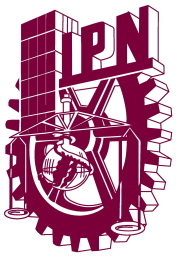


INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Laboratorio de Procesamiento de Lenguaje Natural

Polarity summarization with opinion mining

TESIS

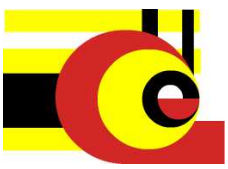
QUE PARA OBTENER EL GRADO DE:

MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

Ing. Iván Omar Cruz García

Directores de tesis:
Dr. Alexander Gelbukh
Dr. Grigori Sidorov



Mexico, D.F. Diciembre 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 11:00 horas del día 30 del mes de octubre de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

"Polarity summarization with opinion mining"

Presentada por el alumno:

CRUZ	GARCÍA	IVÁN OMAR
Apellido paterno	Apellido materno	Nombre(s)

Con registro: B 1 2 1 0 3 0

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA
Directores de Tesis

Dr. Alexander Gelbukh

Dr. Grigori Sidorov

Dr. Sergio Suárez Guerra

Dr. Francisco Hiram Calvo Castro

Dra. Olga Kolesnikova

Dr. Miguel Jesús Torres Ruiz

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 19 del mes noviembre del año 2014, el (la) que suscribe Iván Omar Cruz García alumno (a) del Programa de Maestría en Ciencias de la Computación con número de registro B121030, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Alexander Gelbukh y Grigori Sidorov y cede los derechos del trabajo intitulado Polarity Summarization with opinion mining, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección ivancruz.bht@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Nombre y firma

Resumen

Este trabajo de tesis presenta un nuevo modelo para resúmenes de opinión. Esta tarea consiste en la extracción de partes las principales opiniones de un texto y construir un resumen de opinión legible y entendible para los humanos. Esta tarea es muy difícil y está constituida por diferentes métodos de minería de opinión, procesamiento del lenguaje natural, aprendizaje de máquina, recuperación de información y otras áreas.

El objetivo principal de este trabajo es expandir el estado del arte en la minería de opinión, investigando en nuevas métodos de resúmenes de opinión y las tareas involucradas. Otro objetivo de este trabajo es proveer a la comunidad científica nuevas herramientas para afrontar los retos en esta área. Al momento de escribir esta tesis existen muy pocas, de existir alguna, herramientas públicamente disponibles para resúmenes de opinión.

Este trabajo explica a detalle los modelos que describen los fenómenos encontrados al investigar esta tarea, los diferentes métodos usados y/o desarrollados para resúmenes de opinión, su implementación y los resultados obtenidos.

Una herramienta de resúmenes de opinión recibe como entrada un texto con opiniones. La salida de este método podría ser las oraciones o frases de dicho documento que mejor resumen las opiniones encontradas en el texto. Otra posibilidad sería una especie de resumen estructurado que describe las propiedades cuantitativas de las opiniones encontradas en el documento. El trabajo propuesto usa ambos enfoques.

El método propuesto para resúmenes de opinión puede ser descrito en 3 pasos.

- Primero es necesario descomponer la entrada en oraciones y determinar cual de estas frases son opiniones, obteniendo su polaridad en el proceso.
- El siguiente paso es determinar que tópicos o aspectos tiene cada oración.
- Finalmente, el resumen es creado usando las oraciones extraídas. Puede ser un texto corto o un resumen estructurado.

Cada paso así como los resultados obtenidos en los experimentos son descritos a detalle. Toda la investigación realizada se centró solamente en texto en inglés.

Abstract

This thesis work presents a novel framework for Opinion Summarization. This task consist in the extraction of parts of a opinionated text that represents the main opinions and build a legible and human-readable opinion summary. This task is very difficult and it involves several methods related to Opinion Mining, Natural Language Processing, Machine Learning, Information Retrieval and other areas.

The main goal of this work is to expand the state of the art in Opinion Mining by researching in new ways for Opinion Summarization and the tasks it involves. Another goal of this work is to provide to the scientific community new tools to tackle challenges in this area. At the moment of writing this thesis there are few, if any, public available opinion summarization tools.

The present work explains in detail the models that describes the phenomena found investigating this task, the different methods used and/or developed for Opinion Summarization, their implementation and the results obtained.

An opinion summarization tool receives an opinionated text document. The output could be the sentences or phrases of the given document that summarize better the opinions found in the text. Another possibility would be some sort of structured summary describing the quantitative properties of the opinions found in the document. This work uses both approaches.

The proposed method for opinion summarization can be described in 3 steps:

- First it is necessary to decompose the input document into sentences and determine which of these sentences are opinionated, obtaining their polarity in the process.

- Next step is determine which topics or aspects each opinionated sentence have.
- Finally the summary is made using the extracted sentences. It could be a short text or a structured summary.

Each step and the results obtained during the performed experiments are described in detail. The whole research work focused only on the English language.

Acknowledgments

This thesis work would not have been possible without the support of many people. First and foremost, I would like to thank my two advisors Alexander Gelbukh and Grigori Sidorov for their support and their knowledge. To all the researchers and colleagues that I met and I work with during my research.

I also want to thank my parents and my family for their support, encouragement and love; to my friends for their support and advise through this journey. I would also thank to the CIC and the IPN for all the support and for the opportunity to study my MSc there; To CONACYT for the financial support

Contents

List of figures **2**

List of tables **3**

1. Introduction **4**

1.1. Overview 4
1.2. Objetives 4
1.3. Opinion Mining and Opinion Summarization 5
1.4. Contributions 6
1.5. Problem definition 7
1.5.1. Opinion Definition 7
1.6. Opinion Mining Analysis Levels 9
1.6.1. Document Level 10
1.6.2. Sentence Level 10
1.6.3. Aspect-based Level 11
1.7. Sentiment Classification 12
1.7.1. Supervised Sentiment Classification 12
1.7.2. Lexicon-based Sentiment Classification 12
1.8. Aspect Extraction 13
1.8.1. Explicit Aspect Extraction 14
1.8.2. Implicit Aspect Extraction 14
1.9. Aspect-based Opinion Summarization 17

2. State of the Art **20**

2.1. Overview 20

2.2. Sentiment Classification 20
2.2.1. Supervised Approaches 20
2.2.2. Lexicon-based Approaches 21
2.3. Aspect Extraction 21
2.3.1. Explicit Aspect Extraction 22
2.3.2. Implicit Aspect Extraction 26
2.4. Aspect-based Opinion Summarization 27

3. Research Methodology **30**
3.1. Overview 30
3.2. Polarity Classification 30
3.2.1. Description 30
3.2.2. Algorithm 31
3.2.3. Implementation 34
3.3. Explicit Aspect Extraction 34
3.3.1. Description 34
3.3.2. Supervised Learning 35
3.3.3. Sequence Labeling 35
3.3.4. Conditional Random Fields 36
3.3.5. Explicit Aspect Extraction Approach 37
3.4. Implicit Aspect Extraction 45
3.5. IAI Extraction 46
3.5.1. IAI Corpus 46
3.5.2. Feature Crafting 49
3.5.3. Implementation 49
3.5.4. Baselines 51
3.6. Mapping IAI to Implicit Aspects 52
3.6.1. Word Representation 54
3.6.2. Neural Network Language Models 55
3.6.3. Learning Word Representations with NNLM 57
3.6.4. IAI to Implicit Aspects Mapping Method 62
3.6.5. Baselines 65

3.7. Opinion Summarization 66
3.7.1. Structured Opinion Summary 66
3.7.2. Textual Opinion Summary 67
3.7.3. Set Covering problem as an ILP problem 69
3.7.4. Sentences Costs 70
3.7.5. Textual Opinion Summary Generation with ILP 72
3.7.6. Textual Opinion Summary Baselines 75

4. Results **81**
4.1. Overview 81
4.2. Polarity Classification 81
4.3. Explicit Aspect Extraction 82
4.4. Implicit Aspect Indicators Extraction 83
4.5. Implicit Aspect Extraction 85
4.6. Opinion Summarization 87
4.6.1. Structured Opinion Summarization 87
4.6.2. Textual Opinion Summarization. 88

Conclusions **96**

A. Probabilistic Graphical Models **98**
A.1. Overview 98
A.2. Introduction 98
A.3. Conditional Random Fields Definition 101
A.4. Features Functions 102
A.5. Linear Chain Conditional Random Fields 103
A.6. Parameter Estimation in Linear Chain CRF 105

B. Neural Networks **107**
B.1. Overview 107
B.2. Supervised classification with Neural Networks 107
B.3. Forward Propagation 110
B.4. Backpropagation 110
B.5. Softmax Regression in Neural Networks 112

C. Linear Programming and Integer Linear Programming **115**
C.1. Overview 115
C.2. Linear Programming 115
C.3. The Simplex Algorithm 118
C.3.1. Standard Augmented Form 119
C.3.2. Simplex Tableaux 119
C.3.3. Pivot operations 120
C.3.4. Algorithm 121
C.4. Integer Linear Programming 122
C.4.1. Solving ILP problems 123

Bibliography **125**

Nomenclature **131**

List of Figures

1.1. Aspect Extraction Example 14
1.2. An aspect-based opinion summary 18

2.1. Visualization of aspect-based summary of opinions on a digital camera 27
2.2. Visual opinion comparison of two digital cameras 27

3.1. Sequence Labeling in POS Tag 37
3.2. Example of a sentence in the reviews dataset. 39
3.3. Disjunction feature example. 40
3.4. A sentence in a CRFClassifier training data file. 42
3.5. CRFClassifier console training command 43
3.6. CRFClassifier properties file 44
3.7. CRFClassifier testing command 45
3.8. CRFClassifier properties file for IAI extraction 50
3.9. A 2-dimensional feature space 57
3.10. A lookup table 58
3.11. Collobert et al. NNLM 59
3.12. Collobert et al. NNLM with one-hot vectors as input and a projection layer. 61
3.13. The Skip-gram model. 62
3.14. Word2Vec command 65
3.15. The bar chart displayed by the opinion summarization tool 68
3.17. ILP model MathProg code 73
3.18. ILP instance MathProg code 74
3.19. GLPSOL command 74
3.16. The graphical representation of the textual summarization approach. 80

4.1. Precision, Recall and F1 Score with Biased CRF 86
4.2. Apex DVD Player structured summary 91
4.3. Canon Camera structured summary 92
4.4. Nikon Camera structured summary 93
4.5. Nokia Cellphone structured summary 94
4.6. Nomad MP3 player structured summary 95

A.1. Probabilistic Graphical representation of a Naive Bayes Classification scheme 100
A.2. Feature Functions Example 103
A.3. Linear Chain CRF 104
A.4. Examples of feature functions for POS Tagging with $k = 2$ 105

B.1. A single neuron 108
B.2. Sigmoid and tanh functions plots 108
B.3. A small neural network 109

C.1. An LP feasible region 117

List of Tables

3.1. Dataset Statistical Properties. 38
3.2. Features Description 41
3.3. Corpus Properties. 47
3.4. Statistical Properties. 48
3.5. Corpus POS Distribution 48

4.1. Polarity classification performance 81
4.2. Explicit aspect extraction performance 82
4.3. Explicit aspect extraction performance comparation. 83
4.4. IAI Extraction performance with different features 84
4.5. Precision, Recall and F1 Score with different IAI Class bias 85
4.6. Precision performance 86
4.7. Recall performance 87
4.8. F1-Score performance 87
4.9. Global Performance 87
4.10. Aspect coverage percentage comparative analysis 89
4.11. Compression rate comparative analysis 89
4.12. Summary entropy comparative analysis 90

1. Introduction

1.1. Overview

This chapter gives and introductory view of the topics involved in Opinion Summarization. Also it describes the scope of this work, its objectives and its justification.

1.2. Objetives

General Objectives

- Design and create software that implements a polarity summarization method. It will be based on opinion mining.
- Design and create new methods for Aspect Extraction and Opinion Summarization.

Particular Objetives

- Implement a Opinion Polarity Classifier.
- Implement an Explicit Aspect Extractor.
- Implement an Implicit Aspect Extractor.
- Implement an Opinion Summarization Tool.

1.3. Opinion Mining and Opinion Summarization

Opinions are central to almost all human activities and are key influencers of our behaviors. Our beliefs and perceptions of reality, and the choices we make are, to a considerable degree, conditioned upon how others see and evaluate the world. For this reason, when we need to make a decision we often seek out the opinions of others. This is not only true for individuals but also true for organizations. Opinions and its related concepts such as sentiments, evaluations, attitudes, and emotions are the subjects of study of sentiment analysis and opinion mining. The inception and rapid growth of the field coincide with those of the social media on the Web, e.g., reviews, forum discussions, blogs, microblogs, Twitter and social networks. We have a huge volume of opinionated data recorded in digital forms for the first time in human history. Since early 2000, sentiment analysis has grown to be one of the most active research areas in natural language processing. It is also widely studied in data mining, Web mining, and text mining. In fact, it has spread from computer science to management sciences and social sciences due to its importance to business and society as a whole. In recent years, industrial activities surrounding sentiment analysis have also thrived. Numerous startups have emerged. Many large corporations have built their own in-house capabilities. Sentiment analysis systems have found their applications in almost every business and social domain.

Unlike factual information, opinions are essentially subjective. One opinion from a single source is usually not sufficient for action. In most applications, one needs to analyze opinions from a large number of people. This indicates that some form of summary of opinions is desired. Although an opinion summary can be in one of many forms, e.g., structured summary or short text summary, the key components of a summary should include opinions about different entities and their aspects and should also have a quantitative perspective. The quantitative perspective is especially important because 20% of the people being positive about a product is very different from 80% of the people being positive about the product

In general, opinion summarization can be seen as a form of multi-document text summarization. Text summarization has been studied extensively in NLP. How-

ever, an opinion summary is quite different from a traditional single document or multi-document summary (of factual information) as an opinion summary is often centered on entities and aspects and sentiments about them, and also has a quantitative side. Traditional single document summarization produces a short text from a long text by extracting some "important" sentences. Traditional multi-document summarization finds differences among documents and discards repeated information. Neither of them explicitly captures different topics/entities and their aspects discussed in the document, nor do they have a quantitative side. The "importance" of a sentence in traditional text summarization is often defined operationally based on the summarization algorithms and measures used in each system. Opinion summarization, on the other hand, can be conceptually defined. The summaries are thus structured. Even for output summaries that are short text documents, there are still some explicit structures in them. The approach proposed in this work includes both structured opinion summary and short text summary.

1.4. Contributions

This thesis work introduces a novel method for extracting implicit aspect indicators (these indicators are explained in sec. 1.8.2.1) using Conditional Random Fields. It is shown that this method significantly outperforms existing approaches. As a part of this effort, this work presents a corpus for implicit aspect indicators extraction and for implicit aspect extraction. To the best of our knowledge, this is the first corpus of its kind.

This work also proposes a novel method for implicit aspect extraction using semantic relatedness computed with distributed representations of words. This work shows that this method outperforms the current approaches based on ontologies.

Finally, this work introduces a novel approach for opinion summarization that tries to maximize the informativeness and the aspect coverage of such summary. This method is compared with 3 baselines, verifying its effectiveness

1.5. Problem definition

Opinion Mining mainly studies opinions which express or imply positive or negative sentiments. This section thus defines the problem in this context.

1.5.1. Opinion Definition

In order to illustrate better the problem, a look to an opinion is useful. The next text is a camera review extracted from a opinion mining corpus and it is an opinionated document.

> "(1) I bought a Canon G12 camera six months ago. (2) I simply love it. (3) The picture quality is amazing. (4) The battery life is also long. (5) However, my wife thinks it is too heavy for her."

The whole phrase was divided by sentences. There are five of them and each one express an opinion.

1.5.1.1. Opinion Polarity

The opinion polarity is the value expressing if an opinion is positive or negative. This approach is also known as binary opinion polarity classification and it is the simplest way to quantize an opinion. Another way to express the polarity is using the neutral class, i.e. a document without opinion. Other approach is to classify how positive or negative an opinion is, usually using a discrete metric, e.g. a five-stars movie rating. Unless stated otherwise, it will be assumed a binary opinion polarity. A person can observe in the review that the sentence (1) express no opinion whatsoever. The phrases (2), (3) and (4) express a positive opinion, although sentence (4) is implicitly positive since a long battery life is consider a good thing. Finally the sentence 5 express again an implicit opinion. In this case is negative since a camera being "too heavy" is considered a drawback. The notation for the opinion polarity used in this work is s, and it expresses the sentiment of an opinionated instance.

1.5.1.2. Entity

Back to our example, it is possible to observe that the sentence (2) expresses a positive opinion about "it". If someone looks just to that sentence there is no way to know what "it" is referring to. But if the whole review is given, it is possible to infer that the sentence (2) is giving a positive opinion of the "camera" *entity.* The entity is the target of an opinion that has been expressed. It is also know as the *opinion target.* The notation for expressing the entity used during this work is e.

1.5.1.3. Opinion Holder

Another thing to notice is that opinions come from a source. It could be a friend or a blog on the internet. In the given example, the source of the sentences (1-4) is the review's author. In the sentence (5) the source is the author's wife. This sources are called *opinion sources* or *opinion holders.* The notation for opinion holder is h.

1.5.1.4. Time

The *time* of when an opinion was expressed is also important. Usually later opinions have more importance than the previous ones. The notation for time is t.

1.5.1.5. Aspects

Finally, is it possible to notice that the entity for the whole review is the "Canon camera". However, some of the sentences describe different *aspects* of the entity. For example the sentence (3) is talking about the picture quality of the camera. The sentence (4) describes the battery life. The sentence (5) implicitly describes the "weight" of the camera.

