (2009)] (See Chapter 2, Section 2.1.3), the authors found that the vast majority of annotations produced through the AMT service were either completely random, or were generated by automated response engines, yielding unsuitable results. Since our annotation task resembles the task of [Gordon and Swanson (2009)], we feared that this kind of service would not give us good results, even though it has been described as effective to expert annotation for natural language processing tasks [Snow et al. (2008)].

Our solution was to annotate the corpus manually by an expert. Following our definition of what counted as lived experience, we annotated 383 user reviews, assigning the label "LivedExperience" or "Non-LivedExperience" to each. The label "LivedExperience" was assigned to 176 reviews (46%). To build our classifier, we used 268 reviews as a training set: 134 reviews with the "LivedExperience" class and 134 reviews with the "Non-LivedExperience" class. The remaining 115 reviews have been used for the test set, 42 reviews were assigned by the label "LivedExperience", and 73 reviews were assigned by the label "Non-LivedExperience".

5.3.3.2 Development of the Review Classifier

In Section 4.2.3, we viewed that text classification task is defined for the situation when there are m objects, each one belonging to one of the n classes, and a classification task would be to assign the belonging class to a new given object. A common approach for solving text classification problem is by training a supervised machine learning classifier such as support vector machines (SVM) [Cortes and Vapnik (1995)] or a Naïve Bayes (NB) [Duda et al. (1973)] classifier. The idea is given a set of training examples with their

associated labels, the training algorithm constructs a model which is able to predict the category of a new example.

Events that include the author as a participant are used as features to train a Support Vector Machine (SVM) and Naïve Bayes (NB) algorithms since they give good results in text classification tasks [Joachims (1998), Rish (2001)] (See Chapter 4 , Section 4.2.3 for more details about SVM and NB).

Using our training set, we were able to recognize and extract 1834 personal event types: 1374 are specific to the reviews containing lived experiences and 460 are specific to non-lived ones. The number of aggregated events is 415: 270 for the lived experiences reviews and 145 for the non-lived ones (Table 5.1).

rev.class	**personal events**	**aggregated personal events**
LivedExperience	1374	270
Non-LivedExperience	460	145
Overall	1834	415

Table 5.1: The number of extracted personal events from our training set for the two classes of review.

For more details, the number of common events which appear in the two classes is 95. Therefore, the number of features that we used in our classifier is 320 $(415 - 95)$ personal event types: 175 are specific to the reviews containing lived experiences, 50 are specific to non-lived ones, and 95 event types are detected in both classes.

The event types that are homogeneously distributed in the two classes are not useful in our classification task, and have been removed. We considered an event as representative to a class if the probability $P(c|e) \geq \sigma$ where:
$c \in C = \{$LivedExperience, Non-LivedExperience$\}$;
e: A generic event;
σ: A threshold that we determined empirically between 4 possibilities: $\{0.6, 0.7, 0.8, 0.9\}$. The best value of σ is 0.7 (See section 5.4.2.2).

For example, the personal event *Get* appeared 69 times in the "LivedExperience" reviews and 12 times in the "Non-LivedExperience" ones. It can be very useful to discriminate the "LivedExperience" reviews.

$$P(LivedExperience|\text{“}Get\text{”}) = \frac{69}{69+12} \simeq 0.85 > 0.7.$$
$$P(Non - LivedExperience|\text{“}Get\text{”}) = \frac{12}{69+12} \simeq 0.15 < 0.7.$$

However, the event *Hesitate* appeared 5 times in the "Non-LivedExperience" reviews. It did not appear in the "LivedExperience" ones. This event type is helpful to discriminate the "Non-LivedExperience" review:

$P(LivedExperience|\text{“}Hesitate\text{”}) = \frac{0}{0+5} = 0.$

$P(Non-LivedExperience|\text{“}Hesitate\text{”}) = \frac{5}{0+5} = 1 > 0.7.$

While, the personal event *Stay* frequented 117 times in the “LivedExperience” reviews and 77 times in the “Non-LivedExperience” ones. This event appears almost identically in the two classes and should be deleted since it cannot be useful for our classification.

$P(LivedExperience|\text{“}Stay\text{”}) = \frac{117}{117+77} = 0.6 < 0.7.$

$P(Non-LivedExperience|\text{“}Stay\text{”}) = \frac{77}{117+77} = 0.4 < 0.7.$

In the end, we removed 45 common events which do not help to discriminate the reviews types, and left 50 events among the events that exist in the two classes (all of them are for the “LivedExperience” reviews). By the end, the number of features that we are used in our classifier became 275 personal event types: 175 are specific to the reviews containing lived experiences, 50 commons events, but important to discriminate the lived experience reviews, 50 are specific to non-lived ones.

In table 5.2 and table 5.3 we show the most frequent event types in the “LivedExperience” and “Non-LivedExperience” reviews, respectively used to identify the class of user reviews. Table 5.4 shows some of common event types that we are deleted.

Event Type	$\mathbf{C}_{livedExp}(e)$	$C_{Non-livedExp}(e)$	$P(LivedExperience\|e)$
Have	156	38	0.80
Get	69	12	0.85
Go	31	11	0.74
Walk	30	5	0.86
Check	21	4	0.84
Give	19	2	0.90
Pay	18	1	0.95
Upgrade	17	0	1
Arrive	14	4	0.78
Ask	14	4	0.78
Eat	14	2	0.88
Use	14	3	0.82
Spend	10	3	0.77
Visit	7	0	1
Decorate	2	0	1

Table 5.2: Most frequent personal event types used for classifying “LivedExperience” reviews

Event Type	$C_{livedExp}(e)$	$C_{Non-livedExp}(e)$	$P(Non-LivedExperience\|e)$
Hesitate	0	5	1
Cannot	0	2	1
Mean	0	2	1
Separate	0	2	1
Admire	0	1	1
Alert	0	1	1
Become	0	1	1
Ensure	0	1	1
Figure	0	1	1
Imagine	0	1	1
Refer	0	1	1
Vacation	0	1	1

Table 5.3: Most frequent personal event types used for classifying "Non-LivedExperience" reviews

Event Type	$C_{livedExp}(e)$	$C_{Non-livedExp}(e)$
Stay	117	77
Recommend	25	12
Book	22	12
Say	13	10
Return	12	8
Call	8	4
Expect	8	12
Do	7	5
Greet	6	3
Travel	6	8
Bring	4	2

Table 5.4: Some removed personal event types

5.3.4 Event Filtering

In Section 5.3.3.1, we proposed to annotate a review with the label "LivedExperience" if it contains at least one lived experience. Therefore, a lived experience review may contain both lived experience and non-lived experience contents. The personal events identification task in Section 5.3.2 allowed us to filter out the events which do not contain the user as a participant, and keep the personal events which have the user as a participant. However, some personal events could not represent lived experience even if they have the author of the review as a participant. These irrelevant personal events should be also filtered out in order to only keep the relevant personal events and use them to extract the authentic lived experiences.

Therefore, we employed a filtering algorithm, which allows to study personal events by taking into account the impact of their neighbors in the same review to finally select the

relevant ones. Event neighbors are supposed to have a big effect on the target event. We considered that two events are neighbors if they are in the same reviews.

The algorithm 1 takes a lived experience review as input, and gives the label of each personal event as output. We considered two labels: "LivedExperience" for relevant events, and "NonLivedExperience" for irrelevant ones.

First, for each personal event in each input review, we calculate its distribution in each class of training set reviews used in our classification task. If an event was distributed homogeneously in the two classes, we deleted it from the input review (line 3 to line 9). We considered that the event distribution is homogeneous if it is not higher than 0.7 in one of the two classes of reviews (See Section 5.3.3.2). For example, the sentence ***"I recommend the Talbott hotel"*** represents, according to our definition, a lived experience content since it contains the event ***"Recommend"*** with a participant ***"I"***, which represents the author of the text. However, this segment does not indicate lived experience since the author gives a general opinion without indicating any specific information. The first step of our filtering algorithm is able to filter out this personal event since it distributes in the two classes of reviews in manner homogeneous. As we can see in table 5.4, this event appeared 25 times in the "LivedExperience" reviews of our training set and 12 times in the "Non-LivedExperience" ones of the training set (i.e. the distribution of this event is $\frac{25}{25+12} \simeq 0.68 < 0.7$ in the "LivedExperience" reviews, and $\frac{12}{25+12} \simeq 0.32 < 0.7$).

Second, we calculate the function $F = \alpha X(\mathbf{e}_j) + \beta Y(\mathbf{e}_j)$ for the remaining events, where α is the value that was weighted to target events, β presents the value that was weighted to event neighbors. We choose α to be equal to the number of event neighbors divided by the total number of personal events, and β is 1/the number of personal events.

$$\alpha = \frac{|R_i| - 1}{|R_i|}$$

$$\beta = \frac{1}{|R_i|}$$

where R_i is the number of personal events in the input reviews.

The value of α is much larger than the β value since it was weighted to the target event. These values stay the same and do not change for each event in the same review. $X(\mathbf{e}_j)$ represents the bigger distribution value of the target event in the two classes of reviews in our training set. $Y(\mathbf{e}_j)$ is the sum of the bigger distributions values of event neighbors in the two classes of reviews in our training set. We assumed that the distribution value for an event, target or neighbors should be positive if the event is distributed significantly in lived experience reviews from our training set, and negative otherwise.

Finally, the event is classified according to the value of the function F (line 21 to line 24). If F value is higher than 0, the target event represents a lived experience.

Algorithm 1: Event Filtering

Input: $\mathcal{R} = \{R_1, R_2, ..., R_k\}$ such as $R_i = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_n\}$, α, β

Output: $\mathcal{L} = \{(\mathbf{e}_1, l_1), (\mathbf{e}_2, l_2), ..., (\mathbf{e}_n, l_n)\}$

$\alpha = 0$, $\beta = 0$

foreach *Review* $R_i \in \mathcal{R}$ **do**

  foreach *event* $\mathbf{e}_j \in R_i$ **do**

    $P(LivedExperience|\mathbf{e}_j) = \frac{frequency(\mathbf{e}_j), \mathbf{e}_j \in TS^+}{frequency(\mathbf{e}_j), \mathbf{e}_j \in TS}$, such as TS^+ is the set of "Lived Experience" reviews of our training set TS

    $P(Non\text{-}LivedExperience|\mathbf{e}_j) = \frac{frequency(\mathbf{e}_j), \mathbf{e}_j \in TS^-}{frequency(\mathbf{e}_j), \mathbf{e}_j \in TS}$, such as TS^- is the set of "Non Lived Experience" reviews of our training set TS

    if $P(LivedExperience|\mathbf{e}_j) < 0.7$ *and* $P(Non\text{-}LivedExperience|\mathbf{e}_j) < 0.7$ **then**

      delete $\mathbf{e}_j$ from R_i: $R_i = R_i - \mathbf{e}_j$

  $\alpha = \frac{|R_i|-1}{|R_i|}$, such as $|R_i|$ is the cardinality of the set of events R_i

  $\beta = \frac{1}{|R_i|}$

  foreach *event* $\mathbf{e}_j \in R_i$ **do**

    if $P(LivedExperience|\mathbf{e}_j) > 0.7$ **then** $\mu_j = +P(LivedExperience|\mathbf{e}_j)$;

    else if $P(Non\text{-}LivedExperience|\mathbf{e}_j) > 0.7$ **then**

      $\mu_j = -P(Non\text{-}LivedExperience|\mathbf{e}_j)$

    Calculate $X(\mathbf{e}_j)$, the bigger distribution value of the target event in the two classes of reviews in our training set.

    $X(\mathbf{e}_j) = \mu_j$

    Calculate $Y(\mathbf{e}_j)$, the sum of the bigger distributions values of event neighbors in the two classes of reviews in our training set

    $Y(\mathbf{e}_j) = \sum_{\mathbf{e}_v \in R_i / \mathbf{e}_j} X(\mathbf{e}_v)$

    Calculate $F = \alpha X(\mathbf{e}_j) + \beta Y(\mathbf{e}_j)$

    if $F > 0$ **then return** ($\mathbf{e}_j$, "*LivedExperience*");

    else

      return ($\mathbf{e}_j$, "*NonLivedExperience*")

The review provided as an example in Section 5.1 has three personal event types *{Get, Ask, Decorate}*. These events are labeled as lived experience events by the first step in our algorithm, since their distribution in the list of lived experience events in our training set is clearly larger than their distribution in the non-lived experience list. The next step is to calculate the function F to decide whether these events are pertinent or not. The value of α in this example is equal to $\frac{3-1}{3} = 0.67$. However, $\beta = \frac{1}{3} = 0.33$. For the target event *{Get}*, the event neighbors will be *{Ask, Decorate}*. Therefore,

$P(LivedExperience|\text{"}Get\text{"}) = \frac{69}{69+12} \simeq 0.85 > 0.7.$

$P(Non - LivedExperience|\text{"}Get\text{"}) = \frac{12}{69+12} \simeq 0.15 < 0.7.$

X(Get) = 0.85.

$P(LivedExperience|\text{“}Ask\text{”}) = \frac{14}{14+4} \simeq 0.78 > 0.7$.
$P(Non-LivedExperience|\text{“}Ask\text{”}) = \frac{4}{14+4} \simeq 0.22 < 0.7$.
X(Ask) = 0.78.

$P(LivedExperience|\text{“}Decorate\text{”}) = \frac{2}{2+0} \simeq 1 > 0.7$.
$P(Non-LivedExperience|\text{“}Decorate\text{”}) = \frac{0}{2+0} \simeq 0 < 0.7$.
X(Decorate) = 1.

Y(Get) = 0.78 + 1 = 1.78.

F(Get) = 0.67 * 0.85+ 0.33 * 1.78 = 1.16

Therefore, this event represents a lived experience. Table 5.5 shows the obtained results for each personal event in our example. According to their F values, all these events represent lived experiences.

Event	$P(LivedExp)$	$P(Non\text{-}LivedExp)$	F
Get	0.85	0.15	$\frac{2}{3} * (0.85) + \frac{1}{3} * (0.78 + 1) = 1.16 > 0$
Ask	0.78	0.22	$\frac{2}{3} * (0.78) + \frac{1}{3} * (0.85 + 1) = 1.13 > 0$
Decorate	1.00	0.00	$\frac{2}{3} * (1) + \frac{1}{3} * (0.85 + 0.78) = 1.2 > 0$

Table 5.5: Results for the event filtering algorithm.

5.3.5 Event participant extraction

A lived experience should involve events and their participants, and the narrator should be among the participants. For each lived experience event, we need to identify its participants, and to build a lived experience graphs. The participants in an event are the arguments of semantic roles (e.g. Agent, Patient, Oblique, Theme, etc.) associated with the event (See Section 4.2.2.3).

Applying the SPARQL query, which we have used in Section 5.3.1, allows to extract event participants from FRED's knowledge graphs for the input review. We are only interested in lived experience events, which are detected by our filtering algorithm in the precedent section. Therefore, we modified this SPARQL query in order to only extract participants of lived experience events. We added a filter condition, which allows to only query the relevant events with their participants and modifiers as follows:

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX ...
...
CONSTRUCT {
     ?e :agent ?x .  ?x ?aspect1 ?xx . ?xx ?aspect11 ?xxx .
     ?e :patient ?y . ?y ?aspect3 ?yy . ?yy ?aspect33 ?yyy .
     ?e ?oblique ?z .  ?z ?aspect4 ?zz.  ?zz ?aspect44 ?zzz .
     ?e ?r ?e1  .  ?e1 ?r1 ?e2 . ?e2 ?r2 ?e3 .
     ?e ?aspect ?mod . ?mod ?aspmod ?mod1 . ?mod1 ?aspmod1 ?mod2
         }
   WHERE{
       FILTER ((regex(str(?e),"event_1","i")) || (regex(str(?e),"event_2","i"))
              || (regex(str(?e),"event_n","i")))
       {?e rdf:type/rdfs:subClassOf* dul:Event}
       UNION {?e rdf:type/rdfs:subClassOf* schemaorg:Event}
       UNION {?e rdf:type/rdfs:subClassOf* d0:Event}
       OPTIONAL {?e ?agent ?x
          FILTER (?agent = vnrole:Agent || ?agent = boxer:agent
                 || ?agent = vnrole:Experiencer || ?agent =   vnrole:Actor
                 || ?agent = vnrole:Actor1 || ?agent = vnrole:Actor2
                 || ?agent = vnrole:Cause || ?agent = vnrole:Theme
                 || ?agent = vnrole:Topic || ?agent = vnrole:Beneficiary
                 || ?agent = vnrole:Actor || ?agent = vnrole:Cause)
          OPTIONAL {?x ?aspect1 ?xx
            FILTER (?aspect1 != rdf:type&& ?aspect1 != pos:boxerpos)
              OPTIONAL {?xx ?aspect11 ?xxx FILTER (?aspect11 != rdf:type
                      && ?aspect11 != pos:boxerpos)}}}
...
...
...
   }
```

`event_1, event_2,..., event_n` in the added FILTER clause represent the lived experience events. Therefore, this SPARQL query generates an event sub-graph containing lived experience events, detected by the filtering algorithm, with their direct or indirect participants. A direct participant ${}_{d}p_i\ \phi\ e_i$ is an argument of an event e_i. An indirect participant ${}_{i}p_j\ \psi\ e_i$ is a direct participant of an event e_j that on its turn occurs as a direct participant of e_i (ϕ and ψ are semantic roles).

To improve our extraction method, and keep our lived experience graphs more informative, we extract event participants for the first, second and third degree. The participants of the first degree represent the direct participant since they are associated directly with the events. However, the participants of the second or the third degree represent the indirect participants. In addition, we take advantage of FRED knowledge graphs by extracting *event modifiers* such as *modality, negation, and adverbial qualities*, which enrich relevant events with additional semantic information that enables to distinguish the nuances or polarity of the reported events. These modifiers will be also extracted for the first, second, and third degree (See Chapter 3).

For example, the generated event-subgraph for the personal event *"Get"* in our example are represented by the following RDF/XML triples:

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
```

```
fred:get_2
        vn.role:Agent              fred:person_6 ;
        vn.role:Theme              fred:voucher_1 ;
        dul:associatedWith         fred:book_1 ;
        boxing:hasModality         boxing:Necessary ;
        fred:upon                  fred:arrival_1 .

fred:voucher_1
        fred:for                   fred:free-drink_1 ;
        :hasDeterminer             quantifier:a .

fred:book_1
        dul:hasQuality             fred:Directly .
```

These triples indicate that the event `get_2` has four direct participants `{person_6, voucher_1, book_1, arrival_1}`, two indirect participants `free-drink_1`, and two event modifiers: a modality `Necessary`, and a quality `Directly`.

Using our training set, we were able to recognize and extract 4751 personal events and event participants: 3534 are specific to the reviews containing lived experiences and 1217 are specific to non-lived ones. The number of aggregated events and participants is 1211: 810 for the lived experiences reviews and 401 for the non-lived ones (Table 5.6).

rev.class	features	aggregated features
LivedExperience	3534	810
Non-LivedExperience	1217	401
Overall	4751	1211

Table 5.6: The number of extracted personal events and their participants from our training set for the two classes of review: **features** represents personal events & personal event participants

For more details, the number of common events and their participants, which appear in the two classes is 248. Therefore, the number of features that we are used in our classifier is 963 $(1211 - 248)$ personal event types: 562 are specific to the reviews containing lived experiences, 153 are specific to non-lived ones, and 248 events and event participants are detected in both classes. As we mentioned in Section 5.3.3.2, some common features, which are homogeneously distributed in the two classes of reviews, are not useful in the identification task. Therefore, we removed them. Using the probability $P(c|e) \geq 0.7$, we removed 134 common events and event participants and kept 114 ones. By the end, the number of features that we are used in our classifier became 829 $(963-134)$ personal events and participants: 562 are specific to the reviews containing lived experiences, 109 commons events but important to discriminate the lived experience reviews, 153 are specific to non-lived ones, and 5 commons events but important to discriminate the Non-lived experience reviews.

5.3.6 Extraction of sentences containing lived experiences

After the previous steps, we are ready to extract lived experience sentences, which consists of the following steps:

- Input Review Segmentation: We divided the input review into its component sentences using punctuation.
- Sentence Ranking: For each relevant event, all review sentences are ranked according to the presence of the relevant event and its participants. We imposed two conditions in this step:
  1. The sentence must have at least one word evoking an event.
  2. The rank should be larger than 2 in order to retrieve an informative sentence.

  In other words, the sentence should have at least one relevant event, and at least 2 participants, where one of them is the author.
- Finally, the sentence with the highest rank is extracted.

The matching between a sentence contents, and events and their participants was performed using the annotations which were provided by FRED. Additional triples are generated by FRED in order to provide annotations that link text fragments to their corresponding graph elements. These annotations are expressed by means of the Earmark [Peroni et al. (2011)], the NLP Interchange Format (NIF) [Hellmann et al. (2013)], and semiotics.owl[6] vocabularies. Text fragments are represented as offsets. For example, some of the annotations which concern the event `get_2` in the semantic graph of FRED can be shown in follows:

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >

fred:offset_537_540_get
        semiotics:denotes     fred:get_2 .

fred:offset_491_494_you
        semiotics:denotes     fred:person_6 .

fred:offset_543_550_voucher
        semiotics:denotes     fred:voucher_1 .