Many approaches for opinion mining don't consider aspects as part of the analysis framework. This work does include aspects as part of the analysis and it will be discussed in depth in the next section. The notation for aspect is a.

1.5.1.6. Opinion quintuple

Considering the previous analysis, it is possible to come up with a more formal definition of what an opinion is. Therefore, an opinion is a quintuple,

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l) \tag{1.1}$$

where e_i is the entity of the opinion, a_{ij} is an aspect j of the entity e_i, s_{ijkl} is the opinion polarity of the aspect a_j of the entity e_i, h_k is the opinion holder and t_l is the time when the opinion is expressed by h_k.

The research presented in this work uses this opinion definition. The main task is to extract every opinion expressed by this quintuple and use them to generate the opinion summary. However our approach only focuses on the sentiment, the aspects and the entities of opinions. The opinion holder and the time the opinions were expressed are considered irrelevant.

1.6. Opinion Mining Analysis Levels

There are different levels of analysis granularity in opinion mining. They can be classified in 3 main levels:

- Document Level
- Sentence Level
- Aspect-based Level

In the following each level is described more deeply.

1.6.1. Document Level

Document level focuses on the opinion polarity classification of an entire document. This level assumes each element of the expression (1.1) is the same for every sentence. Also this level is useful when every element in (1.1), beside the sentiment, is considered irrelevant. A large majority of research papers on this topic classifies online reviews. Document sentiment classification is not easily applicable to non-reviews such as forum discussions, blogs, and news articles, because such postings tend to evaluate multiple entities and compare them. In many cases, it is hard to determine whether a posting actually evaluates the entities that the user is interested in, and whether the posting expresses any opinion at all, let alone to determine the sentiment about them. Document-level sentiment classification does not perform such fine-grained tasks, which require in-depth natural language processing. In fact, online reviews do not need sentiment classification because almost all reviews already have user-assigned star ratings. In practice, it is the forum discussions and blogs that need sentiment classification to determine people's opinions about different entities (e.g., products and services) and topics.

1.6.2. Sentence Level

Sentence level, also called subjective analysis, tries to determine the sentiment for each sentence in a document. This level make the same assumptions for every sentence that document level makes. Sentence sentiment classification can be solved either as a three-class classification problem or as two separate classification problems. In the latter case, the first problem is to classify whether a sentence expresses an opinion or not. The second problem then classifies those opinion sentences into positive and negative classes. The first problem is usually called subjectivity classification, which determines whether a sentence expresses a piece of subjective information or factual (objective) information. Objective sentences are regarded as expressing no sentiment or opinion. This can be problematic because objective sentences can also imply opinions. For example the second sentence in the phrase "My phone worked pretty well the last 6 months. Then, it stopped

working yesterday" is an objective sentence, but it implies a negative sentiment about the phone due to an undesirable fact.

1.6.3. Aspect-based Level

Classifying opinion texts at the document level or the sentence level is often insufficient for applications because they do not identify opinion targets or assign sentiments to such targets. Even if we assume that each document evaluates a single entity, a positive opinion document about the entity does not mean that the author has positive opinions about all aspects of the entity. Likewise, a negative opinion document does not mean that the author is negative about everything. For more complete analysis, we need to discover the aspects and determine whether the sentiment is positive or negative on each aspect.

To extract such details we go to the aspect level, which was also called the feature-based opinion mining. At the aspect level, the objective is to discover every quintuple (1.1) in a given document d. To achieve this goal, two main tasks have to be performed.

1. Aspect Extraction: This task extracts aspects that have to be evaluated. For example, in the sentence, "The voice quality of this phone is amazing," the aspect is "voice quality", and it is an aspect of a larger entity represented by "this phone.". In the sentence "I love this phone" the entity is again the phone but this time the entity is the one which is being evaluated and there are no aspects.

2. Aspect Sentiment Classification: This task determines whether the opinions on different aspects are positive, negative, or neutral.

In task such as opinion summarization, a higher analysis granularity level is needed. This is because this task tries to extract the information that summarizes better the input document. Therefore the aspect-based level is the best choice for opinion summarization and this is the approach used in the research described in this work.

1.7. Sentiment Classification

This is the task of determining the orientation of the sentiment expressed in a sentence. For Aspect-based Opinion Mining, this task also considers the sentiment for each aspect in an opinionated document. There are three main approaches: supervised learning, lexicon-based and unsupervised learning.

1.7.1. Supervised Sentiment Classification

This approach commonly uses supervised machine learning techniques, e.g. it requires labeled data to train a classifier that models the inherent properties of the used training set. The basic approach in Sentiment Classification tasks is to label the sentiment of whole sentences based on the aspects within them. However, the key issue is how to determine the scope of each sentiment expression, i.e., whether it covers the aspect of interest in the sentence.

Supervised learning is dependent on the training data. A model or classifier trained from labeled data in one domain often performs poorly in another domain. Although domain adaptation (or transfer learning) has been studied by researchers, the technology is still far from mature, and the current methods are also mainly used for document level sentiment classification as documents are long and contain more features for classification than individual sentences or clauses. Thus, supervised learning has difficulty to scale up to a large number of application domains.

1.7.2. Lexicon-based Sentiment Classification

Lexicon-based approaches can avoid some of the issues found in supervised techniques, and has been shown to perform quite well in a large number of domains. Such methods are typically unsupervised. They use a sentiment lexicon (which contains a list of sentiment words, phrases, and idioms), composite expressions, rules of opinions, and (possibly) the sentence parse tree to determine the sentiment

orientation on each aspect in a sentence. They also consider sentiment shifters, but-clauses and many other constructs which may affect sentiments.

Sentiment shifters are words that change the sentiment of other words or whole sentences. The negation is the simplest sentiment shifter. For example the sentence "This phone is nice" has a positive sentiment since the word nice have a positive cognotation. On the other hand the sentence "This phone is not nice" has a negative sentient because the sentiment shifter "not" changes the polarity of the whole sentence given by the word "nice".

The lexicon-based approach also has its own shortcomings. They will be discussed later. This approach is the one used for sentiment classification.

1.8. Aspect Extraction

Recalling from the opinion quintuple shown in the equation 1.1, every opinion has an an entity which is the opinion target. Moreover the opinion might be about specific aspects of such entity. These aspects a_{ij} are also considered in equation 1.1. For simplicity, "aspects" is used to reefer the entities and aspects together, which are the opinion targets. Thus the aspect extraction task consists in the extraction of every aspect a_{ij} from every opinion quintuple given as input.

Aspects can be divided in 2 main classes: explicit aspects and implicit aspects.

Explicit aspects are words in a opinionated document that explicitly denote the opinion target. In the sentence "The screen of my new phone is superb", the word "screen" is the opinion target, and the whole sentence gives an opinion about the larger entity "phone".

On the other hand, an implicit aspect is a concept that represents the opinion target of an opinionated document but is not specified explicitly in the text. One can infer that the sentence "This camera is sleek and very affordable" implicitly gives a positive opinion on the aspects *appearance* and *price* of the entity camera.

These aspects would be explicit in an equivalent sentence “The appearance of this camera is sleek and its price is very affordable."

The present work considers explicit aspects and implicit aspects.

1.8.1. Explicit Aspect Extraction

The basic idea in explicit aspect extraction is to mark words from an opinionated input text as aspects. The Fig. 1.1 shows an example of this. There is an opinionated sentence as input. The output is a set of duples. Each duple consists of a token of the sentence and the label assigned by an aspect extractor tool. The label 'A' is for “Aspect” and the label 'O' is for “Other”. One can observe that the words “screen” and “camera” are labeled as aspects, as they should be.

INPUT :“$The\ screen\ of\ my\ phone\ is\ superb,\ but\ its\ camera\ is\ really\ awful.$”

$$\begin{aligned}\textbf{OUTPUT}:\{&('The', \mathrm{O}), ('screen', \mathrm{A}), ('of', \mathrm{O}),\\ &('my', \mathrm{O}), ('phone', \mathrm{O}), ('is', \mathrm{O}),\\ &('superb', \mathrm{O}), (',', \mathrm{O}), ('but', \mathrm{O}),\\ &('its', \mathrm{O}), ('camera', \mathrm{A}), ('is', \mathrm{O}),\\ &('really', 'O'), ('awful', \mathrm{O}), ('.', \mathrm{O})\}\end{aligned}$$

Figure 1.1.: Aspect Extraction Example

The explicit aspect extraction approach used in this research work is described in detail in sec. 3.3.

1.8.2. Implicit Aspect Extraction

Explicit aspect extraction modeling is straightforward and has been widely researched. There are several approaches for this task. Still, limited work has been

done in extracting implicit aspects. This task is very difficult yet very important because the phenomenon of implicit aspects is present in nearly every opinionated document. For example, the following sentence uses only implicit aspects:

> *"This is the best phone one could have. It has all the features one would need in a cellphone: It is lightweight, sleek and attractive. I found it very user-friendly and easy to manipulate; very convenient to scroll in the menu..."*

Here, the word "lightweight" refers to the phone aspect *weight*; the words "sleek" and "attractive" to its *appearance*; the compound "user-friendly" to its *interface*; the phrase "easy to manipulate" to its *functionality*; finally, the phrase "to scroll in menu" can be interpreted as a reference to the *interface* of the phone or its *menu*.

Even though the aspects *appearance*, *weight* and *interface* do not appear in the sentence, the context contains clues that permit us to infer them. Namely, the words "sleek", "lightweight", and "user-friendly" that do occur in the context suggest these aspects.

In contrast to the task of identification of explicit aspects, the general scheme for identification of implicit aspects, a task called implicit aspect extraction, involves two steps:

1. Identify the words or expressions in the opinionated document that suggest an aspect (e.g. "sleek").

2. Map them to the corresponding aspects (*appearance*).

This research work presents a novel approach for each task. Regarding the first one, the present research work proposes a new model and framework for this phenomenon. We call the words or expressions that suggest the implicit aspects of the opinionated document, Implicit Aspect Indicators (IAI). For the second task, this research work proposes a novel approach that consist in mapping IAI to implicit aspects using semantic similarity with vector representation of words. Implicit aspect extraction will be discussed in detail in sec. 3.4

1.8.2.1. Implicit Aspect Indicators Extraction

As mentioned earlier, identify IAI in an opinionated document is a difficult task. This is because the syntactic and semantic properties of IAI are varied and extensive.

An IAI could be a single word, such as “sleek”, a compound, such as “user-friendly”, or even a complete phrase, such as “to scroll in menu”.

They can be of different parts of speech: in the example “This MP3 player is really expensive” the IAI “expensive” suggesting the aspect *price* is an adjective; in “This camera looks great” the IAI “look” suggesting *appearance* is a verb; in “I hate this phone. It only lasted less than six months!”, the IAI “lasted” suggesting *durability* of the phone is a verb. The following examples shows IAI as nouns or noun phrases: in “Even if I had paid full price I would have considered this phone a good deal” the IAI “good deal” suggest the aspect *price*; in “Not to mention the sleekness of this phone” the IAI “sleekness” suggest the aspect *appearance*; in “The player keeps giving random errors” the IAI “random errors” suggest the aspect quality; in “This phone is a piece of crap” the IAI “piece of crap” suggest the aspect quality.

Moreover the same implicit aspect can be implied by different IAI. In the following opinions, the implicit aspect *price* is implied by different IAI (underlined):

- This mp3 player is very affordable.
- This mp3 player also costs a lot less than the iPod.
- This mp3 player is quite cheap.
- This mp3 is inexpensive.
- I bought this mp3 for almost nothing!
- This mp3 player has been fairly innovative and reasonably priced.

1.8.2.2. IAI Extraction

A common approach for IAI identification is to assume that the sentiment or polarity words are good candidates for IAI: for example, in "This MP3 player is really expensive" the word "expensive", which indicates the negative polarity, is also the IAI for the aspect price.

However, this is not always true. For example, in "This camera looks great" the word "looks" implies the appearance of the phone, while the polarity is given by the word "great". In "I hate this phone. It only lasted less than six months!", the word "lasted" is the IAI for durability of the phone, while the polarity is indicated by "hate". What is more, the second sentence of this example could appear without the first one:, i.e."This phone only lasted less than six months". This sentence by itself stills constitute a negative opinion of the phone's durability, but not expressed by any specific word. This phenomenon is known in Opinion Mining as *desirable facts*: communicating fact that by common sense are good or bad, which indirectly implies polarity. Another example: in a camera review, the objective fact "The camera can hold lots of pictures" does not contain any sentiment or polarity words yet gives a positive opinion about the camera's memory capacity (IAI "hold") because it is desirable for a camera to hold many pictures.

In this research work, we present a novel approach for IAI extraction. The sec. 3.4 explains this in detail.

1.9. Aspect-based Opinion Summarization

Aspect-based opinion summarization has two main characteristics. First, it captures the essence of opinions: opinion targets (entities and their aspects) and sentiments about them. Second, 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. The quantitative side is crucial because of the subjective nature of opinions. The resulting opinion summary is a form of structured summary

Digital Camera 1:

Aspect: **GENERAL**

Positive:	105	<individual review sentences>
Negative:	12	<individual review sentences>

Aspect: **Picture quality**

Positive:	95	<individual review sentences>
Negative:	10	<individual review sentences>

Aspect: **Battery life**

Positive:	50	<individual review sentences>
Negative:	9	<individual review sentences>

...

Figure 1.2.: An aspect-based opinion summary

produced from the opinion quintuple (1.1). The Fig. 1.2 shows an aspect-based summary of opinions about a digital camera (Hu and Liu (2004)). The aspect GENERAL represents opinions on the camera as a whole, i.e., the entity. For each aspect (e.g., picture quality), it shows how many people have positive and negative opinions respectively. <individual review sentences> links to the actual sentences (or full reviews or blogs).

In fact, the opinion quintuples allows one to provide many more forms of structured summaries. For example, if time is extracted, one can show the trend of opinions on different aspects. Even without using sentiments, one can see the buzz (frequency) of each aspect mentions, which gives the user an idea what aspects people are most concerned about.

Even though Aspect-based opinion summarization research focuses on structured summaries, it is possible to come up with a more "natural" approach, e.g., a summary in a form of a short text with the main phrases, entities and aspect. The nature of Aspect-based Opinion Mining allows to make this kind of summaries since the granularity level is detailed enough to discriminate between the different elements in the opinion quintuples that conforms an opinionated document. Moreover, the structured nature of the opinion definition given by the opinion quintuple allows to use an optimization approach to generate such summaries. This approach

is also used in this work.

2. State of the Art

2.1. Overview

This chapter shows an overview of the current research made in Opinion Mining focused in Aspect-based Opinion Summarization, which is the approach this work uses. The state-of-the-art for each one of the task involved in opinion summarization are described, along with a brief description of the models, methods and the phenomena involved.

2.2. Sentiment Classification

2.2.1. Supervised Approaches

The current main technique is to use parsing to determine the dependency and the other relevant information. For example, Jiang et al. (2011) used dependency parserto generate a set of aspect dependent features for classification. A related approach was also used by Boiy and Moens (2009), which weights each feature based on the position of the feature relative to the target aspect in the parse tree. For comparative sentences, "than" or other related words can be used to segment a sentence (Ding and Zhang (2009); Ganapathibhotla and Liu (2008)).

Socher et al. (2013) developed a Recursive Neural Tensor Network (RNTN) and a sentiment Treebank. The RNTN was trained with the treebank in order to compute a semantic vector space of words that captures the sentiment compositionality

of large sentences. Then the sentiment of each element in a context is computed using a RNTN language model.

2.2.2. Lexicon-based Approaches

Ding and Yu (2008) proposed a word distance-based metric to determine the scope of each individual sentiment word. In this case, parsing is needed to find the dependency. The basic method is to use the word distance between sentiment words and the aspects of the sentence

We can also automatically discover the sentiment orientation of context dependent words, e.g. the word "long" in the sentence "The battery life is long" has a positive sentiment since the word "long" is a good thing in a battery. Blair-Goldensohn et al. (2008) integrated the lexicon-based method with supervised learning. Kessler and Nicolov (2009) experimented with four different strategies of determining the sentiment on each aspect/target (including a ranking method). They also showed several interesting statistics on why it is so hard to link sentiment words to their targets based on a large amount of manually annotated data. Along with aspect sentiment classification research, researchers also studied the aspect sentiment rating prediction problem which has mostly been done together with aspect extraction in the context of topic modeling. As indicated above, apart from sentiment words and phrases, there are many other types of expressions that can convey or imply sentiments. Most of them are also harder to handle.

2.3. Aspect Extraction

In the following we discuss the related work regarding opinion aspect extraction in opinion mining. Thera are two task in Aspect Extraction: Explicit Aspects Extraction and Implicit Aspects Extraction.

2.3.1. Explicit Aspect Extraction

2.3.1.1. Rule-based approaches.

Aspect extraction from opinions was first studied by Hu and Liu (2004). They introduced the distinction between explicit and implicit aspects (this will be discussed in detail in sec. 3.4). However, the authors only dealt with explicit aspects. Nouns and noun phrases (or groups) were identified by a part-of-speech (POS) tagger. Their occurrence frequencies are counted, and only the frequent ones are kept. A frequency threshold can be decided experimentally.