fred:offset_500_508_directly
        semiotics:hasInterpretant     fred:Directly .
```

These annotations show that the text fragment of the event `get_2` is `get` in the original text. `person_6` is the pronoun `you`, etc. Therefore, using the semiotics.owl properties `semiotics:denotes`, and `semiotics:hasInterpretant`, we are able to retrieve the text fragment for each lived experience event and their participants.

[6] `http://www.ontologydesignpatterns.org/cp/owl/semiotics.owl`

The following shows the lived experience sentences extracted from our first example in Section 5.1:

1. *I asked for a room overlooking the pantheon and I got it.*
2. *My deluxe room was number 32, and was tastefully decorated with a classic and beautiful Pierre Frey wallpaper, and an extra day bed.*
3. *If you book directly through the hotel, you'll get a voucher for a free-drink upon arrival.*

5.3.7 Lived Experience Graph Representation

We represent lived experiences as event sub-graphs containing relevant events, with their participants, and modifiers. In order to create a compact representation of lived experiences in customer reviews, we encode lived experiences as sequences of events, where each event is represented as an n-tuple of participants. For example, for the first example review, the encoding is as follows:

$Ask(\text{“}Person\text{”}, \text{“}Room\text{”}[\text{“}A\text{”}])$.
$Get(\text{“}Person\text{”}, \text{“}Necessary\text{”}, \text{“}Arrival\text{”}, \text{“}Book\text{”}[\text{“}Direcly\text{”}],$
$\text{“}Voucher\text{”}[\text{“}A\text{”}, \text{“}Free-Drink\text{”}])$.
$Decorate(\text{“}Testefully\text{”}, \text{“}Bed\text{”}, \text{“}Room\text{”}[\text{“}Person\text{”}, \text{“}Multiple\text{”}, \text{“}Deluxe\text{”}, \text{“}32\text{”},$
$\text{“}Hotel\text{”}[\text{“}This\text{”}, \text{“}4\text{”}]], \text{“}Wallpaper\text{”}[\text{“}Beautiful\text{”}, \text{“}Classic\text{”}])$.

Figure (5.3), shows a diagram of the lived experience graph from that review:

5.4 Experimental Evaluation

5.4.1 Dataset

[Ott et al. (2011)] have recently created the first publicly available[7] dataset for deceptive opinion spam research. This dataset contains 800 positive reviews (400 truthful reviews and 400 fake reviews), which have been assigned with 5-stars in the open system ranking and 800 reviews for the negative polarity (400 truthful reviews and 400 fake reviews), which have 1-star in the open system ranking. In this work, we are only interested in the positive truthful reviews which are collected from the 20 most popular Chicago hotels on TripAdvisor[8]. We have selected 383 user reviews to be annotated for the training and test set. 17 reviews were excluded because of parsing problems. As we mentioned in Section 5.3.3.1, the label “LivedExperience” was assigned to 176 reviews (46%) and 207 reviews

[7] Available by request at: `http://www.cs.cornell.edu/~myleott/op_spam`
[8] `http://www.tripadvisor.com/`

Figure 5.3: An event sub-graph representing lived experiences extracted from our first user review in Section 5.1.

were assigned by the label "Non-LivedExperience". 268 user reviews have been used as a training set (134 user reviews for the "LivedExperience" reviews and 134 user reviews for the "Non-LivedExperience" ones) for the lived experience identification task, and 115 as a test set (42 user reviews for the "LivedExperience" reviews and 73 user reviews for the "Non-LivedExperience" ones).

We included 50% lived experience reviews and 50% Non-lived experience reviews in our training set in order to build a good event dictionary containing extracted personal events from the same number of reviews for the two classes.

5.4.2 Review identification evaluation

As described, we have used the extracted personal events as a collection of features to train two models: (1) Support Vector Machine (SVM) [Cortes and Vapnik (1995)] model

rev.class	Training Set	Test Set	Total
LivedExperience	134	42	176
Non-LivedExperience	134	73	207
Total	268	115	383

Table 5.7: Characteristics of datasets used in our experiment

with a radial basis function kernel, (2) Naïve Bayes [Duda et al. (1973)] , as our classification models. SVM and NB are very well suited for binary classification tasks, and they have the ability to deal with large space features. Since we were looking at entire reviews rather than segments of sentences, we have also considered event type frequency. Each review is transformed into a feature vector where the $i-$th component value is the frequency of the $i-$th feature in the review, and then classified as either "LivedExperience" or "Non-LivedExperience". We chose the Weka[9] implementation for the classifier using cross-validation techniques [Olson and Delen (2008)] to validate the model (in particular, we used 10-fold cross-validation).

Our system is evaluated in terms of precision (P), recall (R), and F-measure (F) (See Section 4.3.3). We performed 10-fold cross-validation on the training set, and yielded good results, see Table (5.8), and Table (5.9).

To validate the obtained results, we applied our review classifier to the test set containing 115 user reviews. Table (5.8), and Table (5.9) show the result.

As lived experiences consist of personal events, they should also contain their participants. We assumed that event participants can upgrade the identification task results. By using events and their participants as features in our classifier, we obtained very close results to the previous ones that were obtained using only personal events. However, we noticed a small decrease for both training set and test set, as shown in Table Table (5.8), and Table (5.9).

		Training Set			Test Set		
Features	$Nb\_Features$	P	R	F	P	R	F
All_Eve	320	84.3%	84%	84.1%	83.4%	83.5%	83.4%
$Eve_\sigma = 0.7$	275	**86.5%**	**85.8%**	**86.1%**	**85.1%**	**85.2%**	**85.2%**
All_Eve_Part	963	82.7%	81.7%	82%	82.6%	82.6%	82.6%
$(Eve\text{-}Part)\sigma = 0.7$	829	84.4%	83.2%	84%	79%	79.1%	79%

Table 5.8: Overall results for review identification using the method SVM

These tables show the results of our experiments in both training and test sets using four configurations to train two classifiers (SVM and NB): All_Eve presents the results of using all extracted personal events, without deleting the common events, as features, $Eve_\sigma = 0.7$ presents the achieved results by deleting the common events and use the rest

[9] http://www.cs.waikato.ac.nz/ml/weka/

		Training Set			Test Set		
Features	*Nb_Features*	P	R	F	P	R	F
All_Eve	320	82%	81.7%	81.8%	78.5%	78.3%	78.4%
$Eve_{\sigma} = 0.7$	275	**86**%	**86.2**%	**86.1**%	**82.8**%	**81.7**%	**82.2**%
All_Eve_Part	963	78.7%	78.7%	78.7%	73.2%	73%	73.1%
(*Eve-Part*)$\sigma = 0.7$	829	82.1%	82.1%	82.1%	77.4%	77.4%	77.4%

Table 5.9: Overall results for review identification using the method NB

as features, *All_Eve_Part*, presents the obtained results using all personal events and their participants as features, and (*Eve-Part*)$\sigma = 0.7$ indicates the obtained results after deleting some common events and event participants.

As shown in these tables, SVM performs better than NB for all the used configurations in both training and test set. $Eve_{\sigma} = 0.7$ achieves the best results for both training and test sets in our two classifiers. In addition, using all personal events (without deleting the common ones) decreases the performance by about 2% for both training and test sets using SVM classifier and about 4% for both training and test sets using NB classifier. Furthermore, adding participant features to all personal events gives good results, but less effective than the results obtained using personal event features with $\sigma = 0.7$ in our two classifiers. These features decrease the performance, which obtained using $Eve_{\sigma} = 0.7$ by about 4% in the training set and 1% in the test set for SVM classifier, and by about 7% in the training set and 9% for the test set for NB classifier. Besides, deleting common features from the dictionary that contain events and their participants give approximately the same results using all the personal events in the training set, but for the test set we notice some decrease.

Based on this evaluation, we use personal event features and the classifier SVM in the LEE system (see Section 5.4.3) to identify the input reviews with optimal accuracy. In addition, we can notice that the results in our test set are very close to our original cross-validation evaluation. This allows us to conclude that there is no evidence that the results of our classifier may get worse when applied to different user reviews.

5.4.2.1 Comparison to other approaches

In order to meaningfully evaluate our model, we have established a reasonable baseline. We chose to compare our method against three baseline models: (1) The model from [Gordon and Swanson (2009)], called "personal stories", (2) a verb model, and (3) a bag-of-words model.

As we mentioned in Chapter 2 (Section 2.1.3), the closest study to our approach on personal information identification task is [Gordon and Swanson (2009)]. They employ statistical text classification technology on the content of blog entries to identify personal stories in weblog entries. They investigated several variations of n-gram features (e.g. unigrams and bigrams) to train a Support Vector Machine learning algorithm (SVM).

They manually annotated 4252 weblog entries to be used as a training set, and then performed a 10-fold cross validation. Their system achieved precision = 66%, recall = 48%, and F-Measure = 55% on this data. To compare our system to their system, we used their pre-trained model[10], and tested it on our training and test set. The achieved results for this set can be shown in Table 5.10.

The verb model consists in using just verbs, which are extracted from our training set as features for a SVM classifier to discriminate between the two classes of reviews. We chose to compare our method to this model, since most of our event features are represented by verbs. To extract verbs from user reviews, we used the semantic graph of FRED which contain syntactic part-of-speech annotations. We applied the SPARQL query, which are used in Chapter 4 (Section 4.3.6) and allows to retrieve verbs, which have the POS "v", from the semantic graph produced by FRED.

The overall performance on the training set using 10-cross validation and on the test set is shown in Table 5.10.

We also compared our model to another model which only uses bags of words as classifier features. We extracted all the words from our training reviews, and used them to train a Support Vector Machine classifier. We used this model in order to compare our results using only lived experience events, to results obtained using all review contents. Table 5.10 presents all the results (including our own) on the training set using 10-fold cross validation.

	Training Set			Test Set		
Model	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Personal stories	59.8%	59.3%	59.6%	65.4%	55%	60%
Verbs	66.9%	66.7%	66.8%	75.9%	75.7%	75.8%
Bag-of-words	65%	63.3%	64.1%	69.4%	69.4%	69.4%
$Event_{\sigma} = 0.7$	**86.5%**	**85.8%**	**86.1%**	**85.1%**	**85.2%**	**85.2%**

Table 5.10: Overall results for lived experience identification

As shown in this table, we outperform other models by nearly 26% against the personal stories model, about 10% against the verb model, and 17% against the bag-of-words model on the test set.

5.4.2.2 σ Estimation

To properly build and have a classifier allowing to discriminate the two types of reviews, we considered that an event is representative with respect to a class if the probability $P(c|e) \geq \sigma$. To determine the best value of σ, we tested 4 possibilities {0.6, 0.7, 0.8, 0.9}. Table 4.10 and Table 4.11, respectively, show the results of our experiment using Support Vector Machine (SVM) and Naïve Bayes (NB) methods with *10 fold cross-validation*

[10] https://github.com/asgordon/StoryNonstory

technique in the training with the 4 values of σ.

σ	**Precision**	**Recall**	**F-Measure**
0.6	82.1%	81.7%	81.9%
0.7	**86.5%**	**85.8%**	**86.1%**
0.8	84%	83.6%	83.8%
0.9	84.2%	79.9%	82%

Table 5.11: Results for the *10-fold cross* experiments for several values of σ for our training set using SVM method

σ	**Precision**	**Recall**	**F-Measure**
0.6	84%	84%	84%
0.7	**86%**	**86.2%**	**86.1%**
0.8	84.7%	84.3%	84.5%
0.9	82.1%	76.1%	79%

Table 5.12: Results for the *10-fold cross* experiments for several values of σ for our training set using NB method

These tables show that the best performance for review identification task using personal event features can be achieved using $\sigma = 0.7$ for both SVM and NB methods. This justifies our choice of σ as 0.7 to delete some common features and improve the results of our classifiers (Section 5.3.3.2).

5.4.3 Lived experience extraction evaluation

We evaluate the Lived Experience Extraction task using 80 reviews from our dataset. We chose reviews that are classified as lived experience reviews by our system, and are equally annotated by the annotator. In total, 790 sentences are used in the evaluation.

Based on those reviews, 2 human evaluators have been asked to manually extract sentences that denote lived experiences. Human evaluators took one week to perform this task. For this reason, we only chose 80 reviews for the evaluation. Each annotator classifies each of the 790 sentences into two classes (Yes, No). The class "Yes" indicates that the sentence represents lived experience. However, the class "No" indicates that the sentence does not denote lived experience. The first annotator was assigned the label "Yes" to 204 sentences and the label "No" to 586 sentences. However, the second annotator was labeled 176 sentences by the label "yes" and 614 sentences by the label "No". The confusion matrix in Table 5.13 summarizes the annotation results for each annotator.

In order to measure the inter-annotator agreement among the two judges, we used the Kappa coefficient (Cohen's kappa) [Cohen (1960)]. This coefficient (K) is defined by:

		Annotator_1	
		Yes	No
Annotator_2	Yes	175	1
	No	29	585

Table 5.13: Results of human annotation for lived experience extraction

$$K = \frac{p_o - p_e}{1 - p_e}$$

where p_o is the relative observed agreement among annotators (i.e. the proportion of items where there is agreement), and p_e is the probability of random agreement (i.e. the proportion of units which would be expected to agree by chance). For Table 5.13, we get:

$$p_o = \frac{175+585}{790} = 0.96$$

$$p_e = \frac{\frac{(175+1)*(175+29)}{790} + \frac{(29+585)*(1+585)}{790}}{790} = 0.63$$

$$K = \frac{0.96-0.63}{1-0.63} = 0.89$$

The agreement between the two annotators is 0.89. In order to interpret this value, we used the interpretation of Kappa which are proposed by [Landis and Koch (1977)]. In Table 5.14, we show this interpretation.

Kappa Value	**Strength of Agreement**
< 0.00	Poor
$0.00 - 0.20$	Slight
$0.21 - 0.40$	Fair
$0.41 - 0.60$	Moderate
$0.61 - 0.80$	Substantial
$0.81 - 1$	Perfect

Table 5.14: The Kappa interpretation according to [Landis and Koch (1977)]

This table shows that K-Kappa is always less than or equal to 1. A value of 1 implies perfect agreement and values less than 1 imply less than perfect agreement. According to [Landis and Koch (1977)], our K-Kappa value is in the range $(0.81 - 1.00)$, which corresponds to "perfect agreement". Therefore, our operational definition of what is a lived experience in Section 5.2 could be considered efficient and could be used to identify and extract authentic lived experiences from user reviews.

For each review, we applied our system to extract lived experience sentences. All the results extracted by our system are compared to the manually extracted results. In

addition, we also compared our results to a virtual meta-annotator. The $A1 \cap A2$ meta-annotator extracts lived experience sentences when both annotators believe the sentence to denote a lived experience.

Table 5.15 gives the precision (P), recall (R), and F-measure (F) results of our system. We calculated these values at sentence level. For each sentence in the review, we consider it a "*TP*" if it was labeled as lived experience by both the system and the human annotator, "*FP*" if the system labeled it as a lived experience, but not the annotator, "*FN*" if the annotator labeled the sentence as a lived experience, but the system failed to recognize it, and "*TN*" if it was labeled as Non-lived experience by both the system and the human annotator.

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F = \frac{2 * P * R}{P + R}$$

	Precision	**Recall**	**F-measure**
A1	0.55	0.70	0.62
A2	0.48	0.70	0.57
$A1 \cap A2$	0.49	0.74	0.58

Table 5.15: The results of lived experience extraction task.

In this table, column 1 identifies the annotator that we compared our system with. The results indicate that our system is closer to the first annotator than the second one.

The $A1 \cap A2$ meta-annotator found 167 lived experience sentences and 623 Non-lived experience sentences. Using our system, we extract 255 lived experience sentences and 535 Non-lived ones. The "*TP*" is 124 sentences, the "*FP*" is 131 sentences, the "*FN*" is 43, and the "*TN*" is 492.

5.5 LEE as a RESTful API

The LEE system is available as a web application and RESTful web service[11] featuring a graphical user interfaces (Figure 5.4), and provides reusable RDF knowledge graphs e.g. for recommendation services.

[11] `https://lipn.univ-paris13.fr/ClientProj/client.jsp`

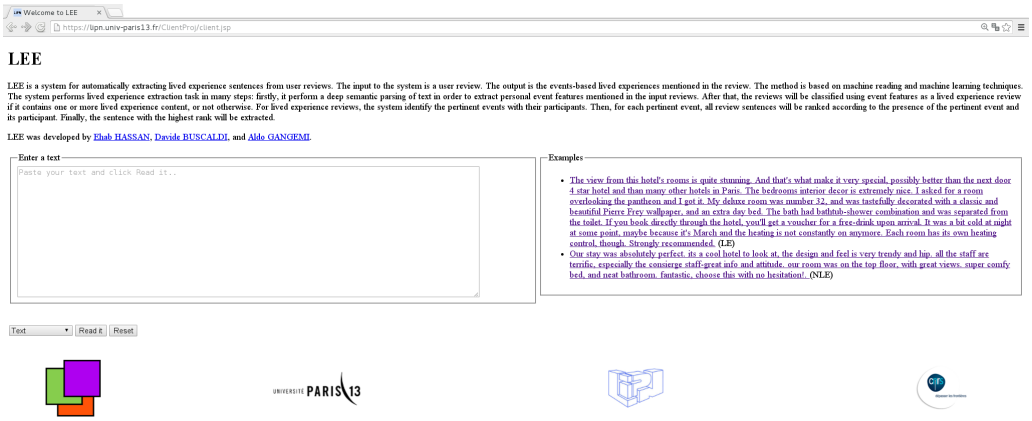


Figure 5.4: LEE RESTful Interface

REST [Fielding (2000)] stands for REpresentational State Transfer. REST is web standard based architecture and uses HTTP protocol for data communication. It revolves around resource where every component is a resource and a resource is identified by global ID - typically using URI and is accessed by a common interface using HTTP standard methods. REST offers several HTTP methods, e.g.:

- GET: provides a read only access to a resource.
- POST: used to create a new resource.
- PUT: used to modify an existing resource.
- DELETE: used to remove a resource.
- OPTIONS: used to get the supported operation on a resource.

A web service, as defined by [Gottschalk et al. (2002)], is a collection of open protocols and standard used for exchanging data between applications or systems. Web services based on REST architecture are known as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture.

The LEE API is implemented as a JAVA/JAVA EE software using the Jersey framework. Jersey provides a servlet which analyzes the incoming HTTP request and selects the correct class and method to respond to this request. More specifically, LEE API processes HTTP GET requests in order to query the LEE system with a user-specified review text. Figure 5.5 shows the LEE API pipeline.