This method was improved by Popescu and Etzioni (2005). Their algorithm tried to remove those noun phrases that may not be aspects of entities. It evaluated each discovered noun phrase by computing a pointwise mutual information (PMI) score between the phrase and some meronymy discriminators associated with the entity class, e.g., a camera class. The meronymy discriminators for the camera class are, "of camera," "camera has," "camera comes with," etc., which were used to find components or parts of cameras. Web search was used to find the number of hits of individual terms and also their co-occurrences. The idea of this approach is clear. If the PMI value of a candidate aspect is too low, it may not be a component of the product because a and d do not co-occur frequently. The algorithm also distinguishes components/parts from attributes using WordNet's is-a hierarchy (which enumerates different kinds of properties) and morphological cues (e.g., "-iness," "-ity" suffixes).

Blair-Goldensohn et al. (2008) refined the frequent noun and noun phrase approach by considering mainly those noun phrases that are sentiment bearing sentences or in some syntactic patterns which indicate sentiments. Several filters were applied to remove unlikely aspects, e.g., dropping aspects which do not have sufficient mentions along-side known sentiment words. They also collapsed aspects at the word stem level, and ranked the discovered aspects by a manually tuned weighted sum of their frequency in sentiment-bearing sentences and the type of sentiment phrases/patterns, with appearances in phrases carrying a greater weight.

A frequency-based approach was also taken by Ku et al. (2006). The authors called the so discovered terms the major topics. Their method also made use of the TF-IDF scheme considering terms at the document level and at the paragraph level.

Guo et al. (2009) proposed a method based on the Cvalue measure for extracting multi-word aspects. The Cvalue method is also based on frequency, but it considers the frequency of multi-word term t, the length of t, and also other terms that contain t. However, Cvalue only helped find a set of candidates, which is then refined using a bootstrapping technique with a set of given seed aspects. The idea of refinement is based on each candidate's co-occurrence with the seeds.

Long et al. (2010) extracted aspects (nouns) based on frequency and information distance. Their method first finds the core aspect words using the frequency-based method. It then uses the information distance to find other related words to an aspect, e.g., for aspect price, it may find "$" and "dollars". All these words are then used to select reviews which discuss a particular aspect most.

Other rule-based methods are described by Kobayashi et al. (2006) and by Somasundaran et al. (2009).

2.3.1.2. Supervised Approaches

Taking a supervised learning approach, aspect extraction can be seen as a general information extraction problem. There are many algorithms for this task. The most dominant methods are based on *sequential labeling.* Since they are supervised methods, they need manually labeled training data. There are basically two techniques for this task: *Hidden Markov Models* (Rabiner (1989)) or HMM's and *Conditional Random Fields* (Lafferty et al. (2001)) or CRF's. Jin and Ho (2009) used a lexicalized HMM for extraction of aspects and opinions jointly.

Jakob and Gurevych (2010) used CRF's. They trained a CRF classifier on review sentences from Hu and Liu (2004). The reviews were from different domains for a more domain independent extraction. A set of domain independent features were

also used, e.g. tokens, POS tags, syntactic dependency, word distance, and opinion sentences.

Li et al. (2010a) integrated two CRF variations: Skip-CRF and Tree-CRF. These variations enable CRF to exploit structure features. Unlike the original CRF, which can only use word sequences in learning, Skip-CRF and Tree- CRF enable CRF to exploit structure features.

Choi and Cardie (2010) used CRF to determine the boundaries of an opinion, its polarity and its intensity. Huang et al. (2012) used CRF with syntactic dependency distributional context extraction features. They also categorize the features.

2.3.1.3. Unsupervised Approaches

There are also unsupervised learning techniques for aspect extraction. They are mainly based on topic modeling. This is an unsupervised learning method that assumes each document consists of a mixture of topics and each topic is a probability distribution. The random variables are words. A topic model is a generative model which describes how a document is generated according the probability distributions for each topic.

The main techniques for this approach are PLSA (Hofmann (1999)) and LDA (Blei et al. (2003)). These are graphical models based in Bayesian Networks and they are applied in aspect extraction under the assumption that the topics of a opinionated document are the opinion targets or aspects. These techniques have the advantage of clustering synonym aspects.

Although they are mainly used to model and extract topics from text collections, they can be extended to model many other types of information simultaneously. For example, in the sentiment analysis context, one can design a joint model to model both sentiment words and topics at the same time, due to the observation that every opinion has a target.

Intuitively topics from topic models are aspects in the sentiment analysis context. Topic modeling can thus be applied to extract aspects. However, there is also a

difference. That is, topics can cover both aspect words and sentiment words. For sentiment analysis, they need to be separated. Such separations can be achieved by extending the basic model (e.g., LDA) to jointly model both aspects and sentiments.

There are several works based on this approach. Mei et al. (2007) proposed a joint model for sentiment analysis based on PLSA. Lin and He (2009) proposed a joint topic-sentiment model by extending LDA, where aspect words and sentiment words were not explicitly separated. Brody and Elhadad (2010) identified aspects using topic modeling and then identify aspect-specific sentiment words by considering adjectives only. Li et al. (2010b) proposed two joint models: Sentiment-LDA and Dependency-Sentiment-LDA to find aspects with positive and negative sentiments.

In spite of being really useful, topic modeling have several disadvantages in aspect extraction. Titov and McDonald. (2008) showed that global topic modeling such as LDA might not be suitable for detecting aspects. This is because LDA relies on word co-occurrence frequencies among documents to model topics and word probability distributions. But opinion documents and reviews are quite homogeneous, meaning that every documents talks about the same aspects. This makes topic modeling very ineffective. The authors then proposed the multigrain topic models. The global model discovers entities while the local model discovers aspects using a few sentences (or a sliding text window) as a document. Here, each discovered aspect is a unigram language model, i.e., a multinomial distribution over words. Different words expressing the same or related facets are automatically grouped together under the same aspect. However, this technique does not separate aspects and sentiment words

Other problem is that topic modeling needs large volumes of data and a significant amount of tuning in order to achieve reasonable results. Also, topic modeling relies on Gibbs Sampling, which can throw different results each run due to Markov Chain Monte Carlo Sampling. While it is not hard for topic modeling to find those very general and frequent topics or aspects from a large document collection, it is not easy to find those locally frequent but globally not so frequent aspects. Such locally frequent aspects are often the most useful ones for applications because they

are likely to be most relevant to the specific entities that the user is interested in. Those very general and frequent aspects can also be easily found by the methods discussed earlier. These methods can find less frequent aspects as well without the need of a large amount of data. Topic modeling is very useful, but the current state-of-the-art approaches for sentiment analysis, and aspect extraction in particular, are far from being practical.

2.3.2. Implicit Aspect Extraction

In this section we discuss the related work regarding implicit aspect extraction in opinion mining.

The OPINE extraction system proposed by Popescu and Etzioni (2005) was the first which tried to extract implicit aspects for a better performance in polarity classification. There are not enough published information on the methods and techniques used in their approach or the implicit aspects extraction performance and the OPINE system is not publicly available.

In Su et al. (2008) a clustering method was proposed to map implicit aspect expressions, which were assumed to be sentiment words, to their corresponding explicit aspects. The method exploits the mutual reinforcement relationship between an explicit aspect and a sentiment word forming a co-occurring pair in a sentence.

In the work of Hai et al. (2011) a two-phase co-occurrence association rule mining approach was proposed to match implicit aspects (which are also assumed to be sentiment words) with explicit aspects.

Finally the work of Zeng and Li (2013) proposes a rule-based method to extract explicit aspects and then map implicit features using a set of sentiment words and clustering explicit features-opinions word pairs.

2.4. Aspect-based Opinion Summarization

The vast majority of research made in Aspect-based Opinion Summarization focuses on structured summaries. The approach proposed in Hu and Liu (2004) was the first one. Liu et al. (2005) proposed another approach with some visualization techniques. The Fig. 2.1 shows how the opinion summary in Fig. 1.2 could be visualized as a bar chart. Fig. 2.2 shows the visual opinion comparison of two cameras. We can see how consumers view each of them along different aspect dimensions including the entities themselves.

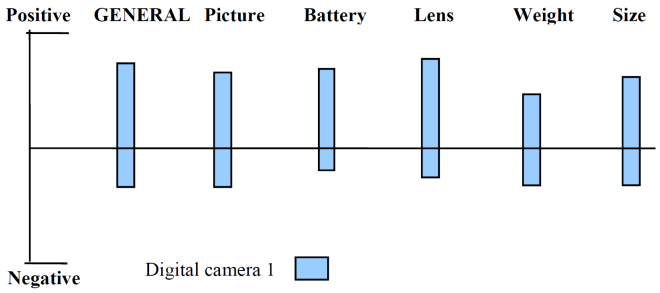


Figure 2.1.: Visualization of aspect-based summary of opinions on a digital camera

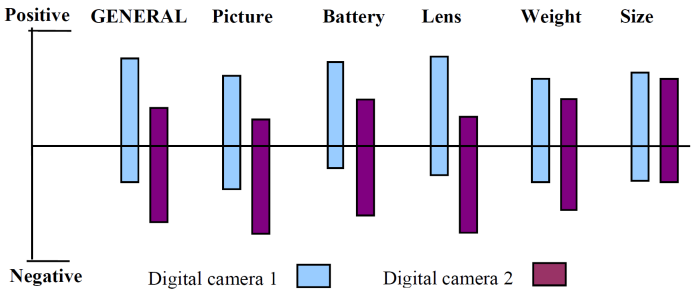


Figure 2.2.: Visual opinion comparison of two digital cameras

Several improvements and refinements have been proposed by researchers for the basic aspect-based summary. Carenini and Pauls (2006) proposed to integrate aspect-based summarization with two traditional text summarization approaches of factual documents, i.e., sentence selection (or extraction) and sentence generation.

Tata and Eugenio. (2010) produced an opinion summary of song reviews similar to that in Hu and Liu (2004), but for each aspect and each sentiment (postive or negative) they first selected a representative sentence for the group. The sentence should mention the fewest aspects (thus the representative sentence is focused). They then ordered the sentences using a given domain ontology by mapping sentences to the ontology nodes. The ontology basically encodes the key domain concepts and their relations. The sentences were ordered and organized into paragraphs following the tree such that they appear in a conceptually coherent fashion.

Lerman and McDonald (2009) defined opinion summarization in a slightly different way. Given a set of documents D (e.g., reviews) that contains opinions about some entity of interest, the goal of an opinion summarization system is to generate a summary S of that entity that is representative of the average opinion and speaks to its important aspects. This paper proposed three different models to perform summarization of reviews of a product. All these models choose some set of sentences from a review. The first model is called sentiment match (SM), which extracts sentences so that the average sentiment of the summary is as close as possible to the average sentiment rating of reviews of the entity. The second model, called sentiment match + aspect coverage (SMAC), builds a summary that trades-off between maximally covering important aspects and matching the overall sentiment of the entity. The third model, called sentiment-aspect match (SAM), not only attempts to cover important aspects, but cover them with appropriate sentiment. A comprehensive evaluation of human users was conducted to compare the three types of summaries. It was found that although the SAM model was the best, it is not significantly better than others.

In Nishikawa et al. (2010) a more sophisticated summarization technique was proposed, which generates a traditional text summary by selecting and ordering sentences taken from multiple reviews, considering both informativeness and readability of the final summary. The informativeness was defined as the sum of frequency of each aspect-sentiment pair. Readability was defined as the natural sequence of sentences, which was measured as the sum of the connectivity of all adjacent sentences in the sequence. The problem was then solved through optimization. Nishikawa et al. (2010a) further studied this problem using an integer

linear programming formulation.

3. Research Methodology

3.1. Overview

This chapter describes the research made and the methodology used to build the Opinion Summarization framework. It describes the models and methods proposed for polarity classification, aspect Extraction and opinion summarization.

3.2. Polarity Classification

3.2.1. Description

The approach used for polarity classification is the one proposed by Ding and Yu (2008). This is a holistic lexicon-based approach that exploits external evidences and linguistic conventions of natural language expressions. This approach allows the system to handle opinion words that are context dependent, which cause major difficulties for existing algorithms. It also deals with many special words, phrases and language constructs which have impacts on opinions based on their linguistic patterns. It also has an effective function for aggregating multiple conflicting opinion words in a sentence.

This method is used because:

- It is domain independent, unlike supervised methods which require training data. Therefore this restrict the domains the classifier can work based on the data domain.

- It has a very good performance.
- It is computationally inexpensive.

The main drawbacks found are the following:

- It requires the opinion targets as input. If the opinion targets are incorrect, or there are not any, the performance decreases drastically.
- It depends on a sentiment lexicon, where there can be some ambiguity of the polarity of words based on the contexts and the domain.

3.2.2. Algorithm

This method can be described in four steps:

1. Mark sentiment words and phrases: For each sentence that contains one or more aspects, this step marks all sentiment words and phrases in the sentence. Each positive word is assigned the sentiment score of +1 and each negative word is assigned the sentiment score of -1. For example, we have the sentence, "The *voice quality* of this phone is not good, but the *battery* life is long." After this step, the sentence becomes "The voice quality of this phone is not good [+1], but the battery life is long" because "good" is a positive sentiment word (the aspects in the sentence are italicized). Note that "long" here is not a sentiment word as it does not indicate a positive or negative sentiment by itself in general, but we can infer its sentiment in this context.
2. Apply sentiment shifters: Sentiment shifters (also called valence shifters) are words and phrases that can change sentiment orientations. There are several types of such shifters. Negation words like not, never, none, nobody, nowhere, neither, and cannot are the most common type. This step turns our sentence into "The voice quality of this phone is not good[-1], but the battery life is long" due to the negation word "not.". Note that not every appearance of a sentiment shifter changes the sentiment orientation, e.g.,

"not only ... but also." Such cases need to be dealt with care. That is, such special uses and patterns need to be identified beforehand.

3. Handle but-clauses: Words or phrases that indicate contrary need special handling because they often change sentiment orientations too. The most commonly used contrary word in English is "but". A sentence containing a contrary word or phrase is handled by applying the following rule: the sentiment orientations before the contrary word (e.g., but) and after the contrary word are opposite to each other if the opinion on one side cannot be determined. The if-condition in the rule is used because contrary words and phrases do not always indicate an opinion change, e.g., "Car-x is great, but Car-y is better." After this step, the above sentence is turned into "The voice quality of this phone is not good[-1], but the battery life is long[+1]" due to "but" ([+1] is added at the end of the but-clause). Notice here, we can infer that "long" is positive for "battery life". Apart from but, phrases such as "with the exception of," "except that," and "except for" also have the meaning of contrary and are handled in the same way. As in the case of negation, not every but means contrary, e.g., "not only ... but also." Such non-but phrases containing "but" also need to be identified beforehand.

4. Aggregate opinions: This step applies an opinion aggregation function to the resulting sentiment scores to determine the final orientation of the sentiment on each aspect in the sentence. Let the sentence be s, which contains a set of aspects $a_1, \ldots, a_m$ and a set of sentiment words or phrases $sw_1, \ldots, sw_n$ with their sentiment scores obtained from steps 1- 3. The sentiment orientation for each aspect a_i in s is determined by equation 3.1, where sw_j is a sentiment word/phrase in s, $dist(sw_j, a_i)$ is the distance between aspect a_i and THE sentiment word sw_j in s. $score(sw_j)$ is the sentiment score of sw_i. The multiplicative inverse is used to give lower weights to sentiment words that are far away from aspect a_i. If the final score is positive, then the opinion on aspect a_i in s is positive. If the final score is negative, then the sentiment on the aspect is negative. It is neutral otherwise.

$$score(a_i, s) = \sum_{sw_j \epsilon S} \frac{score(sw_j)}{dist(sw_j, a_i)} \tag{3.1}$$

The Algorithm 3.1 shows the polarity computation. The variable *orientation* holds the the polarity given as a result in the output. The Algorithm 3.2 describes how the polarity of a given word is computed.

The negation word or phrase usually reverses the opinion expressed in a sentence. Negation words include traditional words such as “no”, “not”, and “never”, and also pattern-based negations such as “stop” + “vb-ing”, “quit” + “vb-ing” and “cease” + “to vb”. Here, vb is the POS tag for verb and “vb-ing” is vb in its -ing form (represented in the Penn Treebank POS tag set as “VBG”). The following rules are applied for negations:

- Negation Negative → Positive
- Negation Positive → Negative
- Negation Neutral → Negative

For pattern-based negations, the algorithm detects the patterns and then applies the rules above. For example, the sentence, “the camera stopped working after 3 days”, conforms to the pattern “stop”+“vb-ing”, and is assigned the negative orientation by applying the last rule as “working” is neutral. Note that “Negative” and “Positive” above represent negative and positive opinion words respectively.

This approach proposes a special handling for the opinionated sentences with a “but” clause. The opinion before and after the word “but” are usually opposite to each other. Phrases such as “with the exception of”, “except that”, and “except for” behaves similarly to “but” and are handled in the same way as “but”.

The Algorithm 3.3 shows the “but” clauses handling. The intuition is that that one must follow the semantic orientation of the “but” clause first. If one cannot get an orientation there, then one must to look at the clause before“but”and negate its orientation. Non-but clauses containing but-like words: Similar to negations and opinion words, a sentence containing “but” does not necessarily change the opinion orientation. For example, “but” in “I not only like the picture quality of this camera, but also its size” does not change opinion after “but” due to the phrase “but also”.

In this research work some improvements where made to the original approach. In the Negation Rule handling, lemmatization to special negation words such as “stop”, “quit” and “cease” was added. This was made aiming to handled past tense sentence like “the camera stopped working after 3 days” better.

3.2.3. Implementation

The polarity classification tool was build using Python 2.7. The syntactic properties of this language makes the development easier. The Natural Language Toolkit (NLTK)[1] is a series of Python libraries that offers natural language processing capabilities. This toolkit was used for Part-Of-Speech Tagging (POS Tagging) and word lemmatization.

The Sentiment Lexicon used by Hu and Liu (2004) was used to determine the polarity of common sentimental words. Some tools for extracting and using these words were developed.

The performance analysis of the polarity classifier implementation is presented in the sec. 4.2.

3.3. Explicit Aspect Extraction

3.3.1. Description

As described in the previous section, the polarity classifier needs as input the words that represent the opinion explicit aspects of the document to classify. Moreover the Opinion Summarization approach proposed in this research work use these explicit aspects. Therefore, an explicit aspect extraction tool is required.

[1] http://www.nltk.org/

The proposed approach is to model explicit aspect extraction as a Name Entity Recognition task. This area has been widely researched an there are several techniques used. The most predominant are those based on sequence labeling. This is a machine learning technique that involves the algorithmic assignment of a categorical label to each member of a sequence of observed values.

3.3.2. Supervised Learning

In order to understand sequence labeling, a brief description of supervised learning is given.

Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Given a set on N of the form $\{(x_1, y_1), ..., (x_n, y_n)\}$ such that x_i is the feature vector of the i-th example and y_i is its label (i.e. the class), a learning algorithm seeks a function $g : X \to Y$, where X is the input space and Y is the output space. The function g is an element of some space of possible functions G, usually called a hypothesis space. It is sometimes convenient to represent g using a scoring function $f : X \times Y \to \mathbb{R}$ such that g is defined as returning the y value that gives the highest score.

3.3.3. Sequence Labeling

In the following an overview of sequence labeling scheme is given. It is out of the scope of this work to make a deeper explanation on this topic.

In sequence labeling we have a dataset with a set of inputs $X = \{x_1, ..., x_m\}$ and a set of assigned labels $Y = \{y_1, ..., y_m\}$ to those inputs. The goal is to predict the set of labels $Y' = \{y_{m+1}, ..., y_n\}$ given a set of new inputs $X' = \{x_{m+1}, ..., x_n\}$. If the model used to predict Y' is obtained with the observed data X and Y then the model is supervised. This is the approach used in this work.

There are 2 models currently used for sequence labeling and supervised learning in general: generative models and discriminative models. This research work only focuses on the discriminative approach since this is the one used.

3.3.3.1. Discriminative Sequence Labeling

Discriminative models, also called conditional models, are mathematical models for modeling the dependence of an unobserved data Y on an observed variable X. Within a probabilistic framework, this is done by modeling the conditional probability distribution $p(y|x)$, which can be used for predicting a random variable Y from a random variable X.

Discriminative sequence labeling tries to model a conditional probability distribution $P(y|x)$ for a set sequence of labels Y given a set of sequential data X.

The Fig. 3.1 shows an example of the basic Discriminative Sequence Labeling scheme used in the Part-Of-Speech Tagging. A sequence labeler has as input a sequence X. Then the labeler computes the conditional probability of a label for each element in X. The label sequence output is the sequence of labels that maximizes the probability.

3.3.4. Conditional Random Fields

The selected approach for aspect extraction task is discriminative sequential labeling. One of the state-of-the-art techniques for this approach are Conditional Random Fields (Lafferty et al. (2001)), commonly known as CRF. Concretely, this work uses the Linear Chain Conditional Random Fields Sequential Labeling

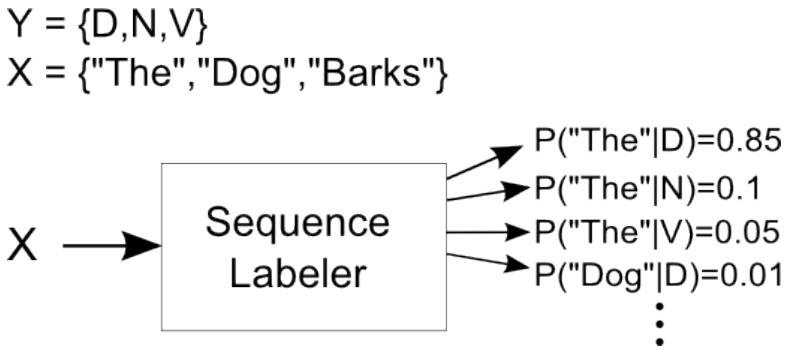


Figure 3.1.: Sequence Labeling in POS Tag

method. In order to understand better what CRF are, the Appendix A gives a brief introduction on Probabilistic Graphical Models and the full description of CRF.

3.3.5. Explicit Aspect Extraction Approach

In this section we describe the approach used for explicit aspect extraction. The proposed approach uses Linear Chain Conditional Random Fields Sequential Labeling. In the following it is described the dataset used in the experiments for training and testing, the features crafted and the implementation of the CRF-based aspect extraction tool.

3.3.5.1. Dataset

We used the corpus from the work of Hu and Liu (2004) in our experiments. It consist of 314 amazon reviews of 5 products in the electronics commodities domain: a DVD player(DVD), a Canon camera (Canon), a MP3 player (MP3), a Nikon camera (Nikon) and a Nokia Cellphone (Phone). The Tab. 3.1 shows some of the properties of this corpus. The field Rev show the number of reviews per document. Sent is how many sentences are in the document. Words is the number of words in the whole document. WPR is Words per Review, and it is on average

how many words a review has. ES shows how many sentences of the documents have at least one explicit aspect. The field EA Dict. indicates how many unique words were seen as an explicit aspect. Finally EA % shows the percentage between sentences with explicit aspects versus the rest of the sentences.

Table 3.1.: Dataset Statistical Properties.

	DVD	Canon	MP3	Nikon	Phone
Rev	99	45	95	34	41
Sent	740	597	1796	346	546
Words	56661	55638	149676	31416	44497
WPR	572.3	1236.4	1575.5	924	1085.3
ES	345	239	721	160	266
EA Dict.	116	105	188	75	111
EA %	46.62%	40.03%	42.01%	46.24%	48.71%

The dataset has the explicit aspects labeled in each opinionated sentence. Also unmentioned aspects are indicated, i.e. pronouns that indicate an aspect mentioned earlier in the text or implicit aspects. Explicit aspect extraction only uses the aspects that appear labeled in the sentence.

The structure of this dataset is as follows: A review starts with a string "[t]" followed by the title of the review. Then each line is a sentence in the original review. The review finishes when the string "[t]" is found. This line is the title of the next review in the file. Each sentence has a 2-pound string ("##"). If the sentence has explicit aspects, they appear in the left side of the "##" string as a comma-separated list of words. If the sentence does not have any aspects, the string "##" is at the beginning of the sentence

The Fig. 3.2 shows an example of a sentence in a review. We have an opinionated sentence in this example . It has a positive opinion about the "audio output" aspect. This sentence is shown in 4 lines due to space issues.

```
audio output##another nice thing is that the unit has both
optical and coax digital audio outputs , though the latter
was not mentioned in the literature i'd scanned before
buying .
```

Figure 3.2.: Example of a sentence in the reviews dataset.

3.3.5.2. Feature Crafting

The proposed features uses different linguistic and syntactic properties of the text and the model itself. These properties captures the opinion target phenomena effectively. This linguistic properties are:

- Part Of Speech: POS is a linguistic category of words (or more precisely lexical items), which is generally defined by the syntactic or morphological behavior of the lexical item in question. Common linguistic categories include noun and verb, among others. The POS Tagging used is based on the Penn Treebank tagging scheme. For every classification instance the POS tagging is performed.

- Character N-Grams: Character N-Gram are fragments of a word. They consist the different segments of characters of length N. These kind of features describe the text morphological properties.

Other proposed features try to model the context of an instance:

- Context: These features are the word and tag for the previous and next word of the current instance to be labeled .

- Class sequences: These are the combination of a particular word with the labels given to the previous words. We used a label window of 2, i.e. the labels of the 2 previous words plus the current word and as features.

- Disjunction of words: these are features that gives disjunction of words in the context of the token with an specific window size. The Fig. 3.3 gives an example of these features. At training time we have a sentence "the best feature of my phone is the camera" and the token being evaluated is "camera" with a class label of 'A' indicating it is an aspect. In a disjunction window size of 3, the features functions are built to compute the disjunction of the previous 3 words and the next 3 words of the word "camera". In this case, the function f_1 will map to 1 if the previous word is "the" and the current class label is 'A' or if the previous 2 words are "is the" and the current label is 'A' or if the previous 3 words are "phone is the" and the current label is 'A'. Since there are no more words after the word "camera", there is no function modeling the disjunction words for the next 3 words.

$$f_1(y_j, \vec{x}, j) = \begin{cases} 1 & if\, y_j = 'A'\, and \\ & (\vec{x}_{j-1} = "the" \\ & or\, \vec{x}_{j-2,j-1} = "is\, the" \\ & or\, \vec{x}_{j-3,\,j-2,\,j-1} = "phone\, is\, the") \\ 0 & Otherwise \end{cases}$$

Figure 3.3.: Disjunction feature example.

The Tab. 3.2 show a detailed description of the features used in the experiments. Recalling from Subsection sec. A.5, at training time a feature function takes the current sentence token to be labeled (it sometimes takes the context tokens, i.e. the previous or the next token, the two previous tokens, etc.) or some property of this token (e.g. its POS tag) and the class label assigned (it can also take k previous labels). The field "Features" shows this information for each feature built. For example, the first row "word, class" indicates that there will be features indicating if a specific word given a class label exist.

Table 3.2.: Features Description

Features	Description
word, class	Vocabulary word
tag, class	Part of Speech
prev. word, class	The previous word to a token being labeled
next word, class	The next word to a token being labeled
prev. tag, class	The tag of the previous word to a token being labeled
next tag, class	The tag of the next word to a token being labeled
prev. word, word, class	Previous word pair
next word, word, class	Next word pair
(prev. word; prev. 2 words; prev. 3 words; ...), class	The disjunction of context words
prev. class, class	The sequence of class labels of order 1
prev. class, word, class	Combination of word and the sequence of class labels of order 1.
char-n-gram(word), class	The character n-grams of a word

3.3.5.3. Implementation

The Stanford NER[2] is a Java implementation of a Named Entity Recognizer. It includes a general implementation of an arbitrary-order linear-chain Conditional Random Field sequence model. It is called CRFClassifier. This work uses such implementation for the aspect extractor. This software allows to train a NER model with an annotated text. The training data is arranged in a tab-separated column text file. For convention, the most-left column is the text tokens, the most-right column is the class label and the inner columns are the token features, such as tag, lemma, chunk, etc. For the proposed features, the CRFClassifier needs the text token, the token POS tag and the class label as input. The Fig. 3.4 shows part of a CRFClassifier training data file. Each line is a token in a sentence. An empty line is used to indicate a new sentence. The first tab-separated column indicates the token, the second column is the POS tag of that particular token in the whole sentence context. The third column is the class label. The label 'A' indicates that

[2]http://nlp.stanford.edu/software/CRF-NER.shtml

the token is an explicit aspect. The label 'O' is for "Other", i.e. it is not an aspect.

i	PRP	O
think	VBP	O
apex	NN	A
is	VBZ	O
the	DT	O
best	JJS	O
dvd	NN	A
player	NN	A
you	PRP	O
can	MD	O
get	VB	O
for	IN	O
the	DT	O
price	NN	O
.	.	O

Figure 3.4.: A sentence in a CRFClassifier training data file.

In order to train the CRFClassifier with the corpus of Hu and Liu (2004), a parser tool was built as part of this research. It uses the files of the corpus as input and creates a CRFClassifier training data file. The NLTK POS Tagger is used in this parser for POS Tagging.

The CRFClassifier can be used programatically as a Java class or it can be used out-of-the-box as an executable program. The latter option was used.

The command to train the CRFClassifier is shown in the Fig. 3.5. The executable file name is "StanfordNER.jar". The CRFClassifier can be configured using a configuration file. The name of this file can be set with the option "-prop". In the example given in theFig. 3.5 the configuration file name is "CRFAspectExtractor.prop"

```
java -cp StanfordNER.jar edu.stanford.nlp.ie.crf.CRFClassifier
-prop CRFAspectExtractor.prop
```

Figure 3.5.: CRFClassifier console training command

The StanfordNER package includes the documentation needed to set it up (in a Javadoc format). The CRFClassifier parameters used in this research work are described. It is out of the scope of this work a full description of the Stanford NER and the CRFClassifier capabilities.

The Fig. 3.6 shows the properties file used to train the CRFClassifier used as an aspect extractor. In the following these properties are listed and explained:

- trainFile: This is the training data file name. The .tsv extention is for convention. It can use any extension as long as the training data file has an ASCII or UTF-8 format.
- serializeTo: The CRFClassifier model can be serialized to a file. This parameter allows to set the file name of this file. If the .gz extension is added to the file name, the CRFClassifier will compress the model automatically.
- map: This set the training set structure mapping. It consist of a comma-separated set of assignations. The assignation left side is the data label and the right side is the column number counted from left to right.

The rest of the file is the CRFClassifier features configuration. These parameter are set according to the feature crafting described in sec. 3.3.5.2.

```
trainFile = CRFPOSTaggedCamTrainData.tsv
#location where you would like to save (serialize to) your
#classifier; adding .gz at the end automatically gzips
#the file,  making it faster and smaller
serializeTo = CRFAspectExtractor-model.ser.gz
#structure of your training file; this tells the classifier
#that the word is in column 0 and the correct answer is in
#column 1
map=word=0,tag=1,answer=2
#these are the features we'd like to train with
#some are discussed below, the rest can be
#understood by looking at NERFeatureFactory
useClassFeature=true
useWord=false
useNGrams=true
useTags=true
noMidNGrams=true
maxNGramLeng=6
usePrev=true
useNext=true
useSequences=true
usePrevSequences=true
useNextSequences=true
useDisjunctive=true
```

Figure 3.6.: CRFClassifier properties file

The command to test the trained model is shown in the Fig. 3.7. The "-loadClassifier" parameter specifies the model file name to be used. The "-testFile" indicates the data to label. The " > TestResults.txt" saves the labeled data in the file "TestResults.txt".

```
java -cp StanfordNER.jar edu.stanford.nlp.ie.crf.CRFClassifier
    -loadClassifier CRFAspectExtractor-model.ser.gz
    -testFile TestData.tsv > TestResults.txt
```

Figure 3.7.: CRFClassifier testing command

In order to validate the results and check for over-fitting, a 10-fold cross validation scheme was set up. It consists of a Python script that divides the dataset in 10 parts of the same size. Then it creates 10 batch files. Each batch file trains and test the CRFClassifier with a different part of the dataset, making this a fold. In each fold, the 80% of the dataset is used for training and the rest is used for testing.

The testing command gives the precision, recall and F1-Score measures at the end of the testing. These metrics were used to report the results.

The sec. 4.3 shows the performance analysis of the implementation of this explicit aspect extraction approach.

3.4. Implicit Aspect Extraction

Most of the research in Aspect-based Opinion Mining does not consider implicit aspects, mainly because the extraction of this type of aspects have been not studied enough and it is a phenomenon difficult to model. This research work does consider implicit aspects as part of aspect extraction.

As described in sec. 2.3.2, implicit aspect extraction is divided in 2 subtasks:

1. Identify the implicit aspect indicators (IAI) in the opinionated document.
2. Map them to the corresponding aspects.

This research work proposes a novel approach for both of these subtasks. The first one is called IAI extraction and the second one Implicit Aspect Extraction. The IAI extraction approach proposed is the state-of-the-art, as the results obtained prove it.

3.5. IAI Extraction

As described in sec. 1.8.2.2, this tasks consist on the identification of words in an opinionated document that suggests the opinion aspects, e.g. the sentence "This camera is sleek and very affordable" implicitly gives a positive opinion on the aspects *appearance* and *price* of the entity camera. One can infer these aspect because the words "sleek" and "affordable" suggest them.

The proposed approach is to model IAI extraction as a Name Entity Recognition task. The CRF-based framework proposed for explicit aspect extraction is used again to tackle this problem. As a part of this effort, a corpus for IAI extraction was developed by manually labeling IAI and their corresponding aspects in a well-known opinion-mining corpus. This is the first corpus that specifies implicit aspects along with their indicators.

3.5.1. IAI Corpus

It was noticed that there was no suitable dataset for IAI extraction. As explained in sec. 1.8.2, limited work has been done in extracting implicit aspects. Moreover, the task that this research work defines as IAI extraction was not defined since the common approach to infer implicit aspects was to take sentiment words as the best words to infer such aspects. Therefore, it is natural that there are no resources for IAI extraction. As a result of this, the first corpus for IAI extraction was created during this research work.

Hu and Liu (2004) developed a corpus for explicit aspect extraction. This corpus has been widely used in many opinion mining subtasks. The texts of this corpus

were used to create a new one for IAI extraction. The text of this corpus was labeled indicating the IAI and their corresponding implicit aspects. The sentences that have at least one implicit aspect were selected in order to label the corpus. Therefore, not every opinionated sentence is labeled.

The Tab. 3.3 shows some of the properties of the IAI corpus. It consists of 314 Amazon reviews of 5 products in the electronics commodities domain: a DVD player (the column "DVD" in the table), a Canon camera ("Canon"), an MP3 player ("MP3"), a Nikon camera ("Nikon") and a Nokia Cellphone ("Phone"). This table describes the number of reviews per document. It also describes how many words and sentences a review has on average.

The corpus statistical properties at different granularity levels are shown in Tab. 3.4. The the name of each column is the same as the name of the columns in Tab. 3.3. This table is divided in 3 section for each granularity level:

- Sentence level.
- Token level.
- Type level.

The sentence-level section shows how many sentences are in the document, how many sentences of the documents have at least one IAI (shown in the row labeled as "IAI#") and the percentage of sentences that have at least one IAI ("IAI%"). The token-level and type-level section describes the same properties for these granularity levels.