We found in Section 5.4.2 that the LEE system achieves the best performance for the review filtering task using personal event features for SVM classifier, in particular with a radial basis function kernel. Therefore, in this pipeline, the LEE system extracts personal events and uses them with the SVM model that we described in Section 5.4.2 to predict whether the input review contains lived experiences or not. If the input review contains

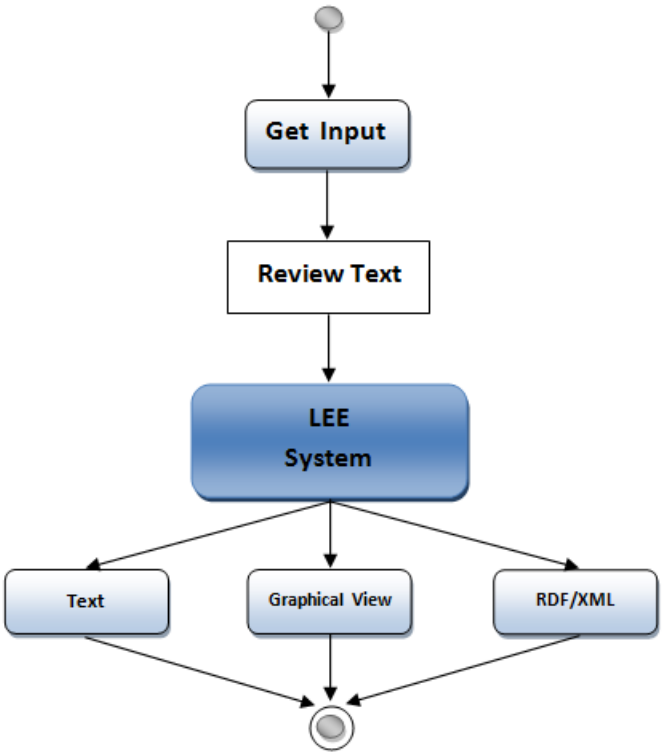


Figure 5.5: LEE RESTful Pipeline.

lived experiences contents, the system start the extraction task (See 5.4.3). The response of the GET request will be the output of LEE system which could be represented in three types:

- Text: this type shows the text segments which represent lived experience contents, (e.g. the three lived experience sentences in Section 5.3.6).
- Graphical View: this type allows to show the extracted live experiences in the form of semantic graphs, (e.g. figure 5.3).
- RDF/XML: using this type, we represent the results as a set of RDF triples.

If the input review does not contain lived experience contents, the output will be "This review do not have any Lived Experience".

5.6 Conclusion

In this chapter, we defined lived experiences and we proposed a set of techniques for identifying and extracting them from user reviews.

Our experimental results indicate that the proposed techniques are effective in identifying lived experiences. The results in the review identification task show that personal events can be used to discriminate reviews which contain lived experience content. In

addition, the results in the extraction task show that our techniques are very promising, especially considering that according to our knowledge, no method has yet attempted to extract lived experiences from user reviews.

We believe that this problem will become more and more important, considering the growing amount of user-opinion-based decisions made on the Web. Extracting lived experiences from user reviews is useful for potential users, due to the ability to take into account the cognitive or emotional identification of the reader with the author of a review, either for establishing non-fictional experiences, or for filtering reviews based on the closeness in taste or life habits.

The work presented in this chapter has resulted in the following publication:

- Ehab Hassan, Davide Buscaldi, and Aldo Gangemi. Event-based recognition of lived experiences in user reviews. In Knowledge Engineering and Knowledge Management: 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings 20, pages 320-336. Springer, 2016.

Chapter 6

Applications

Contents

6.1 Correlating open rating systems and lived experiences extraction from text . . . 111
6.1.1 Introduction . . . 111
6.1.2 Our Approach . . . 113
6.1.3 Experiments . . . 115
6.1.4 Conclusion . . . 122
6.2 User Reviews Summarization . . . 123
6.2.1 Introduction . . . 123
6.2.2 The proposed Techniques . . . 124
6.2.3 Experiments and Results . . . 130
6.2.4 Conclusions . . . 131

In this chapter, we discuss the experiments that we carried out for two potential applications of lived experiences: opinion mining and summarization. In the first case, we show how lived experiences may be applied to opinion mining to predict the rating of reviews. We carried out some experiments with machine reading for summarization that are discussed in the second section of this chapter. Lived experiences were not used for this task, but we think that they may be successfully applied to user reviews summarization.

6.1 Correlating open rating systems and lived experiences extraction from text

6.1.1 Introduction

In chapter 4, we studied the correlation between event types which are extracted from user reviews and the rating scores given by users for these reviews. We tested whether it is possible to extract relevant event dictionaries from user reviews and then use these dictionaries to classify the polarity of reviews. The results show that some events have an influence on the rating scores given by users. We found that event features can discriminate the two types of reviews (Positive, Negating) with a F-measure of 88.3% (See table 4.8).

In chapter 5, we classified mentioned events in user reviews into two types: (1) Personal events, which contain the authors of the reviews among their participants, and (2) General events, which do not have the authors of the reviews as participants. This classification is

based on our definition of lived experience. We considered a lived user experience as an event mentioned in a review, where the author is among the participants. In addition, we viewed that personal events could be also classified into two types: (i) Lived Experience events, which represent authentic lived experiences, and (ii) Non-Lived Experience events, which have the author as a participant, but do not represent authentic lived experience.

The events, that we are used in Chapter 4 to study the correlation are all mentioned events in the text, Personal and General events. However, in this research, we aim to study the correlation between only live experience events (the (i) type, according to our classification in chapter 5) and ranking derived from open rating systems. The objective is to show if there is an influence of lived experience contents on user ranking. In other words, we aim to build an approach to test if lived experience contents can distinguish between two types of user reviews (Positive, and Negative).

As in Chapter 4, we formulate these tasks as a binary text classification task. We explored Web Semantic and natural language processing techniques to extract lived experience features from user reviews, and machine learning techniques to build a classifier so as to classify two types of reviews (Positives and Negatives) using these features.

As an example, let us consider the first hotel review in Chapter 5, Section 5.2:

- *The view from this hotel's rooms is quite stunning. And that's what make it very special, possibly better than the next door 4 star hotel and than many other hotels in Paris. The bedrooms interior decor is extremely nice. I asked for a room overlooking the pantheon and I got it. My deluxe room was number 32, and was tastefully decorated with a classic and beautiful Pierre Frey wallpaper, and an extra day bed. The bath had bathtub-shower combination and was separated from the toilet. If you book directly through the hotel, you'll get a voucher for a free-drink upon arrival. It was a bit cold at night at some point, maybe because it's March and the heating is not constantly on anymore. Each room has its own heating control, though. Strongly recommended.*

According to our definition of lived experience, this review contains three lived experiences: (1) *I asked for a room overlooking the pantheon and I got it.* (2) *My deluxe room was number 32, and was tastefully decorated with a classic and beautiful Pierre Frey wallpaper, and an extra day bed.* (3) *If you book directly through the hotel, you'll get a voucher for a free-drink upon arrival.* These lived experiences are represented by three personal event types *{Got, Ask, Decorate}*, respectively. All these event types have the author among their participants *{I, My, You}*.

In this work, we firstly extract these event types by performing a deep semantic parsing of text. Then, we verified whether they represent authentic lived experiences or not. Afterward, we use the events, which represent authentic lived experiences as features to build the classifier and study the correlation between these features and the rating given by the user, who is wrote this review.

6.1.2 Our Approach

Our approach for the polarity classification task using lived experience contents is similar to our approach in Chapter 4. However, in this work, we added two main steps, which consists of identifying personal events, which have the author among their participants, and then detecting the relevant events, which represent lived experience, and filtering out the irrelevant ones. Figure 6.1 show an architectural overview of this approach.

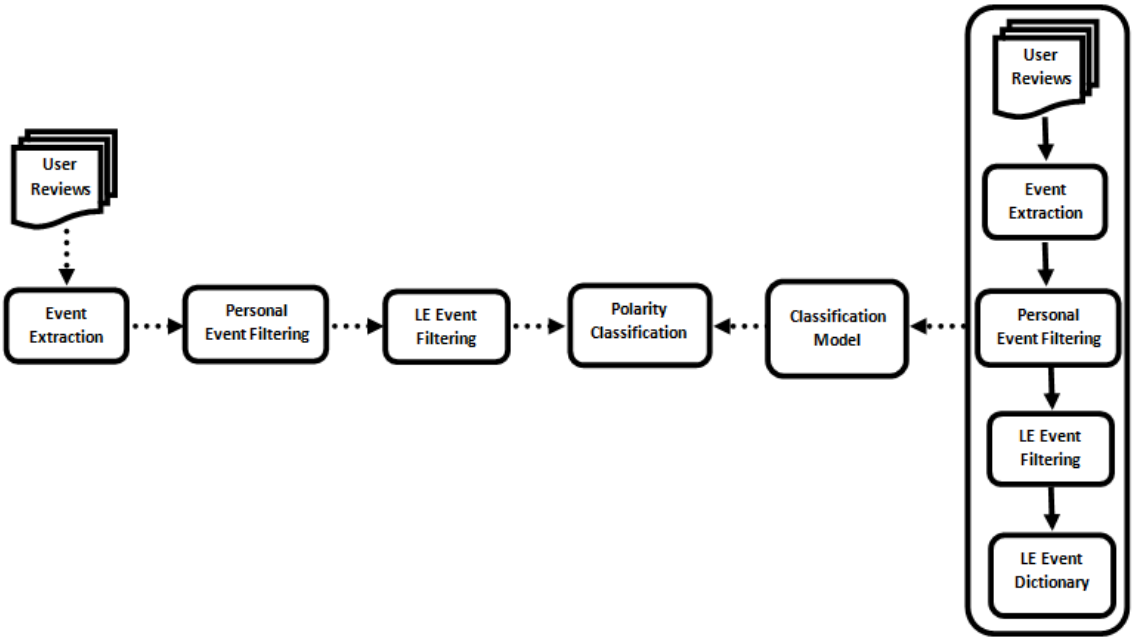


Figure 6.1: Overview of the proposed approach

Given a set of user reviews, we firstly extract all mentioned event types. Then, we filter out the general events, and keep the personal ones, which have the narrator among the participants. Afterward, we study personal events by taking into account the impact of their neighbors in the same reviews to finally select the relevant ones, which represent authentic lived experience. By the end, we use lived experience events to build an event dictionary D_E, and use the events in the dictionary as features to build machine learning classifiers to classify the polarity of the input reviews. Each of these steps will be discussed in the following sections.

6.1.2.1 Event Extraction

In order to extract events from user reviews, we followed our approach in Chapter 4, Section 4.2.2. We performed a deep semantic parsing of text which allow to obtain a RDF/OWL knowledge graph representation of the text. We employed a deep variety of machine reading [Etzioni et al. (2006)], as implemented in the FRED tool[1] [Gangemi et al. (2016), Presutti et al. (2012)].

[1] http://wit.istc.cnr.it/stlab-tools/fred

By applying the SPARQL query, which we have used in Chapter 5, Section 5.3.1, to the semantic graph produced by FRED, we extract all mentioned event types in the user reviews with their participants. For our example in Section 6.1.1, we are able to extract eight event types *{Get, Overlook, Recommend, Ask, Decorate, Have, Make, Separate}*. These events represent all mentioned events in the reviews, general and personal.

6.1.2.2 Personal Events Identification

The second step of our approach consists of detecting personal event types and filtering out the general ones. According to our definition of user lived experience, we considered events which have the author of the review among their participants as lived experiences events. In other word, lived experience events are the events, which have the first or second person pronoun (i.e. *I, You, We, Me, Us, My, Mine, our, ours, Your, Yours, ...*) as a participant (See Chapter 5, Section 5.3.2). For instance, from our example in Section 6.1.1, we are able to detect three lived experience event types *{Get, Ask, Decorate}*, and filter out the rest (i.e. *{Overlook, Recommend, Have, Make, Separate}*).

6.1.2.3 Event Filtering

We have shown in Chapter 5, Section 5.3.4, that some personal events could not represent lived experience even if they have the author of the review as a participant. Our objective in this step is to detect relevant events which really represent lived experience, and filter out the irrelevant ones. Therefore, we employed a filtering algorithm, which allows to study personal events by taking into account the impact of their neighbors in the same review to finally select the relevant ones, which represent lived experience events.

The algorithm 1 takes a personal event as input, and gives the label of it as output. We considered two labels: "LivedExperience" for relevant events, and "NonLivedExperience" for irrelevant ones. For this, we calculate the function $F = \alpha X(\mathbf{e}_j) + \beta Y(\mathbf{e}_j)$ for each personal event, and then classifies it according to the value of the function F. If F value is higher than 0, the target event represents a lived experience. The review provided as an example in Section 6.1.1 has three personal event types *{Get, Ask, Decorate}*. These events are labeled as lived experience events using this algorithm, and will be then used as features to construct the polarity classifiers.

6.1.2.4 Event Participants Extraction

In Chapter 5, we have shown that lived experiences involve events and their participants. These participants are the arguments of semantic roles (e.g. Agent, Patient, Oblique, Theme, etc.) associated with these events. We assumed that event participants could be useful for the polarity classification task and can upgrade our results. Therefore, we extracted and used them as features in order to classify the two types of user reviews.

In this work, we are interested in event participants for the first, second, and third degree in the semantic graph. Therefore, in order to extract them, we use the SPARQL query, which we used in Chapter 5 , Section 5.3.5. This query generates an event subgraph containing lived experience events, with their direct and indirect participants. Direct participants are the arguments which connect to an event directly. i.e. the direct object of the events. Indirect participants are the participants of direct event participants. i.e. the direct objects of the direct event participants.

This query also allows to extract event modifiers such as modality, logical negation, and adverbial qualities for the first, second, and third degree in the semantic graph. For example, From our example in Section 6.1.1, we are able to extract the following participants and modifiers:

$Ask(\text{“}Person\text{”}, \text{“}Room\text{”}[\text{“}A\text{”}])$.
$Get(\text{“}Person\text{”}, \text{“}Necessary\text{”}, \text{“}Arrival\text{”}, \text{“}Book\text{”}[\text{“}Direcly\text{”}],$
$\text{“}Voucher\text{”}[\text{“}A\text{”}, \text{“}Free - Drink\text{”}])$.
$Decorate(\text{“}Testefully\text{”}, \text{“}Bed\text{”}, \text{“}Room\text{”}[\text{“}Person\text{”}, \text{“}Multiple\text{”}, \text{“}Deluxe\text{”}, \text{“32”},$
$\text{“}Hotel\text{”}[\text{“}This\text{”}, \text{“4”}]], \text{“}Wallpaper\text{”}[\text{“}Beautiful\text{”}, \text{“}Classic\text{”}])$.

These lived experience events and participants will be then used as features to build a new classifier and test their abilities to classify the polarity of user reviews.

6.1.3 Experiments

6.1.3.1 Dataset

In order to experiment our approach, we used the dataset of [Ott et al. (2011)], that we have used in Chapter 4, Section 4.3.1, in order to classify the polarity of user reviews using all mentioned events in the text, personal and general. This dataset contains 600 user reviews collected from the 20 most popular Chicago hotels on TripAdvisor[2], 300 user reviews for the positive reviews and 300 user reviews for the negative ones. In our experimentation, 420 user reviews were used as a training set, 210 reviews for the positive class and 210 reviews for the negative one, and 180 user reviews were used to evaluate our classifier (90 for each class).

rev.class	Training Set	Test Set	Total
Positive	210	90	300
Negative	210	90	300
Total	420	180	600

Table 6.1: Characteristics of datasets used in our experiment

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

6.1.3.2 Review Classifier Construction

Using our training set, we were able to recognize and extract 2903 lived experience events: 1104 for the positive reviews and 1799 for the negative ones. The number of aggregated lived experience events is 759: 306 for the positive reviews and 453 for the negative ones (Table 6.2).

rev.class	LivedExp events	aggregated LivedExp events
Positive	1104	306
Negative	1799	453
Overall	2903	759

Table 6.2: The number of extracted lived experience events from the two classes of review.

For more details, the number of common events which appear in the two classes is 142. Therefore, the number of features that we used in our classifier is 617 $(759 - 142)$ lived experience event types: 164 are specific to the positive reviews, 311 are specific to negative ones, and 142 lived experience event types are detected in both classes.

As we have shown in Chapter 4, Section 4.3.2, that some common events are not useful in our classification task and could not be used to discriminate between the two types of reviews. Therefore, these event types should be removed in order to build a good classifier. We considered that an event is representative to a class if the probability $P(c|e) \geq \sigma$ where:
$c \in C = \{+,-\}$;
e: A generic event;
σ: A threshold that we determined empirically between 4 possibilities: $\{0.6, 0.7, 0.8, 0.9\}$. The best value of σ was 0.7 (See section 6.1.3.6).

For example, the personal event *Love* appeared 28 times as a lived experience event in the positive reviews and 7 times in the negative ones. It can be very useful to discriminate the positive reviews.

$$P(+|\text{“}Love\text{”}) = \frac{28}{28+7} \simeq 0.8 > 0.7.$$
$$P(-|\text{“}Love\text{”}) = \frac{7}{28+7} \simeq 0.2.$$

However, the event type *Ask* appeared 26 times as a lived experience event in the negative reviews and 9 times in the positive ones. This event type is helpful to discriminate the negative reviews.

$$P(+|\text{“}Ask\text{”}) = \frac{9}{9+26} \simeq 0.26.$$
$$P(-|\text{“}Ask\text{”}) = \frac{26}{9+26} \simeq 0.74 > 0.7.$$

While, the event *Stay* frequented 118 times as a lived experience event in the positive reviews and 125 times as a lived experience event in the negative ones. This

event appears almost identically in the two classes and should be deleted since it cannot be useful for our classification.

$$P(+|\text{“}Stay\text{”}) = \frac{118}{118+125} = 0.49 < 0.7.$$
$$P(-|\text{“}Stay\text{”}) = \frac{125}{118+125} = 0.51 < 0.7.$$

In the end, we removed 97 common events which do not help to discriminate the reviews types, and left 45 events among the events that exist in the two classes (23 events for the positive reviews and 92 for the negative ones): the result was a dictionary containing 520 characterizing events: 164 uniques for the positive reviews, 10 common events discriminating the positive reviews, 311 uniques for the negative reviews, and 35 common events but important to discriminate the negative reviews.

As we mentioned in Section 6.1.2.4, that event participants and modifiers could be useful for the polarity classification task. Therefore, we extract these arguments for the first, second and third degree and used them with their events to build an event-participant dictionary.

Using our training set, we were able to extract 2486 lived experience events and event participants: 668 for the positive reviews, 1153 for the negative ones, and 458 for both positive and negative reviews (i.e. common events and event participants which appear in the two classes of reviews). As we mentioned above, some common features could not improve the results of the polarity classification task. So, we should delete them. Using the probability $P(c|e) \geq 0.7$, we removed 191 common events and event participants and kept 237 ones to obtain at the end a new dictionary of event and event participants containing 2058 features: 668 uniques for the positive reviews, 76 common events and event participants discriminating the positive reviews, 1153 uniques for the negative reviews, and 161 common events and event participants which are important to discriminate the negative reviews.

6.1.3.3 Evaluation

As we mentioned above, we used our dictionaries as a collection of features to train a multinomial Naïve Bayes and Support Vector Machine classifiers as our classification models. Each review is transformed into a feature vector where the $i-$th component value is the frequency of the $i-$th event in the review, and then classified as either “Positive" or “Negative".

Our approach is evaluated in terms of precision (P), recall (R), and F-measure (F). We performed 10-fold cross-validation on the training set and yielded good results. The obtained results can be observed in Table 6.3 and Table 6.4, respectively.

To validate the obtained results, we used our dictionaries to classify the test set which contains 180 user reviews. The achieved results for this set can be also shown in Table 6.3 and Table 6.4, respectively.

		Training Set			Test Set		
Features	*Nb_Features*	P	R	F	P	R	F
All_LE_Eve	617	70.5%	70.5%	70.5%	65.0%	65.0%	65.0%
$LE\_Eve_{\sigma} = 0.7$	520	75.8%	75.5%	75.7%	65.7%	65.6%	65.7%
All_LE_Eve_Part	2486	77.9%	77.4%	77.7%	**76.8%**	**76.7%**	**76.7%**
$(LE\_Eve\text{-}Part)\sigma = 0.7$	2058	**84.1%**	**83.8%**	**84.0%**	73.4%	73.3%	73.4%

Table 6.3: Overall results for review classification using NB method with four configurations.

		Training Set			Test Set		
Features	*Nb_Features*	P	R	F	P	R	F
All_LE_Eve	617	70.3%	68.8%	69.5%	67.3%	67.2%	67.2%
$LE\_Eve_{\sigma} = 0.7$	520	73.7%	70.7%	72.2%	65.7%	65%	65.3%
All_LE_Eve_Part	2486	73.3%	71.4%	72.3%	**70.8%**	**70%**	**70.4%**
$(LE\_Eve\text{-}Part)\sigma = 0.7$	2058	**79.8%**	**76.4%**	**78.1%**	68.5%	66.7%	67.6%

Table 6.4: Overall results for review classification using SVM method with four configurations.

These tables show the results of our experiments using lived experience features with four configurations: *LE_All_Eve* configuration presents the results of using all extracted lived experience events, without deleting the common events, as features, $LE\_Eve_{\sigma} = 0.7$ presents the obtained results by deleting the lived experience events which cannot discriminate the user reviews (common events) and used the rest as features, *LE_All_Eve_Part* indicates the results using all extracted lived experience events and their participants as features, $(LE\_Eve\text{-}Part)\sigma = 0.7$ presents the obtained results when we delete some common lived experience events and event participants.

As shown in these tables, the results using lived experience events when $\sigma = 0.7$ have been particularity good. However, using all lived experience events without deleting the common ones decrease the performance by about 5% for the training set and 1% for the test set for NB method, and about 3% for the training set using SVM. Further adding participant features to all lived experience events give good results, but less effective than the results obtained using lived experience features (event and participants) with $\sigma = 0.7$, which achieves the best results for both training set and test set, and also for both classifiers NB and SVM. Using these features allows to obtain a slight increase, (8%) for both training and test sets for NB, and (4%) for the training set and (2%) for the test set for SVM, compared to the achieved results using event features.

In table 6.5 and table 6.6, we show the most frequent features in the positive and negative dictionaries, respectively, used to classify the polarity of reviews and gave the best results.Table 6.7 shows some of the common features that we are removed from our dictionary.

Lived Experience Features (f)	$C_+(f)$	$C_-(f)$	$P(+\|f)$
Great	59	20	0.75
Walk	41	9	0.82
Love	28	7	0.8
Enjoy	17	5	0.77
View	15	2	0.88
Really	15	5	0.75
Friendly	12	5	0.71
Excellent	10	2	0.83
DefinitelyStay	10	1	0.91
HighlyRecommend	7	1	0.88
DefinitelyRecommend	6	0	1
Shopping	6	1	0.86

Table 6.5: Most frequent features used for classifying positive reviews

Lived Experience Features (f)	$C_+(f)$	$C_-(f)$	$P(-\|f)$
Call	10	76	0.88
Pay	17	44	0.72
bed	10	28	0.74
Ask	9	26	0.74
Charge	2	23	0.92
Hear	6	20	0.77
Manager	1	19	0.95
End	5	15	0.75
Hard	5	15	0.75
Change	1	12	0.92
Complain	2	12	0.82
Cancel	3	11	0.79

Table 6.6: Most frequent features used for classifying negative reviews

Table 6.8 shows the results that we are obtained in studying the correlation between all event features, personal and general, with the ranking given by users for the same training set and test set using NB classifier. In this table, All_Eve configuration presents the results of using all extracted events, without deleting the common events, as features, $Eve_\sigma = 0.7$ presents the obtained results by deleting the events which cannot discriminate the user reviews (common events) and used the rest as features, All_Eve_Part indicates the results using all extracted events and their participants as features, $(Eve\text{-}Part)\sigma = 0.7$ presents the obtained results when we delete some common events and event participants. Using $(Eve\text{-}Part)\sigma = 0.7$ features, we achieved the best results (See Chapter 4, Section 4.3.3).

From Table (6.3) and Table (6.8), we observe that performance using all event features, general and personal, with their participants ($(Eve\text{-}Part)\sigma = 0.7$) is more efficient

Lived Experience Features (f)	$C_+(f)$	$C_-(f)$
Hotel	135	177
Stay	118	125
Go	41	75
Here	45	30
Again	35	23
Day	18	23
Feel	17	18
Bathroom	10	6
Breakfast	10	5
Come	10	14
Know	10	20
Make	15	30

Table 6.7: Some common features removed from the dictionary

		Training Set			Test Set		
Features	$Nb\_Features$	P	R	F	P	R	F
All_Eve	1547	79.8%	79.8%	79.8%	78.3%	77.8%	78%
$Eve_\sigma = 0.7$	1342	82.6%	82.6%	82.6%	75.5%	73.3%	74.4%
All_Eve_Part	3821	83.4%	83.1%	83.3%	**82.8%**	**82.8%**	**82.8%**
$(Eve\text{-}Part)\sigma = 0.7$	3197	**88.4%**	**88.1%**	**88.3%**	80.3%	79.4%	79.7%

Table 6.8: Overall results for review classification using all event features, personal and general, for NB method

than performance using only lived experience events, with their participants ($(LE\_Eve\text{-}Part)\sigma = 0.7$) for the polarity classification task. However, It is clear that lived experience events and their participants, which constitute (64%) of the total events and participants, achieve results very close to the results that have been obtained using all events and participants. In other word, lived experience events present the most efficient events in user reviews and could be used to classify the sentiments of user reviews with good results.

6.1.3.4 Error Analysis

As we have shown in Chapter 4, Section 4.3.4, the classification errors are due to discriminant events for a type reviews, which exist in reviews of the other type. For example, from a positive review, according to a user, we extracted the following events: {*Help*, *Put*, *Take*, *Experience*}. The classification results indicate that this review is negative. By looking in our dictionary, we only find the event {*Put*} among the other specified events. This event discriminates the negative reviews in our dictionary, but it exists in a positive review. For this reason, we are motivated to study and understand the impact of events. The other events of this review have been removed from the dictionary because they are common events and do not help to discriminate the two types of reviews.

In addition, many user reviews do not contain lived experience events. The user writes his opinion in general: {*The bath had bathtub-shower combination and was separated from the hotel*, *The room was small*, *Staff Helpful*, ...}. These reviews are difficult to be classified using lived experience event features. Therefore, we attempt to resolve this problem by adding some related features to our dictionaries.

6.1.3.5 Application to ESWC2014 challenge

As in Chapter 4 , Section 4.3.7, we compared our approach with the systems which participated in the Polarity Detection task, the elementary task in the ESWC-14 challenge on Concept Level Sentiment Analysis. The reviews which are used in this task were extracted from the Blitzer dataset[3]. To build our classifier for this task, we extract personal events with their participants from the training set which contain 8000 reviews (4000 positives and 4000 negatives). Then, we used them as features for a multinomial Naïve Bayes classifier.

Table 6.9 shows the results of our approach and the results of the top three participants in this challenge. The evaluation is carried out on the test set, which is composed of 2429 sentences constructed in the same way and from the same sources as the Blitzer dataset. Our system using lived experience features achieved the second best performance on Recall and the third best system in Precision and F-measure.

Participant	Precision	Recall	F-Measure	Final position
NCU	0.78	0.57	0.66	1
IBM	0.66	0.59	0.62	2
FBK	0.42	0.47	0.44	3
Event Approach	0.68	0.60	0.63	
LivedExp Event Approach	0.64	0.59	0.61	

Table 6.9: Results of Polarity Detection Task at ESWC2014

6.1.3.6 σ Estimation

To properly build and have a dictionary, allowing to discriminate the two types of reviews, we considered that a lived experience event is representative with respect to a class if the probability $P(c|e) \geq \sigma$. To determine the best value of σ, we tested 4 possibilities {0.6, 0.7, 0.8, 0.9}. Table 6.10, show the results of our experiment using NB method and *10 fold cross-validation* with the 4 values of σ.

This table indicates that when $\sigma = 0.7$, we obtain the best performance. These results justify our choice of the value of σ as 0.7 when we built our lived experience event dictionary in Section 6.1.3.2.

[3] http://www.cs.jhu.edu/\char'~mdredze/datasets/sentiment/

σ	**Precision**	**Recall**	**F-Measure**
0.6	73.6	73.5	73.5
0.7	**75.8**	**75.7**	**75.5**
0.8	72.5	72.9	72.3
0.9	71.9	71.8	71.9

Table 6.10: Results for the *10-fold cross* experiments for several values of σ for our training set

6.1.4 Conclusion

We presented an approach to study the correlation which could be found between lived experience events and the ranking given by users. We employed FRED to extract lived experience-based features from text. In order to detect lived experience events, we firstly identified personal events, which have the narrator among their participants. Then, we employed a filtering algorithm, which allow to filter out irrelevant personal events, which do no represent lived experience and keep relevant ones, which identify authentic lived experience. We were capable to build a lived experience feature dictionary, which can discriminate the two types of reviews. We compared the classification results using lived experience events with the result of using all mentioned events, general and personal. We found that lived experience events are very important arguments in user reviews and very useful for the polarity classification task.

The work presented in this section has resulted in the following publication:

- Ehab HASSAN, Davide BUSCALDI, and Aldo GANGEMI. Correlating open rating systems and lived experiences extraction from text. In the 12th International Conference on Semantic Systems, SEMANTiCS 2016, Leipzig, Germany, September 12-15, 2016.

6.2 User Reviews Summarization

6.2.1 Introduction

With the rapid expansion of e-commerce, the number of reviews for a specific product or service grows rapidly. Some popular products or services can get hundreds or even thousands of reviews at some large merchant sites (e.g. TripAdvisor[4], Booking[5], Amazon[6], etc.). In addition, many reviews are long and have a few sentences containing opinions on the product or the service. This makes the process of making a decision about a service or a product is hard. Therefore, summarizing user reviews could be helpful to resolve this problem.

Text summarization is a task consisting in the production of a concise description of a longer, more complex text [Radev et al. (2002)]. As we have shown in Chapter 2 (Section 2.3.4), summarization approaches can classified into two types: extractive and abstractive. In the first case, the original text is reduced to a smaller one, keeping the most important fragments. In the latter, a new text is produced on the basis of the context of the original one. Therefore, abstractive summarization needs a deeper comprehension of the underlying semantics, where extractive summarization can be considered as a shallower task, where the semantics does not play an important role.

In this research, we study the problem of generating feature-based summaries of user reviews in the touristic domain. This summarization is different from traditional text summarization because we only mine the features (or attributes) of the hotel on which the customers have expressed their opinions and whether the opinion are positive or negative. Therefore, we proposed an abstractive summarization method based on a machine reader and sentiment analysis dictionaries. Abstractive summarization is a useful task, especially in the summarization of customer reviews.

One of the most recent applications of the abstractive approaches is the summarization of product reviews and opinions [Ganesan et al. (2010)]. This is particularly useful in cases where there are many reviews and most of them are redundant: a user may have to read a great quantity of text before being able to obtain a precise idea of the qualities and the disadvantages of a product.

Machine readers have been introduced by [Etzioni et al. (2006)] as tools for text understanding. They combine different text analysis layers (Part-Of-Speech tagging, syntactic analysis, disambiguation, named entity recognition) to produce a rich semantic representation of the text, which is the reason why we chose to apply them to user reviews in the touristic domain for the abstractive summarization of opinions.

Our approach is close to the work of [Hu and Liu (2004)], where the author mined and summarized all the customer reviews of a product (See Chapter 2, Section 2.3.4).

[4] http://www.tripadvisor.com
[5] www.booking.com
[6] www.amazon.com

The authors generate feature-based extractive review summaries by (1) mining product features, that have been commented on by customers; (2) identifying opinion sentences, which contain one or more features, in each review and deciding their orientations (positive, or negative); (3) generating the summary, according to, firstly, the frequency of appearances of extracted features in the reviews, and then to the orientation of each extracted sentence.

However, in our work, given a set of customer reviews of an hotel, we produced an abstractive summary represented by the 4-tuple (Feature, Attribute, Frequency, Polarity). The feature tuple represents the aspect that customers have expressed their opinion on, attribute is the opinion words which could be associated with the features, frequency indicates the features frequency in an hotel reviews, and polarity represent the opinion orientation (positive, negative, neutral) of the extracted attributes.

6.2.2 The proposed Techniques

Figure 6.2 gives the architectural overview of our opinion summarization system. The input to the system is a set of user reviews. The output is the abstractive summary of these reviews.

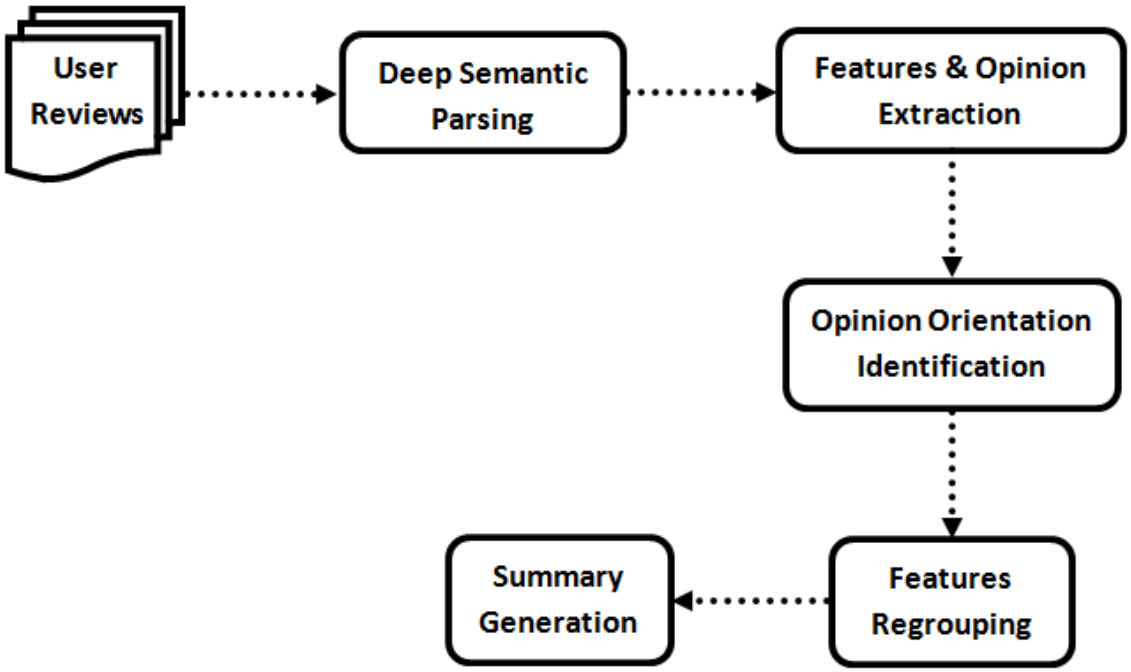


Figure 6.2: Overview of the proposed approach

Our system performs the summarization in four main steps: (1) Identify features that have been commented by customers; (2) Identify opinion words and their polarity, and deciding whether each opinion word is positive, negative, or neutral; (3) Regroup similar features together; (4) Summarize the results using the redundant opinions.

Given an input composed by a set of user reviews, the system first extract all features that appear explicitly as nouns or noun phrases in the reviews and have at least one opinion

word associated with them, together with their attributes and eventually the associated logical negation. Then, the polarity of each opinion word is identified using three sentiment lexicons (SentiWordNet ([Baccianella et al. (2010)]), AFINN[7] and [Liu (2012)]). In the last two steps, features are regrouped and a final summary is produced. Below, we discuss each of these steps.

6.2.2.1 Features and Opinion Extraction

The first step consists in the identification of features (or aspects) that are the object of evaluation by users. When users write a review of an hotel, for instance, they usually evaluate not the hotel in its entirety, but specific features of the hotel. Then, we need to find the attributes (or opinion words) used to express the opinions. We assumed that features are usually nouns or noun phrases, and such attributes are usually expressed as adjectives. For example, the following segment of a user review ***"The hotel is Great, the staff is helpful, and the room is nice"*** contains three explicit features {***"Hotel", "Staff", "Room"***} represented by nouns, and three explicit attributes {***"Great", "Helpful", "Nice"***}, which are associated with these features, respectively.

In order to extract the features with their associated attributes from user reviews, we perform a deep semantic parsing of text, obtaining a RDF Linked-Data-ready graph representation of the text. We employ a deep variety of machine reading systems, as implemented in the FRED tool[8] [Gangemi et al. (2016), Presutti et al. (2012)], which extracts knowledge (named entities, senses, taxonomies, relations, events) from text, resolves it onto the Web of Data, adds data from background knowledge, and represents all that in RDF and OWL (See Chapter 3, Section 3.2).

FRED is a tool to automatically transform knowledge extracted from text into RDF and OWL, i.e. it is a *machine reader* for the Semantic Web. It is available as a RESTful API and as a web application. In its current form, it relies upon several NLP components: Boxer[9] for the extraction of the basic logical form of text, BabelNet [Navigli and Ponzetto (2010)] for word sense disambiguation, and Apache Stanbol[10] for named entity resolution.

The figure 6.3 shows the output diagram for the mentioned sentence above. In this diagram, the following terms are detected, `hotel_1`, `staff_1`, and `room_1`, and classified as occurrences of the `Hotel`, `Staff`, and `Room` frames, respectively. Each term is associated with an adjective, which is represented as a quality modifier of the term using the DOLCE property `dul:hasQuality`.

As we have shown in Chapter 4, Section 4.3.6 and Chapter 5, Section 5.4.2.1, the semantic graphs produced by FRED contain additional triples that annotate the fragments from the text with their syntactic part-of-speech annotations. These annotations are ex-

[7] https://github.com/abromberg/sentiment_analysis/tree/master/AFINN
[8] http://wit.istc.cnr.it/stlab-tools/fred
[9] http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer
[10] http://stanbol.apache.org

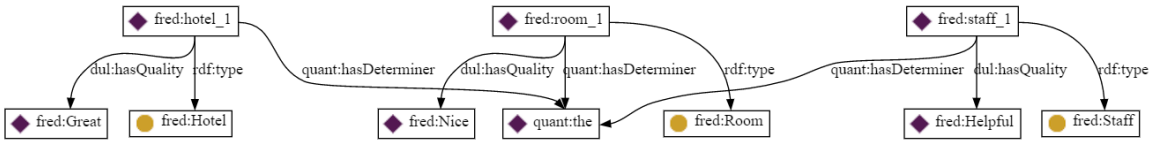


Figure 6.3: A FRED graph depicting the core subset of triples representing event-related knowledge.

pressed by means of the *pos.owl* ontology, which contains three part-of-speech tags (v, a ,n). In the following, we show the generated triples for our mentioned example above.