Table 3.3.: Corpus Properties.

	DVD	Canon	MP3	Nikon	Phone
Reviews	99	45	95	34	41
Words per Review	572.3	1236.4	1575.5	924	1085.3
Sentences per Review	7.47	13.26	18.90	3.64	13.31

The Tab. 3.5 shows the POS distribution for the IAI labeled in the corpus. Each row represents a general Penn Treebank POS tag. The first row represents all the

Table 3.4.: Statistical Properties.

	DVD	Canon	MP3	Nikon	Phone
Sentence level					
Sentences	740	597	1796	346	546
IAI#	147	63	155	36	44
IAI%	19.86%	10.55%	9.03%	10.40%	8.05%
Token level					
Tokens	56661	55638	149676	31416	44497
IAI#	164	79	214	50	66
IAI%	0.289%	0.141%	0.142%	0.159%	0.148%
Type level					
Types	1767	1881	3143	1285	1619
IAI#	72	63	136	40	42
IAI%	4.07%	3.34%	4.32%	3.11%	2.59%

tags that are adjectives (JJ, JJR, JJS), the second one represents the noun tags (NN,NNS, NNP, NNPS) and the third one represents the verb tags (VB, VBD, VBG, VBN, VBP, VBZ). The last row is the rest of tags seen with an IAI. The IAI column shows how many vocabulary words were seen labeled with the given tag. The third column describes how many words with the given tag were seen in sentences with IAI. The fourth column shows the tag distribution observed in the IAI.

Table 3.5.: Corpus POS Distribution

POS	IAI	POS in IAI Sentences	P(IAI | POS)
JJ	157	527	0.2818
NN	167	1692	0.3000
VB	220	1112	0.3826
other	19	3900	0.0346

3.5.2. Feature Crafting

All the features used for explicit aspect extraction are included in the features set for IAI extraction. These features model the syntactic, semantic and morphological properties of IAI:

1. Part Of Speech.
2. Character N-Gram.
3. Context features.
4. Class sequences.
5. Disjunction of words.

Other features were added. The following features gave a performance boost in IAI extraction.

- Word Pairs: These features are the combination of word and its class label with the previous word.
- Word-Tag combination: These are features crafted with all the combinations of word and tag seen.

The Fig. 3.8 shows the text of the CRFClassifier prop file for IAI extraction. The 4 last lines of the document are the parameters concerning the new features described above. The line "splitDocument=true" makes the classifier to consider the training file and the testing file as a set of documents. Each document is separated by a empty line. There is no gain in terms of labeling performance but it takes a little bit less to train the classifier with this parameter set to "true".

3.5.3. Implementation

The Stanford NER was used for the IAI extraction tool implementation. The IAI corpus has the same text structure than the original corpus with some minor

changes. Therefore the tools implemented for parsing the original corpus are used again to parse the IAI corpus. Also the tools created for cross-validation are used for experimental validation.

```
trainFile = TrainData0.tsv
#location where you would like to save (serialize to) your
#classifier; adding .gz at the end automatically gzips the file,
#making it faster and smaller
serializeTo = CRFImplicitAspectExtractor-model.ser.gz
#structure of your training file; this tells the classifier
#that the word is in column 0 and the correct answer is in
#column 1
map=word=0,tag=1,answer=2
#these are the features we'd like to train with
#some are discussed below, the rest can be
#understood by looking at NERFeatureFactory
useClassFeature=false
useWord=true
useNGrams=true
useTags=true
noMidNGrams=true
useDisjunctive=true
maxNGramLeng=6
usePrev=true
useNext=true
useSequences=true
usePrevSequences=true
maxLeft=1
useTypeSeqs=true
useWordPairs=true
useWordTag=true
splitDocuments=true
```

Figure 3.8.: CRFClassifier properties file for IAI extraction

3.5.4. Baselines

The IAI extraction approach proposed in this research work turned out to be the state-of-the-art. As explained in sec. 1.8.2.2 and sec. 2.3.2 the common approach for IAI identification is to assume that the sentiment or polarity words are good candidates for IAI: for example, in “This MP3 player is really expensive” the word “expensive”, which indicates the negative polarity, is also the IAI for the aspect *price*. Also it was explained that this is not always true. For example, in “This camera looks great” the word “looks” implies the appearance of the phone, while the polarity is given by the word “great”. The IAI extraction approach proposed does not do such assumptions since the objective is to find the best word that suggest the implicit aspects, whether these words are sentiment words or not.

Therefore, 3 baselines were implemented to compare the performance of our approach. These baselines were also the state-of-the-art since there was no other proposed methods for IAI extraction. These were developed under the assumption that sentiment words are the best candidates.

The first baseline is to label the sentiment words as IAI for each sentence in a review (Su et al. (2008); Hai et al. (2011); Zeng and Li (2013)). We use the sentiment lexicon used by Hu and Liu (2004) to determine if a word has an opinion polarity. This lexicon is conformed by two word lists. The first list is conformed by "positive words", which are words that suggest a positive opinion in an opinionated context (e.g. "awesome"). The second list is conformed by "negative words". These words suggest a negative opinion (e.g. "awful"). The algorithm for this baseline is as follows: for each word in a sentence we determine if this word is in any of the two list of the lexicon. If it is, we label it as IAI. We call this baseline BSLN1.

A second baseline based on text classification was proposed and it was based on a Naive Bayes (NB) text classifier. The classifier was trained with the texts of the IAI corpus. The task of this classifier is to determine whether a sentence has at least one IAI or not. If a sentence is classified as a one with IAI, we label the sentiment words as IAI.

The features used in the NB classier were:

- Corpus vocabulary stems. Stop words are excluded.
- The best 500 bi-gram collocations obtained by a Point-wise Mutual Information association measure.

Finally, we also implemented a second-order Hidden Markov Model sequence labeler. This is the standard method for sequence labeling. We called this method BSLN3. We trained this labeller with our corpus. Since the labeller is a second order HMM, we use bi-grams and trigrams as features. The training data is per-processed as follows:

- The words that appears less than 5 times in the corpus (rare words) are changed in the training data for the label *RARE*.
- The rare words that contains at least one numeric character are changed for the label *NUMERIC*
- The rare words that consists entirely of capitalized letters are changed for the label *ALLCAPS*

All baselines were implemented in Python. We used the NB Classifier included in NLTK for the BSLN2.

The sec. 4.4 shows the performance analysis of the proposed IAI extraction approach and the different baselines used.

3.6. Mapping IAI to Implicit Aspects

As explained in sec. 3.4, implicit aspect extraction is performed in 2 steps:

1. Extract the indicators of these aspects
2. Map these indicators to their corresponding implicit aspects.

Regarding step two, it is intuitively clear for a human that the word "expensive" in the sentence "This phone is very expensive" is more related to the aspect *price* than the aspect *appearance*. Therefore, given the words *price* and *appearance*, the natural inference process would be to associate the word "expensive" with *price.* However this association process is a non-trivial task for a computer.

To overcome this issue, the present research work proposes to use semantic relatedness. The main idea is to exploit the semantic relatedness of IAI with their implicit aspects to infer such aspects. Semantic relatedness is a metric defined over a set of documents or terms, where the idea of distance between them is based on the likeness of their meaning or semantic content. This can be achieved with mathematical tools that are used to estimate the strength of the semantic relationship between units of language, concepts or instances, through a numerical description obtained according to the comparison of information formally or implicitly supporting their meaning or describing their nature.

Many authors uses semantic similarity and semantic relatedness as synonyms. However, there are subtle differences between them. Pedersen et al. (2004) defines semantic similarity as a metric of relatedness between *is–a* relations. E.g. a car and a train are semantic similar since both have an is-a relation with the concept of transport, while the concepts of car and road are not semantically similar since they do not share such relation. Semantic relatedness is defined as a more general metric of a relation of two or more entities. In this case, the concepts of car and road can be considered related since cars can use a road, while the concepts of a train and a road are less related. This research work uses the concept of semantic relatedness.

There are many approaches to tackle semantic relatedness. It can be estimated by defining a topological similarity. This can be achieved using ontologies to define the distance between terms/concepts. Another approach is to use statistical similarity techniques such Latent Semantic Similarity, Latent Dirichlet Allocation, Pointwise Mutual Information and others.

Recent research made in distributed word representation has shown that linguistic regularities such as semantic relatedness can be embed in these representations.

Then they can be used to compute the semantic similarity between words. These techniques outperforms drastically the existing semantic similarity methods (Collobert et al. (2011); Mikolov et al. (2013a)).

Moreover, it was found out in the development of this research work that the common topological approaches performs very poorly in the inference of implicit aspects with IAI. The usual approaches for semantic similarity, like the Leacock similarity, Resnick similarity and others are based on ontologies, such as WordNet. These ontologies commonly organizes nouns and verbs into hierarchies of is–a relations. While WordNet includes adjectives and adverbs, these are not organized into is–a hierarchies so similarity measures can not be applied (Pedersen et al. (2004)). This is a serious drawback for the implicit aspect extraction approach proposed because adjectives are a key part of it. The Tab. 3.5 describes the POS distribution found in the IAI corpus. It is possible to observe that IAI that are adjectives are almost a third part of all IAI found. There are some techniques to overcome the issue of using topological-based approaches. However, they are far from being practical.

Therefore, distributed word representation is the approach chosen to compute the semantic similarity between IAI and their aspects. In the following an introduction to distributed word representation is given. It is out of the scope a full description of these techniques.

3.6.1. Word Representation

Representation of text is very important in many NLP applications. This is closely related to language modeling. There are several approaches for word representation. However they can be classified in two main groups:

- Local representations: These techniques considers words as the atomic units. There is no notion of similarity between words, as these are represented as an index in a vocabulary. This choice has several good reasons: simplicity, robustness and the observation that simple models trained on huge amounts

of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling. It is possible to train N-grams on virtually all available data. Similar to this is the Bag-of-Words (BOW) model and the 1-of-N coding model.

- Continuous representation: These techniques represents words as vectors of identifiers. They are closely related to the vector space model. Latent Semantic Analysis and its variations, Latent Dirichlet Allocation and Distributed Representations of Words are the common approaches.

With the progress of machine learning techniques in recent years, it has become possible to train more complex continuous representation models on much larger data set, and they typically outperform the simple local models. Probably the most successful concept is to use distributed representations of words. For example, neural network based language models significantly outperform N-gram models.

3.6.1.1. Distributed Word Representation

A distributed representation of a symbol is a tuple (or vector) of features which characterize the meaning of the symbol, and are not mutually exclusive. If a human were to choose the features of a word, he might pick grammatical features like gender or plurality, as well as semantic features like “animate" or “invisible”, morphological or even discourse properties. Unlike vector space model used in Information Retrieval, the dimensions of these vectors are not computed in terms of word frequencies. Each dimension of that space corresponds to a semantic or grammatical characteristic of words.

The main technique to compute these distributed representation of words are the Neural Network Language Models (NNLM).

3.6.2. Neural Network Language Models

A language model is a function, or an algorithm for learning such a function, that captures the salient statistical characteristics of the distribution of sequences of

words in a natural language, typically allowing one to make probabilistic predictions of the next word given the preceding ones.

A neural network language model is a language model based on Neural Networks, exploiting their ability to learn distributed representations to reduce the impact of the curse of dimensionality.

In the context of learning algorithms, the curse of dimensionality refers to the need for huge numbers of training examples when learning highly complex functions. When the number of input variables increases, the number of required examples can grow exponentially. The curse of dimensionality arises when a huge number of different combinations of values of the input variables must be discriminated from each other, and the learning algorithm needs at least one example per relevant combination of values. In the context of language models, the problem comes from the huge number of possible sequences of words, e.g., with a sequence of 10 words taken from a vocabulary of 100,000 there are 10^{50} possible sequences.

With a neural network language model, one relies on the learning algorithm to discover these features, and the features are continuous-valued. This is very useful because it makes the optimization problem involved in learning these features much simpler.

The basic idea is to learn to associate each word in the dictionary with a continuous-valued vector representation. Each word corresponds to a point in a feature space. The objective is that functionally similar words get to be closer to each other in that space, at least along some directions. The Fig. 3.9 shows an example of this concept. This is an euclidean space and each instance is described in terms of the dimensions X_1 and X_2. Terms that belongs to name of countries such "Germany" and "France" are grouped closer according to the euclidean distance. The terms that corresponds to names of the days appear grouped closer.

A sequence of words can thus be transformed into a sequence of these learned feature vectors. The neural network learns to map that sequence of feature vectors to a prediction of interest, such as the probability distribution over the next word in the sequence. What pushes the learned word features to correspond to a form

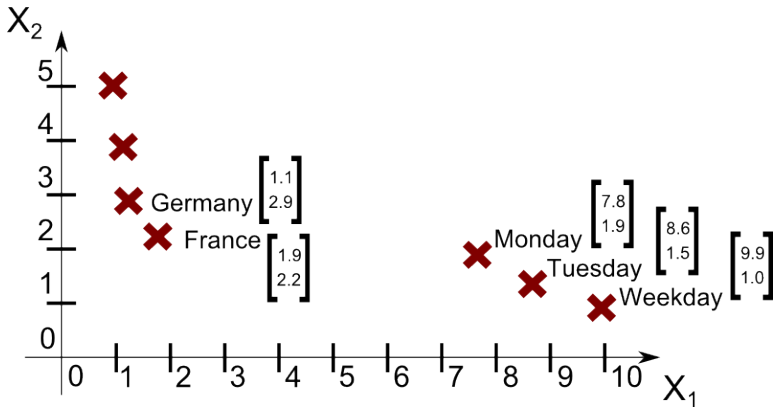


Figure 3.9.: A 2-dimensional feature space

of semantic and grammatical similarity is that when two words are functionally similar, they can be replaced by one another in the same context, helping the neural network to compactly represent a function that makes good predictions on the training set, the set of word sequences used to train the model. The Appendix B gives a brief description of neural networks.

3.6.3. Learning Word Representations with NNLM

Several NNLM have been proposed for Word Representation Learning. The general approach is described as follows.

Let D be a finite dictionary of the training data. Let w be a word such that $w \in D$. Given a task of interest, a relevant representation of each word is then given by a lookup table W, which is trained by backpropagation. More formally, for each word $w \in D$, an internal k-dimensional feature vector representation is given by the lookup table layer W_i where $W \in \Re^{k \times |D|}$. The parameter k is usually predefined and it will define the dimensionality of the feature vector for each word. In other words, for each $w \in D$, a k-dimensional vector is built. Then all the vectors are put together in a matrix W. The initial values of W are random generated, usually subject to a normal distribution $\Gamma(0, \epsilon^2)$ with a small ϵ value. The Fig. 3.10

shows a graphical representation of a lookup table W given the dictionary D. The dimensionality of each vector is defined by the hyperparameter $k = 5$.

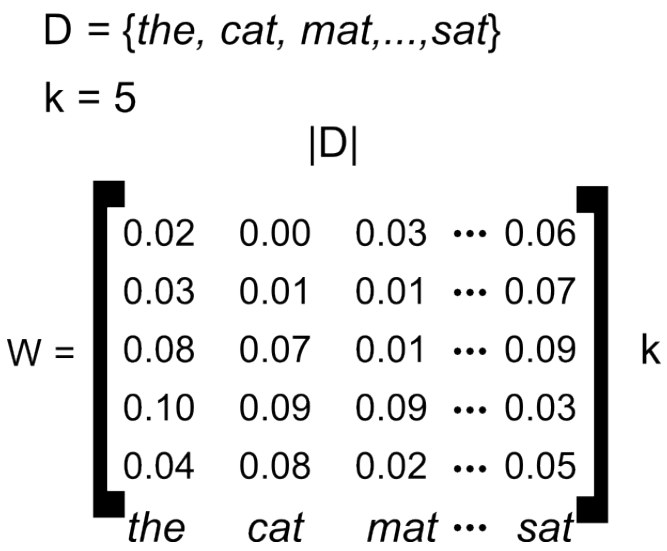


Figure 3.10.: A lookup table

From here, one can use the vectors in W as input to a neural network. The backpropagation algorithm (and the underlying architecture given by the NNLM used) will modify the values of W.

As a concrete example, an explanation of the NNLM proposed by Collobert et al. (2011) is given. The main idea behind this work is that a word and its context is a positive training sample; a random word in that same context gives a negative training sample. E.g. the sentence "cat chills on a mat" in the training data is a positive example. The word "on" is changed for the random dictionary word "Jeju" and the sentence "cat chills Jeju a mat" constitutes a negative training example.

The formal definition of this model is as follows. Let $score : \Re^{k \cdot l} \mapsto \Re$ be a function that determines the score of a sentence. To compute this score, a neural network is used. The score of a positive training example must be greater than the score of a negative training example. In the case when a positive example and a negative example are forward propagated and the result is that the score of the negative

example is greater than the score of the positive example, then there will be a loss in terms of a defined loss function. With this loss is possible to learn a better representation of the words by using the backpropagation algorithm.

A single layer in a neural network is a combination of a linear layer $z = Wx + b$ and a non-linearity $a = f(z)$ where $f(\cdot)$ is commonly the sigmoid function. The neural activations a can then be used to compute some function. For instance, the score of a positive or negative training example, i.e. $score(x) = U^T a \in \Re$.

The first step to compute the score of a sentence is to build the appropriate input. To describe a sentence with n tokens, retrieve (via index) the corresponding vectors from the lookup table W and concatenate them to a $n \cdot k$ vector. This vector will be forward propagated to a 3-layer neural network. The hidden layer will compute a, which would be used by the output vector to compute the score. The Fig. 3.11 describes conceptually the score computation with the sentence "cat chills on a mat", using the lookup table W shown in the Fig. 3.10, where the cardinality k of each feature vector was 5. In this example, the number of units in the hidden layer is 10.