```
xmlns:fred="http://www.ontologydesignpatterns.org/ont/fred/domain.owl#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

fred:hotel_1
    dul:hasQuality fred:Great ;
    rdf:type fred:Hotel .

fred:Hotel
    pos:boxerpos pos.owl:n .

fred:staff_1
    dul::hasQuality fred:Helpful ;
    rdf:type fred:Staff .

fred:Staff
    pos:boxerpos pos.owl:n

fred:room_1
    rdf:type fred:Room ;
    dul:hasQuality fred:Nice .

fred:Room
      pos:boxerpos pos.owl:n
```

The triples indicate that the part-of-speech of each detected class is a noun (n). Each nous class is associated with an adjective, which is represented as a quality modifier. Since review features are usually nouns or noun phrases in user reviews, we only interested in features that appear explicitly as nouns or noun phrases in the reviews. Therefore, applying the following SPARQL query to the semantic graph produced by FRED, we can extract these features with their opinion words:

```
PREFIX dul: <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>
PREFIX vnrole: <http://www.ontologydesignpatterns.org/ont/vn/abox/role/>
PREFIX boxing: <http://www.ontologydesignpatterns.org/ont/boxer/boxing.owl#>
PREFIX boxer: <http://www.ontologydesignpatterns.org/ont/boxer/boxer.owl#>
PREFIX : <http://www.ontologydesignpatterns.org/ont/boxer/test.owl#>
PREFIX d0: <http://www.ontologydesignpatterns.org/ont/d0.owl#>
PREFIX schemaorg: <http://schema.org/>
PREFIX fred: <http://www.ontologydesignpatterns.org/ont/fred/domain.owl#>
PREFIX pos: <http://www.ontologydesignpatterns.org/ont/fred/pos.owl#>
PREFIX BE: <http://www.essepuntato.it/2008/12/earmark#>
SELECT  distinct ?Feature ?neg ?qlt
WHERE {
{
{?Feature rdf:type ?FeatureType}.
```

```
{{?FeatureType pos:boxerpos pos:n}
  UNION{?FeatureType rdfs:subClassOf* ?FeatureType_1 . ?FeatureType_1 pos:boxerpos pos:n}}.
  {{?FeatureType dul:hasQuality ?qlt}
         UNION{?Feature dul:hasQuality ?qlt}}
}
OPTIONAL {?sit boxing:involves ?Feature . ?sit boxing:involves ?qlt .
          ?sit boxing:hasTruthValue ?neg}
}
```

This SPARQL query allows to extract noun features (?*Feature*) with their adverbial qualities (opinion words) (?*qlt*). Therefore, from our example, we are able to extract three features `{Hotel, Staff, Room}`and three opinion words `{Great, Helpful, Nice}`. (?*neg*) represents the logical negations, which could be associated with a feature. Logical negations are very important to determine the polarity of features. Therefore, we extracted them from the FRED's graphs using the property `boxing:hasTruthValue` (See Chapter 3 , Section 3.3). If a segment of a user review contains a logical negation which modify the polarity of a feature, we extracted and added it to the associated attribute with this feature. For example, the segment ***"The room is not nice"*** describes a feature "Room", an attribute "Nice", and a negation "Not". Using our query, we extracted this feature and we considered that its associated attribute is "Not-Nice".

6.2.2.2 Opinion Orientation Identification

We now identify the polarity of the extracted opinion word. We had three types of polarity which can be assigned to opinion word (e.g. positive, negative, or neutral). In order to detect the opinion words polarities, we used three sentiment lexicons:

- SentiWordNet [Baccianella et al. (2010)] lexicon: an enhanced lexical resource explicitly devised for supporting sentiment classification and opinion mining applications. It is the result of the automatic annotation of all the synsets of WordNet [Miller et al. (1990)] according to the notions of "positivity", "negativity", and "neutrality". Each synset s is associated to three numerical scores *Obj(s), Pos(s)*, and *Neg(s)*, describing how objective, positive, and negative the terms contained in the synset are. Each of the three scores ranges from 0.0 to 1.0, and their sum is 1.0 for each synset. In our work of polarity detection of opinion words, we considered that the polarity is "Positive", if the score > 0.5, "Negative", when the score < 0.5 and "Neutral" otherwise (i.e. score $= 0.5$).
- AFINN[11] lexicon: is a list of English words rated for valence with an integer -5 (Negative) and $+5$ (Positive). The words have been manually labeled by Finn Arup Nilesen in $2009-2011$. This lexicon contains 1468 unique words and phrases. In our work, we considered the polarity of an opinion word is "Positive" if the score > 0, "Negative" if the score < 0, and "Neutral" if the score $= 0$.

[11] `https://github.com/abromberg/sentiment_analysis/tree/master/AFINN`

- Liu (2012) lexicon: a list of English positive and negative opinion words (around 6800 words). In our work, we considered an opinion word to be "Positive" if it has been found in the positive list, "Negative" if it has been found in the negative list, and "Neutral" if we do not find it in these two lists.

Using these lexicons, each extracted opinion word has three polarities *(P_1, P_2, P_3)*. Therefore, to determine the final polarity of an opinion word, we proposed four simple rules, which reduce the three polarities to a single value. Table 6.11: show the proposed rules:

P_1	P_2	P_3	**Result**
a	a	a	a
a	a	b	a
a	a	c	a
a	b	c	Neutral

Table 6.11: The proposed rules to obtain the final polarity of an opinion word.

In this table, a, b, and c are the three extracted polarities using our lexicons. Each of them could be Positive, Negative, or Neutral. The first, second, and third rules in this table indicate that the final polarity depends on the majority of the three extracted polarities. For example, the final polarity of (Positive, Positive, Negative) will be Positive. However, the last rule shows that if we have three different polarities for an opinion word (i.e. Positive, Negative, Neutral), then the final polarity will be Neutral. Besides, for the features which have a logical negation, the final polarity will be the opposite of the detected final polarity using the three lexicons. For our example ***"The room is not nice"***, the final polarity which is detected using the three lexicons is "Positive". However, the final polarity that we are mentioned for the "Room" feature is "Negative", since a logical negation is associated with it.

In this work, we are interested in only positive and negative orientations. Therefore, we deleted the features which have opinion words with neutral polarity, and kept the features that have attributes with positive and negative polarities. These features are then put into positive and negative categories according to the detected polarity. This is helpful for the next step of features regrouping, which allows to regroup features in each category separately.

6.2.2.3 Features Regrouping

In this step, we aim to regroup the remaining features, in each category, in order to generate the abstractive summary. We assumed that such features can have degrees of similarity between them. Therefore, measuring the similarity between features, which are in the same category, could be helpful to regroup them. For this, we used WordNet:Similarity package

[Pedersen et al. (2004)]. This package contains several semantic relatedness measures (e.g., Path Length [Patwardhan et al. (2003)], lch [Leacock and Chodorow (1998)], wup [Wu and Palmer (1994)], res [Resik (1995)], and Lin [Lin (1998)], jsn [Jiang and Conrath (1997)]).

In this research, we used the Lin similarity. In this measure, the similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are. This similarity could be calculated using the following function:

$$sim(A, B) = \frac{2 * logP(LCS(c_1, c_2))}{logP(c_1) + logP(c_2)}$$

This similarity ranges from 0.0 to 1.0. In our work, we considered that two features can be grouped together (i.e. consider as synonyms) if their similarity score is greater than 0.5. For example, the Lin similarity between the two features `(Room, Area)` is 0.98. Therefore, we have grouped them. In order to choose the final feature between the two regrouped features, we proposed to rank them according to the frequency of their appearances in the reviews, and chose the feature with the bigger ranking. In our example, the `Room` feature appeared 7 times in the positive category of 15 user reviews, and `Area` feature appeared 4 times in the same category. Therefore, we considered them as a `Room` feature.

We did not take into account this method to group together the attributes since the WordNet:Similarity package does not offer good semantic similarity measures for adjectives (the only one is the Lesk measure which is not as reliable as the Lin one).

6.2.2.4 Summary Generation

After all the previous steps, we are ready to generate the final feature-based reviews summary, which consists of the following steps:

- Each pair (Feature, Attribute), in each category (Positive, Negative), are ranked according to its frequency in the category.
- For each feature, in each category, we chose the pair (Feature, Attribute) with the highest ranking to be as a part of our abstractive summary.
- The highest ranking for a pair (Feature, Attribute) should be greater than 2.

By the end, the abstractive summary is represented using 4-tuple (Feature, Attribute, Frequency, Polarity). This representation enables potential users to see how the existing customers feel about the most frequent features (positive or negative). In addition, it could be helpful for user decision since they contain the most redundant features with their attributes (opinion word) and their polarities.

6.2.3 Experiments and Results

We used the dataset which was created by [Ott et al. (2011)] (Chapter 4, Section 4.3.1). We selected 300 truthful reviews. These reviews present the comments of users in the 20 most popular Chicago hotels. We have 15 reviews for each hotel. The summarization task carried out for each hotel separately. In other word, we summarize the reviews for each hotel separately.

For example, for the Affinia hotel, we analyzed the reviews with FRED, extracting 140 attributed features. In Figure 6.4 we show a subset (80) of the features retrieved from the 15 reviews and the associated attribute/opinion word.

Affinia,Perfect,	city,Windy,	hallway,Dark,	location,Nott,Better,
Affinia_chicago,Thing,	close,Great,	hallway,Loud,	location,Off,
apartment,Small,	Customer_service,Big,	hotel,Average,	location,Perfect,
area,Clean,	Customer_service,Incredible,	hotel,Big,	location,Perfect,
area,Nice,	Customer_service,Nott,Unusual,	hotel,Birthday-great,	location,Right,
area,Pleasant,	Customer_service,Thing,	hotel,Close,	lot,Topic,
area,Separate,	deal,Great,	hotel,Friendly,	low,New,
bathroom,Large,	desk,Front,	hotel,Great,	mile,Magnificant,
bathroom,Nice,	dollar,Internet,	hotel,Great,	morning,Last,
bathroom,Spacious,	door,Fantastic,	hotel,Great,	N_michigan_ave,Great,
bathroom,Spacious,	door,Next,	hotel,Nice,	N_michigan_ave,Thing,
bed,Comfortable,	end,American,	hotel,Nott,Excellent,	neuter,Magnificent,
bed,Comfortable,	end,High,	hotel,Thing,	noise,Little,
bed,Perfect,	end,Medium,	hotel,Too,	noise,Outside,
bed,Quiet,	end,Most,	hotel,Upscale,	Nyc,Better,
bedroom,Clean,	end,To,	hotel,Warm,	Nyc,Like,
birthday,40th,	equipment,New,	idea,Great,	Nyc,Our,
Chicago_affinia_dec,Fine,	example,Noisy,	location,Excellent,	Nyc,Still,
Chicago_affinia_dec,Impress	food,Good,	location,Excellent,	occasion,Several,
Chicago_affinia_dec,Thing,	gym,Nott,Open,	location,Great,	pantry,Stocked,

Figure 6.4: An excerpt (80 out of 140) of the attributed features extracted from the 15 reviews concerning Affinia hotel.

The next step was to find the polarity of each attribute. As we mentioned in Section 6.2.2.2, we used three sentiment analysis dictionaries to perform this task. Therefore, for each attribute, we obtain three polarities. We reduced the three polarities to a single value using the rules in Table 6.11. For instance, “comfortable" has a positive polarity in all three dictionaries, and “nervous" has negative polarity in all dictionaries. Therefore, the final polarity for “comfortable" is positive. However, the final polarity for “nervous" is negative. The features, which have attributes with a neutral polarities, have been filtered out. For instance, the features “Area", “Bathroom”, and “Noise” in Figure 6.4, which have the attributes “Separate”, “Large”, and “Little”, respectively, have been deleted. Then, the remaining features were regrouped according to the relatedness between them, and also according to their frequencies in the used 15 reviews . For example, the features {“Room”, “Bathroom”}, have been regrouped as one feature “Room”. The Lin similarity between

them is 0.75. The frequency of “Room” is 7. However, the “Bathroom” frequency is 4. Afterward, we summarized the reviews according to the polarity of each feature and then, according to the features frequencies. By the end, the 15 reviews were summarized to the attributed features in Table 6.12.

Feature	Attribute	Freq	Polarity	Feature	Attribute	Freq	Polarity
Staff	Helpful	2	+	Hotel	Great	3	+
Location	Perfect	2	+	Location	Excellent	2	+
Rate	Great	2	+	Rate	Bad	2	−
Room	Spacious	3	+	Room	Nice	2	+
Bed	Comfortable	2	+				

Table 6.12: The result of the summarization of the 15 reviews for Affinia hotel.

6.2.4 Conclusions

We proposed a set of techniques for summarizing user reviews based on Semantic Web and sentiment analysis methods to provide a feature-based abstractive summary of a large number of customer reviews. Therefore, we firstly performed a deep semantic parsing on the user reviews in order to transform them into RDF/OWL graphs. Then, we queried the generated graphs in order to extract features with their associated attributes. Afterward, the polarity for each extracted attribute has been detected using three sentiment analysis lexicons. Finally, the extracted features were regrouped and the final summary was generated according to the redundant features and their associated polarities.

Although this is a very preliminary work, we were able to reduce effectively the complete set of opinion to a synthetic table of features and attributes. Further directions may be to combine the attributes that are very similar (“perfect", “excellent"), using semantic similarity measures developed for Semeval STS[12], and find a way to deal with conflicting ratings. We need also to carry out a more comprehensive evaluation and compare to other summarization methods, such as the one proposed by [Popescu and Etzioni (2007)].

The work presented in this section has resulted in the following publication:

- Ehab Hassan, Davide Buscaldi, and Aldo Gangemi. Machine reading for abstractive summarization of customer reviews in the touristic domain. Septième Atelier Recherche d’Information SEmantique RISE, Rennes 30 juin 2015, page 6.

[12] http://alt.qcri.org/semeval2015/task2/

Chapter 7

Conclusion and Perspectives

Contributions

The first contribution of this thesis has been to adapt the machine reader tool FRED as an open event extraction tool. The ability of FRED to automatically generate RDF/OWL ontologies and linked data from natural language text has been exploited for event extraction. The information provided by FRED doesn't limit to the events themselves, but also their classes, modifiers, and participants.

The second contribution of this work is the demonstration that events are correlated to the rankings assigned by users to the reviews. To do this, we used the events identified by FRED as features to classify reviews using machine learning methods.

The third contribution is the definition itself of lived experiences: *A user lived experience is an event mentioned in a user review, where the author is among the participants.* This definition considers events, which have the narrators of the reviews among their participants, as the most important elements in lived experiences.

A related contribution is the method to recognize these lived experiences in user reviews. The method works in three main steps. Firstly, it identifies reviews which contain lived experiences contents. Afterward, it extracts lived experiences from identified reviews in the first step. Finally, it represents extracted lived experiences as event-based graphs. These events are extracted by performing a deep semantic parsing on user reviews, and then used in all our system steps. For the review identification step, we transformed this task into a binary text classification task. We used extracted events as features to train machine reading classifiers, which allow to identify the relevant reviews. For the extraction step, a filtering algorithm was employed to filter out irrelevant events, which do not represent lived experiences, and keep relevant ones. After that, event participants and modifiers for each relevant event are extracted and used to extract mentioned lived experiences. The last step of our system consists of using relevant events and their participants and modifiers to represent authentic lived experiences as event graphs.

Finally, we were able to use lived experiences, together with their participants and modifiers, as features in an opinion mining system, showing that, exactly like events, lived experiences are correlated to the rankings assigned by users to the reviews.

More contributions are constituted by the corpora that have been built from TripAdvisor, currently available at `http://www-lipn.univ-paris13.fr/~hassan/Lived_Experience_Extraction/` and a web application and RESTful API, which are pub-

licly accessible at https://lipn.univ-paris13.fr/ClientProj/client.jsp, and provide reusable RDF knowledge graphs that can be reused in other applications, e.g. for recommendation services.

Perspectives

The system presented in this thesis succeeds at identifying, extracting, and representing lived experiences from user reviews. Nevertheless, many aspects can be improved or extended and some issue are still open for lived experience extraction and representation.

Gold Standard for Lived Experience Reviews

In future, we intend to create a large-scale comprehensive corpus of lived experience reviews in order to provide a *gold-standard*. The application of our lived experience classifier, which we have used in order to identify reviews containing lived experiences, to a large corpus of user reviews, allows to identify a large dataset of lived experience reviews. A sample of this dataset will be manually annotated in order to provide the gold standard. According to our knowledge, no method has yet attempted to extract lived experiences from user reviews. Therefore, creating a gold standard and a corpus of texts containing lived experiences will enable this research and motivate researchers in this area to discover and ameliorate this kind of knowledge extraction from text.

Experiments in Large Dataset and other domains

Our system of lived experience recognition, presented in Chapter 5, has been tested on a small corpus consisting of 383 user reviews in the touristic domain. In future work, we aim to test our system in a large dataset. In addition, we will increase the scale of experiments to include other review domains (e.g. Products, Restaurants, Books, etc).

Sentiment Analysis of Lived Experiences

Since user preferences change among users, user sentiment on extracted lived experiences may be different, depending on their desires and requirements. Representing lived experiences with their associated sentiments could be helpful in user decision. Our lived experience extraction system, presented in Chapter 5, extracts authentic lived experiences without detecting sentiments which could be associated with them. Therefore, we aim to extend our system to apply a fine-grained sentiment analysis to the extracted lived experiences. We can follow our approach, presented in Chapter 6, Section 6.2, to perform this task. We firstly identify lived experience features and opinion words. Afterward, sentiment lexicons and opinion ontologies will be used to detect the polarity of extracted lived experiences. Finally, any detected sentiment will be associated with the aspects of lived experience, which could be then matched to user preferences.

Enriching Lived Experience Graphs

We represented extracted lived experience as event sub-graphs containing lived experience events with their participants and modifiers. As merchant websites offer reviews with additional information about users (e.g. profile name, city, number of posted reviews, etc.), it could be useful to add this information to our lived experience graphs. Therefore, we aim to link the generated event sub-graphs to contextual knowledge (user ranking, user profiles, etc), which can enhance these sub-graphs. This information with the detected sentiment of each sub-graph allow to create a bank of knowledge graphs representing event-oriented lived experiences.

Lived Experiences Summarization

Extracting lived experience from reviews helps users making a good and faster decision. However, the number of extracted lived experiences from a large dataset of user reviews could be in hundreds. Eventually, the process of making a decision about a service or product requires some time. Therefore, summarizing user reviews based on lived experience could be used to avoid this problem. We have shown in Chapter 2 that summarization approaches can be classified into two types: extractive and abstractive. In future work, we aim to generate both extractive and abstractive summaries for user reviews based on extracted lived experiences. For the extractive summary, we will use machine learning techniques e.g. clustering to regroup similar lived experience. Then, for each cluster, we retrieve the most representative sentence to be in our final summary. For the abstractive summary, we can perform two approaches. The first one is similar to our approach, represented in Chapter 6, Section 6.2, where lived experience features will be extracted and regrouped based on their frequencies and polarities. The second approach will be based on regrouping the different event-based lived experience graphs that share experiencers, places, time, sentiments, or other features. Semantic similarities will be used to firstly measure the similarity between lived experience events. Then, for similar events, the similarity between their participants will be measured.

Other potential applications of Lived Experience Extraction

Spam Detection

Opinion spam detection is a new research direction in the sentiment analysis area. The objective of this research is to detect such spamming activities to ensure that the opinions on the Web are a trusted source of valuable information. In future work, we aim at using lived experience contents to detect opinion spam in user reviews. We can formulate this task as a binary text classification task with two classes *spam* and *non-spam*. We will explored the use of machine learning technique to build the reviews classifiers. Extracted lived experience from each class will be used as features to train the classifiers.

Product or Service Profile

Many merchant websites propose several types of products or services. For example, TripAdvisor[1] proposes five types of travel, *{Families, Couples, Solo, Business, Friends}*, for each of its services (e.g. Hotel, Restaurant, Holiday Rentals, etc). In addition, Tripadvisor classifies posted user reviews according to these types. Therefore, extracting lived experience from reported reviews for each type, and summarizing them could be an application to determine the most appropriate travel type for each service.

[1] https://www.tripadvisor.com/

Bibliography

Muhammad Abdul-Mageed, Mona T Diab, and Mohammed Korayem. Subjectivity and sentiment analysis of modern standard arabic. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 587–591. Association for Computational Linguistics, 2011.

Eneko Agirre and Aitor Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Association for Computational Linguistics, 2009.

David Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics, 2006.

James Allan. Topic detection and tracking: event-based information organization. 2002.

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. In *Proceedings of the Broadcast News Transcription and Understanding Workshop (Sponsored by DARPA)*, 1998a.

James Allan, Ron Papka, and Victor Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45. ACM, 1998b.

Mohamed A Aly and Amir F Atiya. Labr: A large scale arabic book reviews dataset. In *ACL (2)*, pages 494–498, 2013.

Renzo Angles and Claudio Gutierrez. Querying rdf data from a graph database perspective. In *European Semantic Web Conference*, pages 346–360. Springer, 2005.

Chinatsu Aone and Mila Ramos-Santacruz. REES: a large-scale relation and event extraction system. In *Proceedings of the sixth conference on Applied natural language processing*, pages 76–83, 2000.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204, 2010.

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

Michele Banko, Oren Etzioni, and Turing Center. The tradeoffs between open and traditional relation extraction. In *ACL*, volume 8, pages 28–36, 2008.

Dave Beckett and Brian McBride. Rdf/xml syntax specification (revised). *W3C recommendation*, 10, 2004.

David Beckett, Tim Berners-Lee, and Eric Prud'hommeaux. Turtle-terse rdf triple language. *W3C Team Submission*, 14:7, 2008.

Farah Benamara, Baptiste Chardon, Yvette Yannick Mathieu, Vladimir Popescu, et al. Towards context-based subjectivity analysis. In *IJCNLP*, pages 1180–1188, 2011.