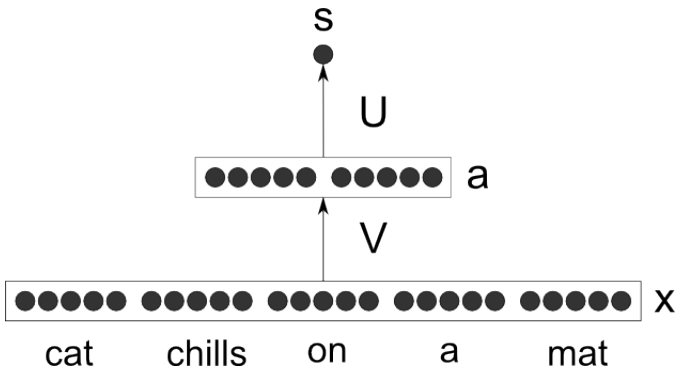


Figure 3.11.: Collobert et al. NNLM

The score is defined as:

$$s = U^T \cdot a \tag{3.2}$$

where $U \in \Re^{10 \times 1}$. The vector a is defined as:

$$a = f(Vx + b) \tag{3.3}$$

where $V \in \Re^{25\times10}$. Note that the input layer encodes a window of 5 words. Let l denote the window size, and l is a hyperparameter defined in the model. Thus the number of units in the input layer is defined by $l \cdot k$, i.e., the window size times the cardinality of the feature vectors from W.

This model proposes to compute the cost function as follows:

$$J = max(0, 1 - s + s_c) \tag{3.4}$$

where s is the score of a positive sentence and s_c is the score of a negative sentence. If the score of a positive sentence is greater than the score of a negative sentence, there will be no loss. However, if the score of a negative sentence is greater than the score of a positive sentence, there will be a loss, and it will be possible to learn from it with backpropagation. The function J is differentiable, therefore it is possible to train the model with gradient descend techniques.

Note that the backpropagation algorithm not only modifies the parameters U and V, but also the lookup table W. The intuition is that these modifications to the word vectors in W will make better word representations. Although the conceptual representation of the neural network described in the Fig. 3.11 does not show the W table as parameter of the model, the table W could be represented as the matrix of weights between the input layer and a previous layer where the window of words is encoded by a $|D|$-sized one-hot vector per word and a layer of size $|D| \cdot l$, i.e., for each word in a window of words of size l, create a $|D|$-sized one-hot vector and use it as input to a new $|D| \cdot l$-sized input layer. Therefore the old input layer becomes a *projection* layer, i.e, a layer that reduces the dimensionality of the input. The Fig. 3.12 shows conceptually the neural network with the new input layer and the projection layer. The matrix of weights W is the lookup table with the word representations.

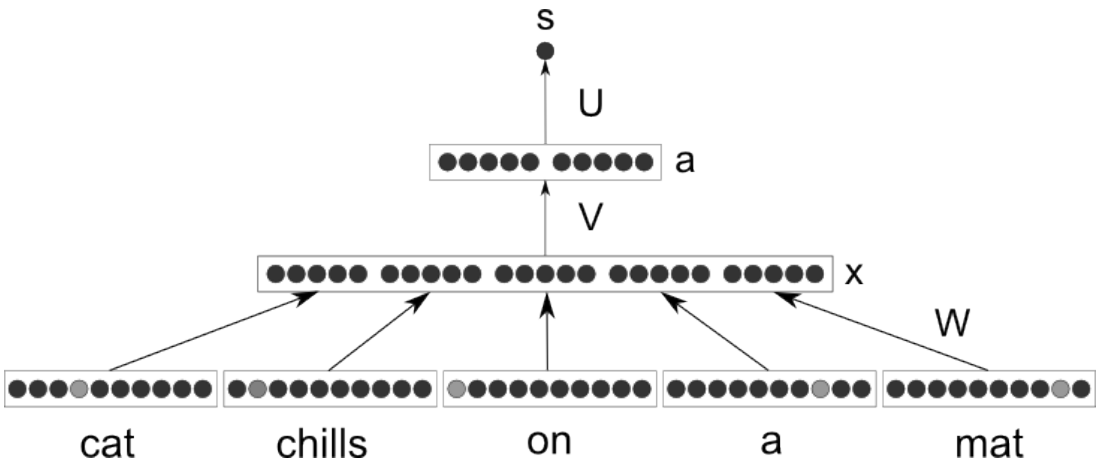


Figure 3.12.: Collobert et al. NNLM with one-hot vectors as input and a projection layer.

3.6.3.1. Continuous Skip-gram NNLM

This research work uses the Continuous Skip-gram (Skip-gram) Neural Network Language Model proposed by Mikolov et al. (2013b). The Fig. 3.13 describes conceptually the Skip-gram model. The intuition behind this model is that the Skip-gram architecture predicts the context of a word based on the word itself. Words that are semantically similar tend to be used in the same contexts, so this architecture will try to fit the word representations according to the context in which these words appear. This make semantically similar words grouped closer in the feature space. The non-linear hidden layer is removed in this model and the projection layer is shared for all words (not just the projection matrix); thus, all words get projected into the same position (their vectors are averaged). The neural network topology gives a better training performance along with a better word representation.

It was shown by Mikolov et al. (2013b) that the word vectors computed with this model captures many linguistic regularities. Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space. For example vector operations vector(Paris) - vector(France) + vector(Italy) results in a vector that is very close to vector(Rome), and vector(king)

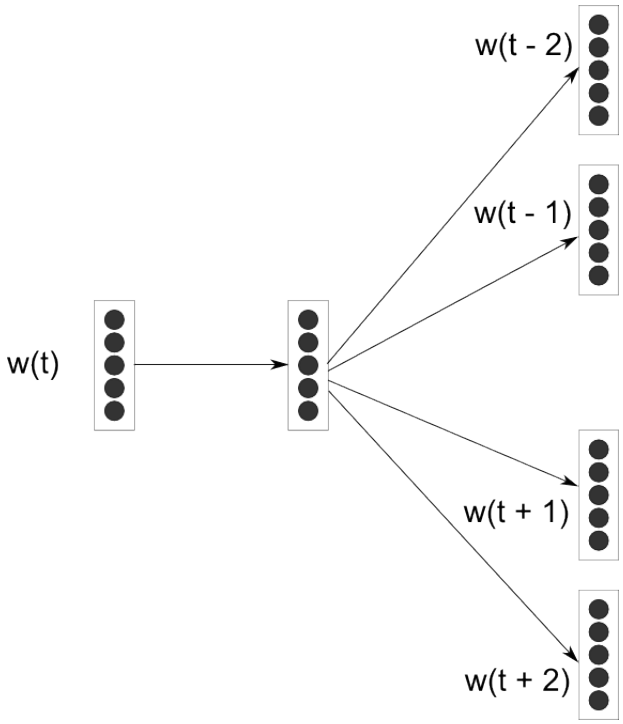


Figure 3.13.: The Skip-gram model.

- vector(man) + vector(woman) is close to vector(queen).

3.6.4. IAI to Implicit Aspects Mapping Method

The mapping between IAI and implicit aspects is as follows. The proposed approach consist firstly in computing a vector space of words using the Skip-gram model and some training data. The semantic relatedness between words can be measured with the euclidean distance between these vectors.

The equation 3.5 describes the objective function to map an IAI with its respective aspect. Let I denote the set of implicit aspect indicators present in a document. Let A denote the set of possible implicit aspects for any indicator $i \in I$. The map M(i) for a given indicator i is computed by finding the aspect a that is closest to it in the vector space according to the euclidean distance.

$$\mathrm{M}(i) = \arg\min_{a \in A} \mathrm{dist}(i, a) \tag{3.5}$$

The Algorithm 3.4 describes how to compute the objective function . The semantic similarity between an IAI and a set of implicit aspects is computed for every IAI in the document being evaluated. The algorithm uses the vector representation of the IAI and a particular aspect and computes the euclidean distance between those vectors. The aspect that is closer to the IAI is chosen as implicit aspect. The IAI corpus described in sec. 3.5.1 was used to perform the tests.

The data used to compute the vector space comes from the site Amazon.com, which is one of the largest e-commerce sites. One of its features is that it allows to its users to submit reviews of the different products offered. These reviews are publicly available and they are organized by product, by opinion polarity and other search criteria. The data amount available is quite large, and new reviews are added daily by thousands.

To gather the data needed to build the vector representations, an Amazon.com web-crawler was built. It collected the data of 250,000 reviews in the electronics commodities domain. The data was normalized by removing the punctuation and changing all the characters to lowercase.

The Word2Vec[3] toolkit is an efficient C++ implementation of the Skip-gram model for computing vector representations of words. There are several parameters that allow to configure how the vector space is computed. These parameters are the following:

- -train: This parameter specifies the training data file name. This file must be an UTF-8 or ASCII encoded text file.
- -output: This parameter specifies the file name where the vector representation will be saved.

[3]https://code.google.com/p/word2vec/

- -cbow: This parameter specifies if the CBOW NNLM is used. This is because this toolkit also implements the CBOW NNLM. It is out of the scope of this research work a deeper analysis of this model since the Skip-gram model is the only one used.

- -size: This parameter defines the dimensionality of the vectors. The size used was 200.

- -window: This parameter specifies the size of the context window used by the Skip-gram model during training. The window size value used was 5.

- -hs: This parameter specifies if the training algorithm will use Hierarchical Softmax (Morin and Bengio (2005)). This is a computationally efficient approximation of the full softmax. The Appendix B describes the softmax method in the context of Neural Networks. This option is used to compute the vector representations of words. It is out of the scope of this work a full description of hierarchical softmax.

- -ns: This parameter specifies if the algorithm will use Negative Sampling (Mikolov et al. (2013c)). This is a simplified version of the Noise Contrastive Estimation Method, which performs a similar function to hierarchical softmax. Negative sampling is the method used.

- -threads: This parameter specifies the number of threads used in training time. The default value of 12 was used.

- -binary: This parameter defines the output format as binary or as text. The binary option was used.

The Fig. 3.14 shows the command used to compute the vector space. There are other parameters that were not described since they are irrelevant or not used.

```
./word2vec -train reviews.txt -output vectors.bin -cbow 0 -size 200
    -window 5 -negative 1 -hs 0 -sample 1e-3 -threads 12 -binary 1
```

Figure 3.14.: Word2Vec command

3.6.5. Baselines

Two semantic relatedness techniques were used as baselines: *lesk* (Banerjee and Pedersen (2003)) and *vector* (Patwardhan et al. (2003)). The *wop* (Wu and Palmer (1994)) semantic similarity technique was also used as baseline. All these methods are based in WordNet. The *wup* method finds the path length to the root node in the WordNet ontology from the least common subsumer (LCS) of two concepts to be evaluated, which is the most specific concept they share as an ancestor. This value is scaled by the sum of the path lengths from the individual concepts to the root.

Each concept (or word sense) in WordNet is defined by a short gloss. The *lesk* and *vector* measures use the text of that gloss as a unique representation for the underlying concept. The *lesk* measure assigns relatedness by finding and scoring overlaps between the glosses of the two concepts, as well as concepts that are directly linked to them according to WordNet. The *vector* measure creates a co–occurrence matrix from a corpus made up of the WordNet glosses. Each content word used in a WordNet gloss has an associated context vector. Each gloss is represented by a gloss vector that is the average of all the context vectors of the words found in the gloss. Relatedness between concepts is measured by finding the cosine between a pair of gloss vectors.

All the methods described give as output a number between 0 and 1. The more closer the output is to 1, the more related two concepts are.

The WordNet::Similarity[4] toolkit was used. This toolkit includes implementations of the *lesk* and *vector* methods. We also used the Natural Language Toolkit or NLTK[5], which has an implementation of the *wup* method.

The baselines were implemented in a similar way that our approach described in the Algorithm 3.4, as shown by the Algorithm 3.5. The $sim(\cdot)$ function represents one of the 3 methods proposed as baselines.

The performance analysis of this implicit aspect extraction approach is given in the sec. 4.5.

3.7. Opinion Summarization

As explained in sec. 1.9, this research work uses two approaches for opinion summarization:

- A structured summary. This summary will show every opinion quintuple described by the expression 1.1 in a graphical presentation, such as a table or a bar chart. The opinion holder and time of the opinion are consider irrelevant.
- A text summary. This summary is composed of parts of the original opinion texts. The method will produce this summary under different constraints in order to obtain the best summary according to a specific performance metric.

This section explains the methods used for both opinion summarization approaches.

3.7.1. Structured Opinion Summary

The structured opinion summary method is straightforward once each opinion is retrieved as a quintuple described by the expression 1.1. The aspect extraction and

[4] wn-similarity.sourceforge.net/
[5] http://www.nltk.org/

sentiment classification methods described in sec. 3.2, sec. 3.3 and sec. 3.4 computes the different elements of each opinion quintuple in a document. Then each opinion can be classified according to its polarity or the entity and its aspects. Finally this classification can be displayed in a human-legible way. A table or a graphic is the common approach to display the information.

As part of this research work, a structured opinion summary generator tool was developed. It consist of a Python program that receives the opinions as quintuples. This program generates a report displaying the results using graphics.

The Fig. 3.15 shows an example of the bar charts generated with this tool. This is the structured summary of a set of reviews of a DVD player. These reviews were taken from the corpus introduced by Hu and Liu (2004). The corpus has labeled each aspect found in every review, along with the aspect polarity. Let A be the set of all the aspects in the reviews. Each element in the bar chart x axis is an aspect $a \in A$. The y axis represent the number of positive and negative opinions for an aspect a_i. There are 2 bars for each aspect. One shows the amount of positive opinions while the other shows the negative ones.

3.7.2. Textual Opinion Summary

The main idea that describes the text opinion summary approach proposed is to extract specific contexts (such as a sentence or a paragraph) from a review or a set of reviews that gives the best summary according to some evaluation criteria. E.g. a specific method would extract the key sentences from a set of reviews that best summarize the information in such reviews.

The approach proposed in this research work makes three assumptions regarding the quality of an textual opinion summary. First, the quality of an opinion summary increases as the number of entities and aspects covered by this summary does. This is assumed since an opinion summary has to contain all the information related to the opinion targets in a concise way.

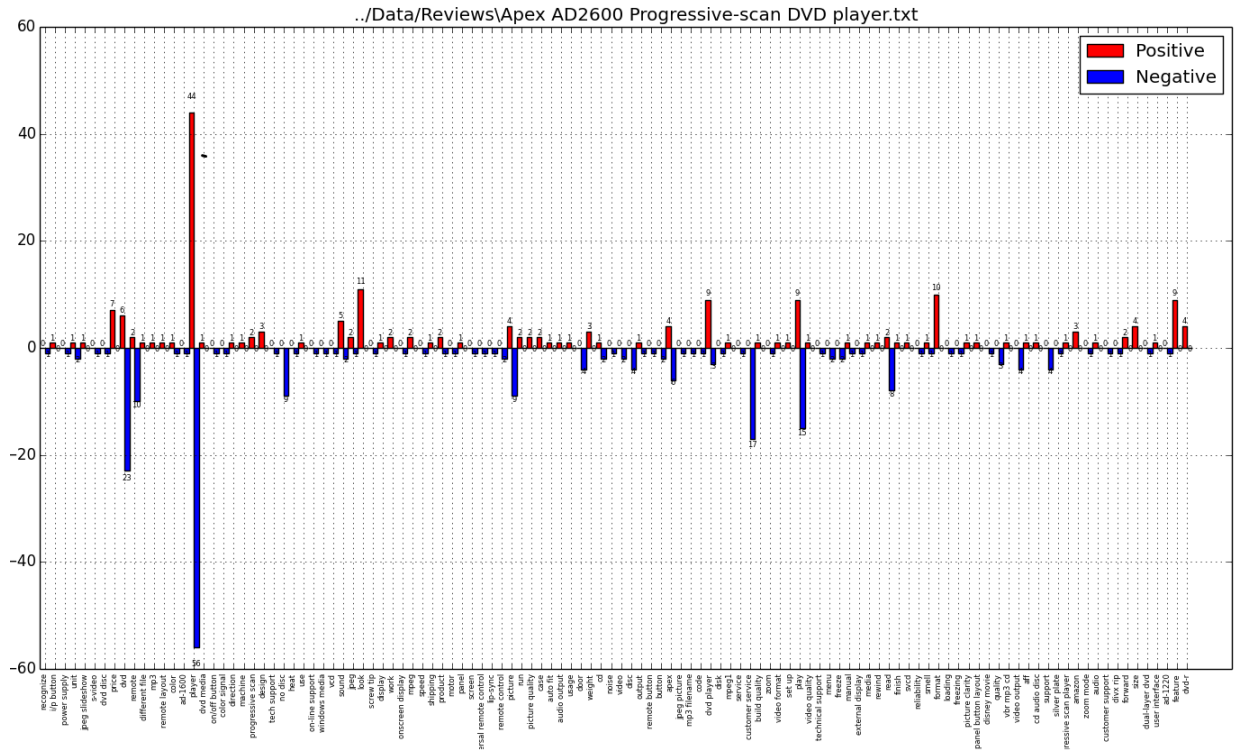


Figure 3.15.: The bar chart displayed by the opinion summarization tool

The second assumption made is that a good opinion summary has to be informative. The quality of an opinion summary increases as the informativeness of such summary does. It is difficult to measure the informativeness of a text, since this concept is ambiguous and subjective. The proposed approach uses the concept of entropy used in information theory as the measure of information content associated to the outcome of data.