Adrian Benton, Lyle Ungar, Shawndra Hill, Sean Hennessy, Jun Mao, Annie Chung, Charles E Leonard, and John H Holmes. Identifying potential adverse effects using the web: A new approach to medical hypothesis generation. *Journal of biomedical informatics*, 44(6):989–996, 2011.

Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

Clive Best, Erik van der Goot, Ken Blackler, Teófilo Garcia, and David Horby. Europe media monitor. Technical report, Technical Report EUR 22173 EN, European Commission, 2005.

Steven Bethard and James H Martin. Identification of event mentions and their semantic class. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 146–154, 2006.

Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

Jiang Bian, Umit Topaloglu, and Fan Yu. Towards large-scale twitter mining for drug-related adverse events. In *Proceedings of the 2012 international workshop on Smart health and wellbeing*, pages 25–32. ACM, 2012.

Maria Björk, Thomas Wiebe, and Inger Hallström. Striving to survive: Families lived experiences when a child is diagnosed with cancer. *Journal of Pediatric Oncology Nursing*, 22(5):265–275, 2005.

Johan Bos. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286. Association for Computational Linguistics, 2008.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Jeen Broekstra and Arjohn Kampman. An rdf query and transformation language. In *Semantic Web and Peer-to-Peer*, pages 23–39. Springer, 2006.

Kevin Burton, Akshay Java, and Ian Soboroff. The icwsm 2009 spinn3r dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media (ICWSM 2009), San Jose, CA*, 2009.

Erik Cambria, Robert Speer, Catherine Havasi, and Amir Hussain. Senticnet: A publicly available semantic resource for opinion mining. In *AAAI fall symposium: commonsense knowledge*, volume 10, 2010.

Gavin Carothers and Andy Seaborne. Rdf 1.1 n-triples: A line-based syntax for an rdf graph. *World Wide Web Consortium*, 24, 2014.

Roberto Casati and Achille C. Varzi. Events. In *E.N. Zalta, editor*. The Standford Encyclopedia of Philosophy. http://philpapers.org/rec/CASE-2, 2014.

Nathanael Chambers and Dan Jurafsky. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics, 2011.

Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 216–223. Association for Computational Linguistics, 2003.

Nancy A Chinchor. Overview of muc-7/met-2. 1998.

Roderick Chisholm. Events and propositions. *Noûs*, pages 15–24, 1970.

Sandra Chung and Alan Timberlake. Tense, aspect, and mood. *Language typology and syntactic description*, 3:202–258, 1985.

Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602. Association for Computational Linguistics, 2006.

Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Saric, and Isabel Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *IJCAI*, pages 659–664, 2005.

Rudi L Cilibrasi and Paul MB Vitanyi. The google similarity distance. *IEEE Transactions on knowledge and data engineering*, 19(3):370–383, 2007.

Jacob Cohen. A coefficient of agreement for nominal scales. 1960.

Patricia J Connell. *A phenomenological study of the lived experiences of adult caregiving daughters and their elderly mothers*. PhD thesis, University of Florida, 2003.

Bonaventura Coppola, Aldo Gangemi, Alfio Gliozzo, Davide Picca, and Valentina Presutti. Frame detection over the semantic web. pages 126–142, 2009.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Walter Daelemans, Jakub Zavrel, Kurt van der Sloot, and Antal Van den Bosch. Timbl: Tilburg memory-based learner. *Tilburg University*, 2004.

Yan Dang, Yulei Zhang, and Hsinchun Chen. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. *IEEE Intelligent Systems*, 25 (4):46–53, 2010.

Mathieu d'Aquin, Enrico Motta, Marta Sabou, Sofia Angeletou, Laurian Gridinoc, Vanessa Lopez, and Davide Guidi. Toward a new generation of semantic web applications. *IEEE Intelligent Systems*, 23(3):20–28, 2008.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 948–956. Association for Computational Linguistics, 2010.

Hal Daumé III. Notes on cg and lm-bfgs optimization of logistic regression. *Paper available at http://www.isi.edu/ hdaume/docs/daume04cg-bfgs.ps*, 198:282, 2004.

Donald Davidson. Actions et événements. 1993.

Ian Davis, Thomas Steiner, and AL Hors. Rdf 1.1 json alternate serialization (rdf/json). *W3C Working Group Note. W3C*, 7, 2013.

David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 348–355. Association for Computational Linguistics, 1997.

Pierre A Devijver and Josef Kittler. *Pattern recognition: A statistical approach*. Prentice hall, 1982.

Barbara Di Eugenio, Nick Green, and Rajen Subba. Detecting life events in feeds from twitter. In *ICSC*, pages 274–277, 2013.

Thomas Dickinson, Miriam Fernandez, Lisa A Thomas, Paul Mulholland, Pam Briggs, and Harith Alani. Identifying prominent life events on twitter. In *Proceedings of the 8th International Conference on Knowledge Capture*, page 4. ACM, 2015.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Proceedings of the fourth International Conference on Language Resources and Evaluation (LREC'04)*, volume 2, page 1, 2004.

John Domingue, Dieter Fensel, and James A Hendler. *Handbook of semantic web technologies*. Springer Science & Business Media, 2011.

Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

Andrew Michael Duffy. *Fellow Travellers: What do users trust on recommender websites? A case study of TripAdvisor. com.* PhD thesis, 2012.

Laura Cox Dzurec. Fatigue and relatedness experiences of inordinately tired women. *Journal of Nursing Scholarship*, 32(4):339–345, 2000.

Michelle Lynn Edmonds. The lived experience of nursing students who study abroad: A qualitative inquiry. *Journal of Studies in International Education*, 2010.

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

Oren Etzioni, Michele Banko, and Michael Cafarella. Machine reading. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

Roy Fielding. Fielding dissertation: Chapter 5: Representational state transfer (rest). *Recuperado el*, 8, 2000.

Charles Fillmore. Frame semantics. *Linguistics in the morning calm*, pages 111–137, 1982.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proceedings of the 13th international conference on World Wide Web*, pages 482–490. ACM, 2004.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348, 2010.

Aldo Gangemi. A comparison of knowledge extraction tools for the semantic web. In *Proceedings of ESWC2013*, 2013.

Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on ontologies*, pages 221–243. Springer, 2009.

Aldo Gangemi, Valentina Presutti, Diego Reforgiato Recupero, Andrea Giovanni Nuzzolese, Francesco Draicchio, and Misael Mongiovi. Semantic Web Machine Reading with FRED. *Sermantic Web*, 2016. URL `http://www.semantic-web-journal.net/system/files/swj1297.pdf`.

Andrew Gordon and Reid Swanson. Identifying personal stories in millions of weblog entries. In *Third International Conference on Weblogs and Social Media, Data Challenge Workshop, San Jose, CA*, 2009.

Andrew S Gordon. Story management technologies for organizational learning. In *International Conference on Knowledge Management. In Special Track on Intelligent Assistance for Self-Directed and Organizational Learning, Graz, Austria*, 2008.

Karl Gottschalk, Stephen Graham, Heather Kreger, and James Snell. Introduction to web services architecture. *IBM systems Journal*, 41(2):170, 2002.

Ralph Grishman, David Westbrook, and Adam Meyers. Nyu's english ace 2005 system description. *ACE*, 5, 2005.

Ramanathan Guha. Open rating systems. In *URL:< citeseer. ist. psu. edu/694373. html*, 2003.

Peter Haase, Jeen Broekstra, Andreas Eberhart, and Raphael Volz. A comparison of rdf query languages. In *International Semantic Web Conference*, pages 502–517. Springer, 2004.

Sigrídur Halldórsdóttir and Elisabeth Hamrin. Experiencing existential changes: the lived experience of having cancer. *Cancer nursing*, 19(1):29–36, 1996.

Alaa Hamouda and Mohamed Rohaim. Reviews classification using sentiwordnet lexicon. In *World Congress on Computer Science and Information Technology*, 2011.

Hilda Hardy, Vika Kanchakouskaya, and Tomek Strzalkowski. Automatic event classification using surface text features. In *Proc. AAAI06 Workshop on Event Extraction and Synthesis*, pages 36–41, 2006.

Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. Sparql 1.1 query language. *W3C Recommendation*, 21, 2013.

Blossom Hart and Victoria M Grace. Fatigue in chronic fatigue syndrome: a discourse analysis of women's experiential narratives. *Health Care for Women International*, 21 (3):187–201, 2000.

Andreas Harth, Maciej Janik, and Steffen Staab. Semantic web architecture. In *Handbook of Semantic Web Technologies*, pages 43–75. Springer, 2011.

Silvana Hartmann, György Szarvas, and Iryna Gurevych. Mining multiword terms from wikipedia. *Semi-Automatic Ontology Development: Processes and Resources*, pages 226–258, 2012.

Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating nlp using linked data. In *International Semantic Web Conference*, pages 98–113. Springer, 2013.

Thorsten Hennig-Thurau, Kevin P. Gwinner, Gianfranco Walsh, and Dwayne D. Gremler. Electronic word-of-mouth via consumer-opinion platforms: What motivates consumers to articulate themselves on the internet? *Journal of Interactive Marketing*, 18(1):38 – 52, 2004. ISSN 1094-9968. doi: http://dx.doi.org/10.1002/dir.10073. URL `http://www.sciencedirect.com/science/article/pii/S1094996804700961`.

Masahiko Higashiyama, Kentaro Inui, and Yuji Matsumoto. Acquiring noun polarity knowledge using selectional preferences. In *Proceedings of the 14th Annual Meeting of the Association for Natural Language Processing*, pages 584–587, 2008.

James Higginbotham, Fabio Pianesi, and Achille C Varzi. *Speaking of events*. Oxford University Press, 2000.

Jerry R Hobbs, Ellen Riloff, N Indurkhya, and FJ Damerau. Information extraction, in handbook of natural language processing, 2010.

Frederik Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska De Jong. An overview of event extraction from text. In *Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011) at Tenth International Semantic Web Conference (ISWC 2011)*, volume 779, pages 48–57. Citeseer, 2011.

Ian Horrocks, Bijan Parsia, Peter Patel-Schneider, and James Hendler. Semantic web architecture: Stack or two towers? In *International Workshop on Principles and Practice of Semantic Web Reasoning*, pages 37–41. Springer, 2005.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

Sheng-Hao Hung, Chia-Hung Lin, and Jen-Shin Hong. Web mining for event-based commonsense knowledge using lexico-syntactic pattern matching and semantic role labeling. *Expert Systems with Applications*, 37(1):341–347, 2010.

Kevin Hutt. A comparison of rdf query languages. In *Proc. of 21th Computer Science Seminar, Hartfort, Connecticut*, pages 1–7, 2005.

Kentaro Inui, Shuya Abe, Kazuo Hara, Hiraku Morita, Chitose Sao, Megumi Eguchi, Asuka Sumida, Koji Murakami, and Suguru Matsuyoshi. Experience mining: Building a large-scale database of personal experiences and opinions from web documents. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 314–321. IEEE Computer Society, 2008.

Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

Hongyan Jing and Kathleen R McKeown. Cut and paste based text summarization. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 178–185, 2000.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

Paula N Kagan. *Feeling listened to: A lived experience of human becoming*. 2004.

Hans Kamp. A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222, 1981.

Victoria E Karian, Stacey M Jankowski, and Judy A Beal. Exploring the lived-experience of childhood cancer survivors. *Journal of pediatric oncology nursing*, 15(3):153–162, 1998.

Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. Rql: a declarative query language for rdf. In *Proceedings of the 11th international conference on World Wide Web*, pages 592–603. ACM, 2002.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics, 2009.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. Opinion mining from web documents: Extraction and structurization. *Information and Media Technologies*, 2(1): 326–337, 2007.

Alice Krieg-Planque. À propos des «noms propres d'événement».. événementialité et discursivité. *Les Carnets du Cediscor. Publication du Centre de recherches sur la didacticité des discours ordinaires*, (11):77–90, 2009.

Lun-Wei Ku, Yu-Ting Liang, Hsin-Hsi Chen, et al. Opinion extraction, summarization and tracking in news and blog corpora. In *AAAI spring symposium: Computational approaches to analyzing weblogs*, volume 100107, 2006.

Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

Mildred L Larson. *Meaning-based translation: A guide to cross-language equivalence*. University press of America Lanham, 1984.

Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

Robert Leaman, Laura Wojtulewicz, Ryan Sullivan, Annie Skariah, Jian Yang, and Graciela Gonzalez. Towards internet-age pharmacovigilance: extracting adverse drug reactions from user posts to health-related social networks. In *Proceedings of the 2010 workshop on biomedical natural language processing*, pages 117–125. Association for Computational Linguistics, 2010.

Shasha Liao and Ralph Grishman. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics, 2010.

Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304. Citeseer, 1998.

Bing Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666, 2010.

Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

Xiao Liu and Hsinchun Chen. Identifying adverse drug events from patient social media: A case study for diabetes. *IEEE Intelligent Systems*, 30(3):44–51, 2015.

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

Robert E Longacre. *The grammar of discourse*. Springer Science & Business Media, 2013.

Juha Makkonen. Investigations on event evolution in tdt. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 student research workshop-Volume 3*, pages 43–48. Association for Computational Linguistics, 2003.

Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Uniform resource identifier (uri): Generic syntax. 2005.

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. Sentiment classification using word sub-sequences and dependency sub-trees. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 301–311. Springer, 2005.

David McClosky, Mihai Surdeanu, and Christopher D Manning. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1626–1635. Association for Computational Linguistics, 2011.

George Miller and Christiane Fellbaum. Wordnet: An electronic lexical database, 1998.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

Richard Montague. On the nature of certain philosophical entities. *The Monist*, 53(2): 159–194, 1969.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224, 2008.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

Roberto Navigli and Simone Paolo Ponzetto. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics, 2010.

David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319, 2011.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106, 2005.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

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

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

Rosemarie Rizzo Parse. Nursing science major paradigms, theories, and critiques. 1987.

Rosemarie Rizzo Parse. The human becoming school of thought a perspective for nurses and other health professionals. 1998.

Rosemarie Rizzo Parse. *Qualitative inquiry: The path of sciencing*. Number 14. Jones & Bartlett Learning, 2001.

Rosemarie Rizzo Parse. The lived experience of feeling very tired: a study using the parse research method. *Nursing Science Quarterly*, 16(4):319–325, 2003.

Jan Pascal. Phenomenology as a research method for social work contexts: Understanding the lived experience of cancer survival. *Currents*, 9(2), 2010.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257. Springer, 2003.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

JW Pennebaker, CK Chung, M Ireland, A Gonzales, and RJ Booth. The development and psychometric properties of liwc2007: Liwc. net, 2007.

Silvio Peroni, Aldo Gangemi, and Fabio Vitali. Dealing with markup semantics. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 111–118. ACM, 2011.

Thomas Ploeger, Maxine Kruijt, Lora Aroyo, Frank de Bakker, Iina Hellsten, Antske Fokkens, Jesper Hoeksema, and Serge ter Braake. Extracting activist events from news articles using existing nlp tools and services. In *Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web*, page 30, 2013.

Thierry Poibeau. Extraction automatique d'information(du texte brut au web s'emantique). 2003.

Simone Paolo Ponzetto and Michael Strube. Taxonomy induction based on a collaboratively built knowledge repository. *Artificial Intelligence*, 175(9-10):1737–1756, 2011.

Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In Anne Kao and StephenR. Poteet, editors, *Natural Language Processing and Text Mining*, pages 9–28. Springer London, 2007. ISBN 978-1-84628-175-4. doi: 10.1007/978-1-84628-754-1_2. URL http://dx.doi.org/10.1007/978-1-84628-754-1_2.

Valentina Presutti, Francesco Draicchio, and Aldo Gangemi. Knowledge extraction based on discourse representation theory and linguistic frames. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 114–129. Springer, 2012.

Eric Prud'Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3:28–34, 2003a.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40, 2003b.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. Iso-timeml: An international standard for semantic annotation. In *Proceedings of the seventh Conference on Language Resources and Evaluation (LREC)*, 2010.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408, 2002.

Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. *arXiv preprint cmp-lg/9505040*, 1995.

Adwait Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution*. PhD thesis, University of Pennsylvania, 1998.

Diego Reforgiato Recupero and Erik Cambria. Eswc'14 challenge on concept-level sentiment analysis. In *Semantic Web Evaluation Challenge*, pages 3–20. Springer, 2014.

Jason D Rennie, Lawrence Shih, Jaime Teevan, David R Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

P Resik. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

Alan Ritter, Oren Etzioni, Sam Clark, et al. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112, 2012.

Barbara Rosario and Marti A Hearst. Multi-way relation classification: application to protein-protein interactions. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 732–739. Association for Computational Linguistics, 2005.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. Evita: a robust event recognizer for qa systems. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 700–707, 2005.

Roser Saurii, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. Timeml annotation guidelines, 2005.

Karin Kipper Schuler. Verbnet: A broad-coverage, comprehensive verb lexicon. 2005.

Andy Seaborne. Rdql-a query language for rdf, w3c member submission 9 january 2004. *World Wide Web Consortium*, 2004.

Yohei Seki, Koji Eguchi, Noriko Kando, and Masaki Aono. Opinion-focused summarization and its analysis at duc 2006. In *Proceedings of the Document Understanding Conference (DUC)*, pages 122–130, 2006.

Michael Sintek and Stefan Decker. Triple-an rdf query, inference, and transformation language. In *INAP*, pages 47–56, 2001.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263, 2008.

Herbert Spielgelberg. Doing phenomenology. *Martinus, The Hague, Netherlands*, 1975.

Mark Steedman. *The syntactic process*, volume 24. MIT Press, 2000.

Veselin Stoyanov and Claire Cardie. Partially supervised coreference resolution for opinion summarization through structured rule learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 336–344. Association for Computational Linguistics, 2006.

Carlo Strapparava, Alessandro Valitutti, et al. Wordnet affect: an affective extension of wordnet. In *LREC*, volume 4, pages 1083–1086, 2004.

Alexa K Stuifbergen and Sharon Rogers. The experience of fatigue and strategies of self-care among persons with multiple sclerosis. *Applied Nursing Research*, 10(1):2–10, 1997.

Charles Sutton. Grmm: A graphical models toolkit, 2006.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 133–140. Association for Computational Linguistics, 2005.

Yap-Peng Tan, Drew D Saur, Sanjeev R Kulkami, and Peter J Ramadge. Rapid estimation of camera motion from compressed video with application to video annotation. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):133–146, 2000.

Hristo Tanev, Jakub Piskorski, and Martin Atkinson. Real-time news event extraction for global monitoring systems. 5039:207–218, 2008.

Meredith Temple-Smith, Sandra Gifford, and Mark StoovÚ. The lived experience of men and women with hepatitis c: implications for support needs and health information. *Australian Health Review*, 27(2):46–56, 2004.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

Carla Treloar and Tim Rhodes. The lived experience of hepatitis c and its treatment among injecting drug users: qualitative synthesis. volume 19, pages 1321–1334, 2009.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484, 2005.

Willem Robert van Hage, Véronique Malaisé, Marieke Van Erp, and Guus Schreiber. Linked open piracy. In *Proceedings of the sixth international conference on Knowledge capture*, pages 167–168. ACM, 2011.

Seth Van Hooland, Max De Wilde, Ruben Verborgh, Thomas Steiner, and Rik Van de Walle. Exploring entity recognition and disambiguation for cultural heritage collections. *Digital Scholarship in the Humanities*, 30(2):262–279, 2015.

Max Van Manen. " doing" phenomenological research and writing: An introduction. 1984.

Max Van Manen. *Researching lived experience: Human science for an action sensitive pedagogy*. Left Coast Press, 2015.

Guido Van Oorschot, Marieke Van Erp, and Chris Dijkshoorn. Automatic extraction of soccer game events from twitter. In *Proceedings of the Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web*, page 15, 2012.

Vapnik N Vladimir and V Vapnik. The nature of statistical learning theory, 1995.

Chris Welty and Lora Aroyo. Harnessing disagreement for event semantics. In *Proc. of DERIVE Wks at ISWC2012*, 2012.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.

Janyce M Wiebe, Rebecca F Bruce, and Thomas P O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 246–253. Association for Computational Linguistics, 1999.

Michael J Witbrock and Vibhu O Mittal. Ultra-summarization (poster abstract): a statistical approach to generating highly condensed non-extractive summaries. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 315–316, 1999.

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

Changsheng Xu, Jinjun Wang, Kongwah Wan, Yiqun Li, and Lingyu Duan. Live sports event detection based on broadcast video and web-casting text. In *Proceedings of the 14th ACM international conference on multimedia*, pages 221–230. ACM, 2006.

Min Xu, Ling-Yu Duan, Jianfei Cai, Liang-Tien Chia, Changsheng Xu, and Qi Tian. Hmm-based audio keyword generation. In *Pacific-Rim Conference on Multimedia*, pages 566–574. Springer, 2004.

YIMING YANG, JG CARBONELL, RD BROWN, T PIERCE, BT ARCHIBALD, and XIN LIU. Learning approaches for detecting and tracking news events. *IEEE intelligent systems & their applications*, 14(4):32–43, 1999.

Jeffrey M Zacks and Barbara Tversky. Event structure in perception and conception. *Psychological bulletin*, 127(1):3, 2001.

Yifan Zhang, Changsheng Xu, Yong Rui, Jinqiao Wang, and Hanqing Lu. Semantic event extraction from basketball games using multi-modal analysis. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 2190–2193, 2007.