The last assumption is that the size of the summary has to be much smaller than the size of the original document.

This research work uses an optimization approach to generate a text opinion summary. The proposed method is as follows. A review (or set of reviews) of an specified entity (e.g. a set of reviews of a camera) can be seen as the universe of aspects U. An specific context i is a subset of aspects $A_i \in U$. E.g. the i-th sentence in a camera review gives a positive opinion about the aspect *lens*, therefore

the set $s_i = \{$"lens"$\}$.

The proposed approach is to maximize the aspect coverage and the information content according to its entropy. This is done as follows. Let S be all the sentences in a document. Let $s_i \in S$ be a sentence i, which can be seen as a set of aspects. The objective is to extract a set of sentences $C \in S$ where the union of each element in C is U. Thus C is the summary of S. This can be modeled as the *Set Covering* problem. This problem is usually formulated with an additional constraint, which is the maximum number of sets required to cover all aspects. Therefore, the optimization approach for textual opinion summarization consist in extract the minimum amount of sentences that covers all the aspect within a review or a set of reviews. The Fig. 3.16 shows the graphical representation of the proposed approach. The first graphic represents the universe of aspects U and each ellipse inside of it represent a sentence and it is a subset of aspects. The shaded ellipses on the second graphic represent the sentences extracted as the summary. These sentences cover all the aspects present in U.

|

3.7.3. Set Covering problem as an ILP problem

To solve the weighted set covering problem, the proposed approach is to use Integer Linear Programing (ILP). This is a NP-complete problem and it is is composed of a linear objective function, subject to linear equality and linear inequality constraints. One of these constrains is that the decision variables involved must be integer, hence the name. The Appendix C gives a deeper explanation of linear programming and integer linear programming.

Both set covering and ILP are NP-complete, and it can be shown that one can be reduced to the other. Therefore a set covering problem can be solved as an ILP problem. The reduction is as follows. Let X_i be a random variable associated with each subset $s_i \in S$. $X_i = 1$ if s_i is a solution, and 0 otherwise. The ILP formulation is shown in the expression 3.6.

$$\begin{aligned} &\text{minimize } \sum c^{\mathrm{T}} X \text{ s.t.} \\ &\qquad \forall a \in U, \sum_{i:a \in s_i} X_i \geq 1 \\ &\qquad \forall X_i, X_i \in \{0,1\} \end{aligned} \tag{3.6}$$

The first constrain ensures that every aspect is present in at least one of the chosen subsets. The second constrain means that every subset is chosen or not. The objective function chooses a feasible function, if any.

E.g. there is a weighted set cover problem that needs to be solved. This problem involves 5 subsets. The set covering problem is reduced to an ILP problem and solved. The solution is given by a vector $X = \{0, 1, 0, 0, 1\}$. This vector indicates that the second and last element satisfies the ILP problem. Therefore the second and last subset satisfies the set covering for the original problem.

3.7.4. Sentences Costs

Note that the objective function in the expression 3.6 is multiplied by the vector c^T. This vector represent the weight, or *cost*, associated with each subset. Concretely, c is in function of the informativeness of an specific sentence.

The entropy defined in information theory is used as the measure of information within a sentence. The entropy H of a random variable X is described in the equation 3.7.

$$H(X) = E[I(X)] = E[-\ln(\mathrm{P}(\mathrm{X}))] \tag{3.7}$$

E is the expected value operator, and I is the information content of X. When taken from a finite sample, the entropy can explicitly be written as the equation 3.8.

$$H(X) = \sum_i P(x_i)I(x_i) = \sum_i P(x_i)\log_b P(x_i) \tag{3.8}$$

The Bag-of-Words (BOW) language model is used to compute the probability distribution of words needed to compute the entropy. In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. Each word in a document is modeled as a random variable where its probability depends entirely on itself.

The probability of an specific term t_i in a set of terms D is the number of occurrences of this term in the document divided by the total of terms in the document. More formally, let $T = \{t|t \in D, t = t_i\}$ be the set of terms t_i. The probability of t_i is described by the equation 3.9:

$$P(t_i) = \frac{|T|}{|D|} \tag{3.9}$$

The proposed method to calculate the cost of a sentence s_i is given by the equation 3.10:

$$c_{s_i} = \lambda \cdot \left(\frac{\sum_{t' \in D} H(t')}{\sum_{t \in s_i} H(t)^2} \right) + (1 - \lambda) \cdot \left(\frac{|U|}{|A_{s_i}|} \right) \tag{3.10}$$

where A_{s_i} is the set of aspects in the sentence s. The first term indicates that the sentences with higher entropy will have a lower cost. The second term gives lower cost to sentences that have a bigger number of aspects. If the second term is omitted, high entropy sentences with few aspects in them will have a low cost, and this is undesirable since the objective is to extract the minimum amount of sentences describing the aspects. Therefore, sentences describing several aspects must have a lower cost. The hyperparameter λ acts as a scale factor and as a balance factor between both terms.

3.7.5. Textual Opinion Summary Generation with ILP

The textual opinion summary is obtained by solving an ILP instance. There are several libraries and tools to solve ILP problems. The GNU Linear Programming Kit (GLPK)[6] is a software package intended for solving large-scale linear programming (LP), mixed integer programming (MIP), ILP and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library. The package is part of the GNU Project and is released under the GNU General Public License. Problems can be modeled in the language GNU MathProg and solved with standalone solver GLPSOL, which is included in the GLPK package.

The GLPSOL software is used to solve the ILP instances. This program can use as input a script written in the MathPro language. This program needs a model and an instance to solve. The model is the formulation of an ILP problem and it describes the data, the objective function, the constrains of the problem, computation and display settings, etc. This model can be saved in a text file.

The instance is the description of the data, i.e. the values of the cost vector, the constrains coefficients and the number of variables and constrains.

The Fig. 3.17 shows the code of the model created for the proposed text opinion summarization tool.

[6]https://www.gnu.org/software/glpk/

```
param NAspects;
param NSentences;
param L;
param AMatrix {i in 1..NAspects, j in 1..NSentences};
param costs { i in 1..NSentences};
var s{i in 1..NSentences} binary;
minimize obj: sum{i in 1..NSentences} costs[i] * s[i];
s.t.
bndConstr {k in 1..NAspects}:

    sum{i in 1..NSentences} AMatrix[k,i] * s[i] >= 1;

LConstr: sum{i in 1..NSentences} s[i] <= L;
solve;
display{i in 1..NSentences} s[i];
end;
```

Figure 3.17.: ILP model MathProg code

The Fig. 3.18 shows an example of the code of an ILP instance. The parameter NAspects indicates the number constrains, which are the number of aspects in the entire document. The parameter NSentences indicates the number of of variables in this ILP instance, which are the number of subsets or sentences in the original problem. The parameter L defines the constrain of maximum number of sentences. The parameter AMatrix represents the relation between aspects and sentences. Each row is a particular aspect while the columns are the sentences in the instance. There are 7 aspects and 11 sentences in the instance shown in the Fig. 3.18.

```
param NAspects := 7;
param NSentences := 11;
param L := 10;
param AMatrix:
1 2 3 4 5 6 7 8 9 10 11 :=
1 1 0 0 0 0 0 0 0 0 0
2 0 1 0 0 0 0 0 0 1 1 0
3 0 0 0 0 1 0 0 0 0 0 0
4 0 0 0 0 0 1 1 0 0 1 0
5 0 0 0 0 0 0 0 0 1 0 0
6 0 0 0 0 0 0 0 0 1 0 0
7 0 0 0 0 0 0 0 0 0 1 0 ;
param costs :=
1 1.0
2 1.0
3 1.0
4 1.0
5 1.0
6 1.0
7 1.0
8 1.0
9 1.0
10 1.0
11 1.0 ;
```

Figure 3.18.: ILP instance MathProg code

The Fig. 3.19 shows the command used to solve an ILP instance stored in the file "instance.data" with the model "ilp.model" using the GLPSOL solver. The parameter "–math" indicates that the data is written in the MathProg language.

```
glpsol --math --model ilp.model --data instance.data
```

Figure 3.19.: GLPSOL command

The textual opinion summarization tool was built using Python and the GLPSOL solver. This tool includes a Python script to compute the BOW language model from a corpus. The structure that contains the BOW language model is serialized in a file. The corpus created by Hu and Liu (2004) was used to compute this model and it is also used for testing.

The opinion summarization tool receives as input the text to summarize divided by sentences. Each sentence must have their respective aspects as part of the input. Then it formulates the ILP problem associated and solves it using GLPSOL. It reads the output and displays the sentences selected as the opinion summary.

3.7.6. Textual Opinion Summary Baselines

Three baselines were created in order to compare the performance of the proposed textual opinion summarization approach. All baselines uses as input a set of reviews. Each review is divided in sentences and each sentence has labeled its aspects.

The first baseline consist in extract L sentences at the beginning of the article. The second baseline extracts a sentence if one of its aspects is not in an extracted sentences. The third baseline models also the textual opinion summarization as an ILP problem. However the cost of each sentence is the same.

The first metric used to compare the performance is the aspect coverage defined by the equation 3.11.

$$\zeta = \frac{|\{a_i | a_i \in U, a_i \in S\}|}{|A|} \times 100 \tag{3.11}$$

The compression rate defined by the equation 3.12 is used to compare the size relation between the entire set of sentences S and the summary C (which cardinality is defined by the number of sentences in it).

$$\mathrm{CR} = \frac{|S|}{|C|} \tag{3.12}$$

Finally the entropy of the summary is measured, which is defined by the equation 3.13.

$$H(C) = \sum_{s \in C} \sum_{t \in s} H(t) \tag{3.13}$$

The performance analysis of both opinion summarization approaches is given in the sec. 4.6.

Algorithm 3.1 Opinion Orientation Algorithm

```
for all sentence s_i that contains a set of aspects do
    aspects = aspects contained in s_i
    for all aspect a_j in aspects do
        orientation = 0
        if feature a_j is in the 'but' clause then
            orientation = ButClauseRule()
        else
            remove 'but' clause from s_i if it exists
            for all unmarked opinion word ow in s_i do
                orientation+ = wordOrientation(ow, a_j, s_i)
            end for
        end if
        if orientation > 0 then
            a_j's orientation in s_i = 1
        else
            if orientation < 0 then
                a_j's orientation in s_i = −1
            else
                a_j's orientation in s_i = 0
            end if
        end if
        if a_j is an adjective then
            (a_j).orientation+ = a_j's orientation in s_i
        else
            let o_{i,j} is the nearest adjective word to a_j in s_i
            (a_j, o_{i,j}).orientation+ = a_j's orientation in s_i
        end if
    end for
end for
//Context dependent opinion words handling

for all aspects as a_j with orientation = 0 in sentence s_i do
    if a_j is an adjective then
        a_j's orientation in s_i = (a_j).orientation
    else// synonym and antonym rule should be applied too
        let o_{i,j} is the nearest opinion word to a_j, in s_i
        if (a_j, o_{i,j}) exists then
            a_j's orientation in s_i = (a_j, o_{i,j}).orientation
        end if
    end if
    if a_j's orientation in s_i = 0 then
        a_j's orientation in s_i = InterSentenceConjunctionRule()
    end if
end for
```

Algorithm 3.2 Word Orientation Algorithm

```
if word is a Negation word then
    orientation = NegationRules()
    mark words in sentence used by Negation rules
else
    if word is a 'TOO' word then
        orientation = apply TOO Rules
        mark words in sentence used by TOO rules
    else
        orientation = orientation of word in opinionWord list
    end if
end if
orientation = orientation / dist(word, feature)
```

Algorithm 3.3 But Clause Rule

```
if aspect a_i appears in the but clause then
    for all unmarked opinion word ow in the but clause of sentence s_i do
        orientation+ = WordOrientation(ow, a_j, s_i)
    end for
    if orientation ≠ 0 then
        return orientation
    else
        orientation = orientation of the clause before "but"
        if orientation ≠ 0 then
            return -1
        else
            return 0
        end if
    end if
end if
```

Algorithm 3.4 IAI to implicit aspect mapping.

A = The set of all possible aspects that an IAI can suggest.
I = The set Implicit Aspect Indicators in the document
L = The lookup table with the vector representation of words.
M = Map hash table.
for all $i \in I$ **do**
  $MinDist = \infty$
  for all $a \in A$ **do**
    $d = \text{dist}(L_i, L_a))$
    if $d \geq MinDist$ **then**
      $MinDist := d$
      $M(i, a) := 1$
    end if
  end for
end for

Algorithm 3.5 Baselines algorithm for implict aspect extraction performance evaluation.

M = Map hash table.
for all $i \in I$ **do**
  $MaxSim = 0$
  for all $a \in A$ **do**
    $s = \text{sim}(i, a)$
    if $s \leq MaxSim$ **then**
      $max\_a := a$
      $MaxSim := s$
    end if
  end for
  $M(i, max\_a) := 1$
end for

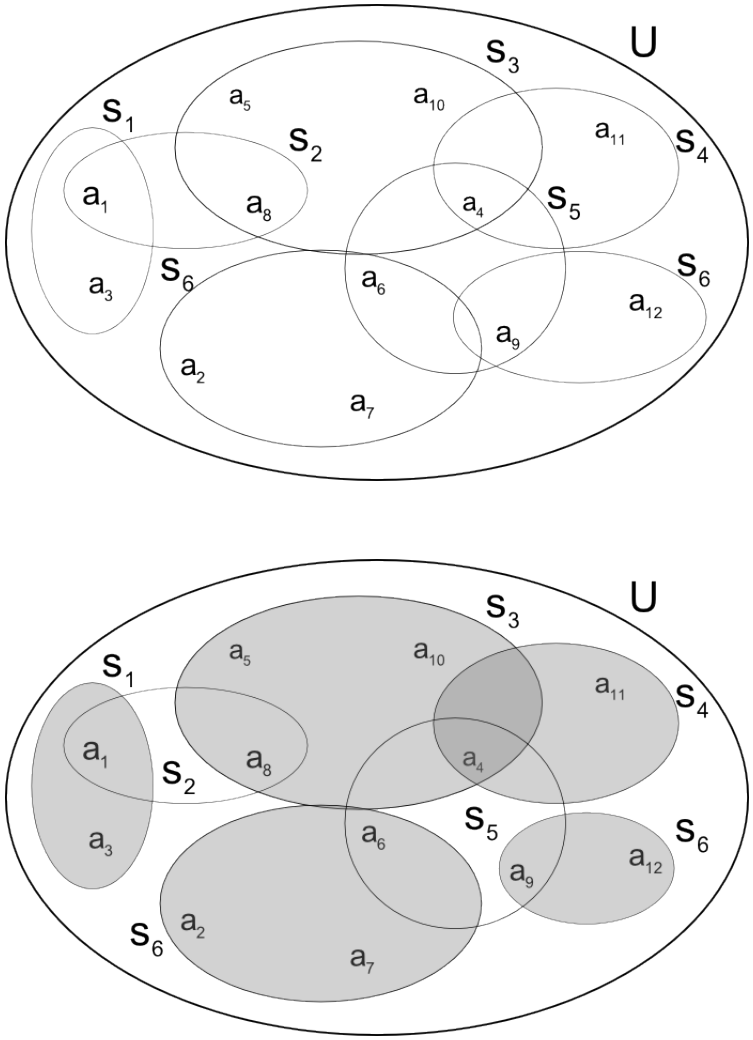


Figure 3.16.: The graphical representation of the textual summarization approach.

4. Results

4.1. Overview

This chapter shows the results obtained with the different experiments described in chapter 3.

4.2. Polarity Classification

The Tab. 4.1 shows the polarity classifier performance. It was built according to the description in sec. 3.2. The classifier implemented (CLSFR in Tab. 4.1) is compared with the results reported by Ding and Yu (2008) (DLY in Tab. 4.1). The results used were the reported for the Opinion Observer System without context dependency handling since this was the method implemented.

Table 4.1.: Polarity classification performance

Classifier	Precision	Recall	F1-score
CLSFR	0.866	**0.861**	0.863
DLY	**0.920**	0.830	**0.870**

The differences between both implementations are attributed to different tools used for POS tagging and lemmatization. This implementation is considered good enough since the difference in terms of the F1 score is minimal.

4.3. Explicit Aspect Extraction

The Tab. 4.2 shows the performance of the explicit aspect extraction tool. The first ten rows correspond to the results of each fold in the 10-fold cross validation scheme. Last row is the average performance.

Table 4.2.: Explicit aspect extraction performance

Fold	Precision	Recall	F1-score
1	0.6667	0.4865	0.5625
2	0.6857	0.5565	0.6144
3	0.6609	0.6586	0.6598
4	0.7189	0.5908	0.6486
5	0.7793	0.6384	0.7018
6	0.5714	0.7174	0.6361
7	0.6638	0.6290	0.6460
8	0.7283	0.6050	0.6610
9	0.6877	0.6429	0.6645
10	0.7284	0.4621	0.5655
Average	**0.6891**	**0.5987**	**0.6497**

The results reported by Huang et al. (2012) were used to compare the performance. This method also uses CRF to extract explicit aspects. This method uses syntactic dependency distributional context extraction features. The shows the results of both approaches. The row named EAE is method proposed in this work. The row named HPLN corresponds to the result reported by Huang et al. (2012).

The performance of the proposed approach is quite poor in comparison with the HPLN method. However, HPLN relies on heavily crafted features. Moreover, the syntactic dependency tree of each sentence is needed to build these features, which is computationally expensive. The method proposed in this work uses simple features that does not need a pre-processing stage, while it has a fairly good performance. Therefore, the proposed approach is consider good enough.

Table 4.3.: Explicit aspect extraction performance comparation.

Classifier	Precision	Recall	F1-score
HPLN	**0.7500**	**0.675**	**0.711**
EAE	0.6891	0.5987	0.6497

4.4. Implicit Aspect Indicators Extraction

The Tab. 4.4 compares the performance of the baselines described in sec. 3.5.4 and the proposed CRF-based approach with different features combinations. The *WT* features are the combination of the word and tag features (points 1 and 3 from the features description in sec. 3.5.2). *CNG* features are the character n-grams features (point 2). *CNTX* are context features and word bigram features (points 4 and 6). *CLS* are the class sequence features (point 5).

One can observe that the WT features give the greatest precision. However the recall is poor.

The CNG features give a recall boost. These features capture the morphological properties of the words (roots, prefixes, suffixes). Words with similar morphological properties tend to be semantically similar. For example the sentence "This phone looks great" could be rephrased as "The phone's look is great" or even "This phone looked great with its case." The root "look" present in the previous sentences is the best IAI to infer the *appearance* aspect. Therefore words with this root should have greater probability of being extracted as IAI. The drawback is that these features decrement the overall precision because more words that are not IAI but contain these character n-grams will have greater probability of being extracted.

The CNTX and CLS features improve both precision and recall. The best performance is obtained with the combination of WT, CNG, CNTX and CLS features.

Table 4.4.: IAI Extraction performance with different features

	Precision	Recall	F1 Score
BSLN1	0.0381	**0.3158**	0.0681
BSLN2	0.1016	0.1379	0.1170
BSLN3	0.5307	0.1439	0.2264
WT	**0.6271**	0.0575	0.1053
CNG	0.4765	0.1925	0.2742
CNTX	0.5030	0.1148	0.1869
WT,CNG	0.4697	0.1992	0.2795
WT,CNG,CNTX	0.5209	0.2031	0.2932
WT,CNG,CNTX,CLS	0.5458	0.2064	**0.2970**

The approach proposed in this work is better for this task. It outperforms significantly BSLN1, BSLN2 and BSLN3. The precision is very high. However the recall is lower than BSLN1.

In order to tradeoff precision for recall in the proposed CRF-based approach, a biased CRF classifier (Minkov et al. (2006)) was used . This method allows to set a bias towards the different classes. These biases (which internally are treated as feature weights in the log-linear model underpinning the CRF classifier) can take any real value. As the bias of a class A tends to plus infinity, the classifier will only predict A labels, and as it tends towards minus infinity, it will never predict A labels. These biases are used to manually adjust the precision-recall tradeoff.

Several experiments with the *IAI* class bias were performed in order to get the best precision-recall tradeoff. The value of this bias was changed within a range of 1.5 to 3.5. The *Other* class bias value was fixed to 1. The set of features used are those shown in the last row of the Tab. 4.4. The Tab. 4.5 shows the precision, recall and F1 Score of several experiments with different IAI class bias values. The Fig. 4.1 shows a graphic of this data.

The best F1 Score is obtained with an IAI class bias value of 2.5. It gives a boost of 28.61% in terms of the IAI extraction performance without IAI class bias.

Table 4.5.: Precision, Recall and F1 Score with different IAI Class bias

IAI Class Bias	Precision	Recall	F1 Score
1.5	**0.5252**	0.2602	0.3479
2.0	0.4636	0.3095	0.3711
2.5	0.4201	0.3503	**0.3820**
3.0	0.3656	0.3850	0.3750
3.5	0.3184	**0.4203**	0.3623

Furthermore both precision and recall are higher than any of the baselines.

4.5. Implicit Aspect Extraction

The corpus proposed and described in sec. 3.5.1 was used for evaluation. This corpus has 8 implicit aspects labelled: *functionality*, *appearance*, *quality*, *price*, *size*, *performance*, *weight* and *behavior*. The Tab. 4.6, Tab. 4.7 and Tab. 4.8 describes the performance of the baselines along with our approach for each one of these implicit aspects.

The corpus used for evaluation has 8 implicit aspects labelled: *functionality*, *appearance*, *quality*, *price*, *size*, *performance*, *weight* and *behavior*. The Tables Tab. 4.6, Tab. 4.7 and Tab. 4.8 describes the performance of the baselines along with our approach for each one of these implicit aspects.

We noted that all baselines cannot extract the implict aspects *functionality* and *behavior*, while our approach do a better job. However, the performance for the implicit aspect *behavior* is still poor. This implicit aspect is difficult to extract since the semantic relatedness between this implicit aspect and its IAI is subtle. An example would be the sentence “The dvd player runs silently”, where the implicit aspect is *behavior* and the IAI is the word “silently”.

Our approach has better precision for the implicit aspects *functionality*, *size*, *performance* and *behavior*. These implicit aspect are more ambiguous in contrast to the implicit aspects *appearance*, *price* and *weight*, which are more specific. As for

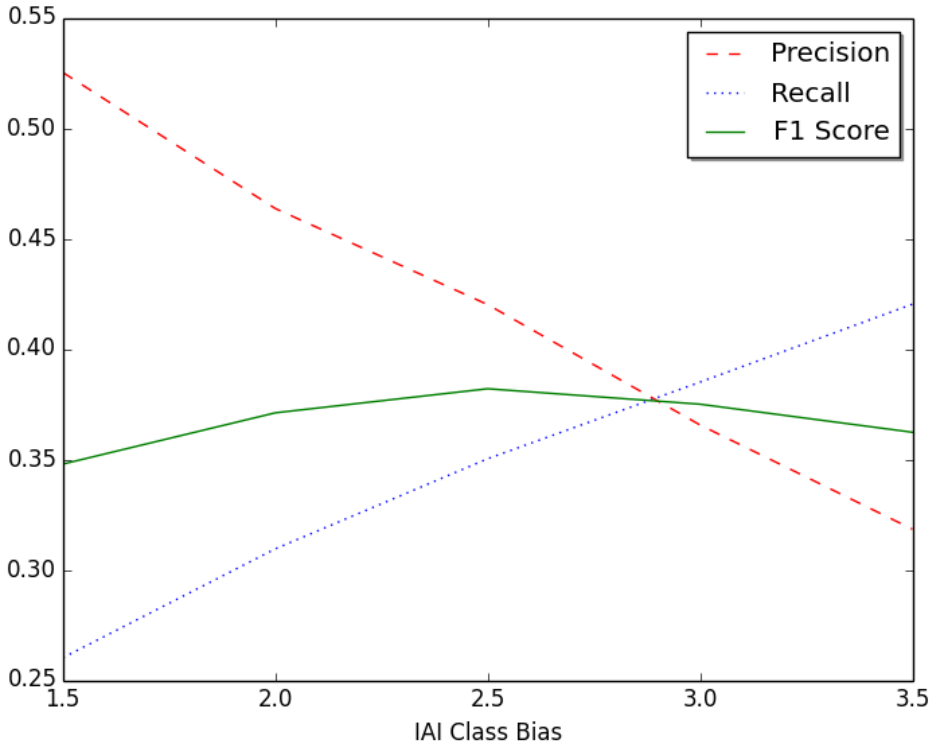


Figure 4.1.: Precision, Recall and F1 Score with Biased CRF

recall, our approach has better performance for the implicit aspects *functionality*, *appearance*, *price*, *size* and *behavior*. Finally, in terms of the F1-score, our approach is better for the implicit aspects *functionality*, *appearance*, *price*, *size*, *weight* and *behavior*.

Table 4.6.: Precision performance

	Functionality	Appearance	Quality	Price	Size	Performance	Weight	Behavior
Wop	0	1.0	0.115	0	0	0	0	0
Vector	0	0.807	0.178	0.785	0.333	0.222	0.583	0
Lesk	0	0.807	0.211	0.333	0.727	0.542	0.625	0
Skip-gram	0.638	0.398	0.038	0.687	0.783	0.571	0.315	0.01

Table 4.7.: Recall performance

	Functionality	Appearance	Quality	Price	Size	Performance	Weight	Behavior
Wop	0	0.344	0.265	0	0	0	0	0
Vector	0	0.362	0.797	0.297	0.037	0.016	0.233	0
Lesk	0	0.362	0.829	0.054	0.1	0.308	0.166	0
Skip-gram	0.303	0.788	0.01	0.970	0.587	0.100	0.966	0.250

Table 4.8.: F1-Score performance

	Functionality	Appearance	Quality	Price	Size	Performance	Weight	Behavior
Wop	0	0.512	0.161	0	0	0	0	0
Vector	0	0.5	0.292	0.431	0.067	0.030	0.333	0
Lesk	0	0.5	0.337	0.09	0.175	0.393	0.263	0
Skip-gram	0.411	0.529	0.01	0.804	0.671	0.169	0.475	0.028

The Tab. 4.9 shows the global performance. The proposed approach in this work outperforms the baselines. The precision and recall is better.

Table 4.9.: Global Performance

	Precision	Recall	F1-Score
Wop	0.135	0.078	0.099
Vector	0.243	0.208	0.224
Lesk	0.310	0.265	0.286
skip-gram	0.375	0.375	0.375

4.6. Opinion Summarization

4.6.1. Structured Opinion Summarization

The Fig. 4.2, Fig. 4.3, Fig. 4.4, Fig. 4.5 and Fig. 4.6 show the graphics generated by the structured opinion summarization tool developed in this research work.

The corpus introduced by Hu and Liu (2004) was used as input to generate such graphics. This tool uses as input the reviews divided by sentences. Each sentence must have the aspects that it contains specified in the input. And the polarity of each aspect must also be given. The corpus has labeled each aspect found in every review, along with the aspect polarity. Let A be the set of all the aspects in the reviews. Each element in the bar chart x axis is an aspect $a \in A$. The y axis represent the number of positive and negative opinions for an aspect a_i. There are 2 bars for each aspect. One shows the amount of positive opinions while the other shows the negative ones.

4.6.2. Textual Opinion Summarization.

The approach proposed in this research work is compared against 3 baselines. The first baseline consist in extract L sentences at the beginning of the article. The second baseline extracts a sentence if one of its aspects is not in an extracted sentences. The third baseline also models the textual opinion summarization as an ILP problem. However the cost of each sentence is the same. The corpus introduced by Hu and Liu (2004) was used to evaluate the performance.

The Tab. 4.10 compares the aspect coverage of the proposed approach and the different baselines. The proposed approach is labeled as TOS. The first, second and third baselines are labelled as BSLN1, BSLN2 and BSLN3 respectively.

Recall that TOS needs to compute a cost for each sentences in a set of reviews. This cost was defined in the equation 3.10. This equation specifies the λ hyperparameter and its purpose is to balance and scale the two terms in the equation 3.10. The λ value used to obtain the following results was 0.5.

One can observe that almost all approaches covers the aspects totally. Only BSLN1 fails to cover all aspects in the reviews.

The Tab. 4.11 shows the compression rate of each method. The TOS method significantly outperforms BSLN1 and BSLN2. However, the performance of BSLN3 is slightly better than TOS. This is natural since the BSLN3 selects the minimum

Table 4.10.: Aspect coverage percentage comparative analysis

	Apex	Canon	Nomad	Nikon	Nokia
TOS	100	100	100	100	100
BSLN1	67.24	40.95	36.70	50.66	36.93
BSLN2	100	100	100	100	100
BSLN3	100	100	100	100	100

number of sentences possible, while TOS considers the entropy of the resulting summary. A higher number of sentences would result in a higher entropy level.

Table 4.11.: Compression rate comparative analysis

	Apex	Canon	Nomad	Nikon	Nokia
TOS	8.13	7.10	11.07	5.86	6.57
BSLN1	1.96	3.43	4.58	2.68	3.54
BSLN2	7.11	5.91	9.58	5.08	5.35
BSLN3	8.13	7.19	11.14	5.86	6.65

The Tab. 4.12 compares the summary entropy of each method. One can observe that the summary produced by BSLN1 has the higher entropy and this is expected. It is possible to show that what gives higher entropy to a document are words that are not rare or not common, i.e. words that have a probability near to 0.5 in the language model used give higher entropy to a document. The summary produced by BSLN1 uses the 4 first sentences of a set of reviews to form a summary, whether these sentences have aspects or not. That being the case, it is more probably that this summary would have words of varying entropy levels. A summary containing sentences that include all the aspects have words that are regularly seen with these aspects (e.g. it is expected that sentences that are about the aspect *price* have words in common, such as “expensive”, “cheap”, “money”, etc). Therefore this summary is quite homogeneous, meaning that it talks about the same aspects and uses words common between these sentences to describe these aspects. Thus the words that form this summary are more common. Consequently, the summary will have a low entropy level.

As for the comparison between TOS and BSLN2, one can observe that there are mixed results depending on the document evaluated. TOS summary has a much higher entropy level for the Nikon reviews and a slightly higher entropy level for the Nomad reviews, while BSLN2 has a higher, though slightly, entropy level for the Apex, Canon and Nokia reviews.

Finally, TOS summaries have a higher entropy level in comparison to the summaries produced by BSLN3

Table 4.12.: Summary entropy comparative analysis

	Apex	Canon	Nomad	Nikon	Nokia
TOS	11.50	10.91	21.88	17.05	10.59
BSLN1	39.58	20.63	42.83	15.93	15.00
BSLN2	12.22	12.09	21.71	8.19	11.36
BSLN3	9.98	9.69	18.35	7.14	9.86

A qualitative analysis allows to assert that TOS outperforms all baselines. TOS has a better aspect coverage than BSLN1 by several orders of magnitude and a much higher compression rate.

Regarding BSLN2, TOS has a better compression rate. The entropy level of the summaries produced by TOS are comparable to the summaries produced by BSLN2.

As for the BSLN3, TOS summaries have a much higher entropy level. However, TOS compression rate is slightly lower than BSLN3.

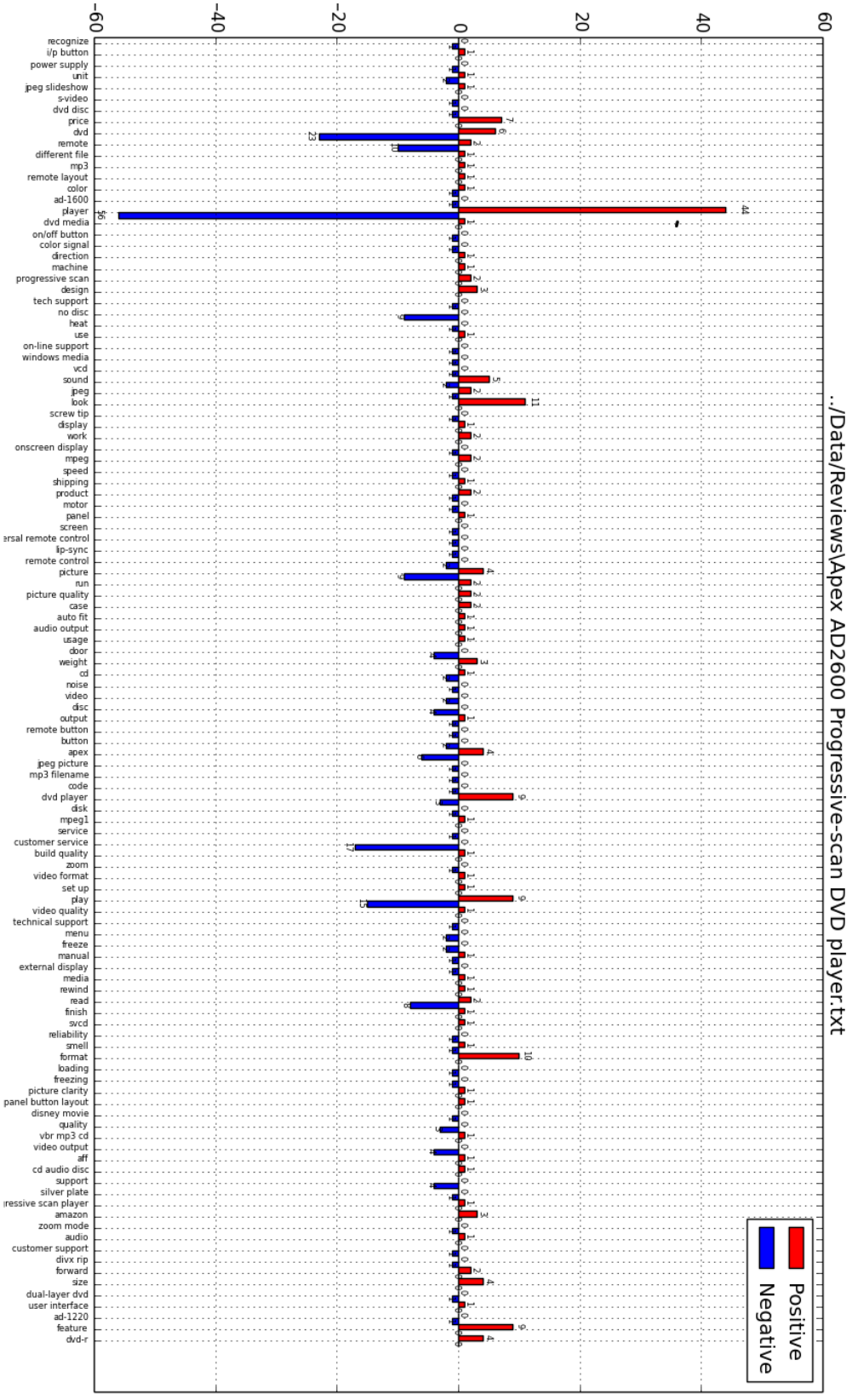


Figure 4.2.: Apex DVD Player structured summary

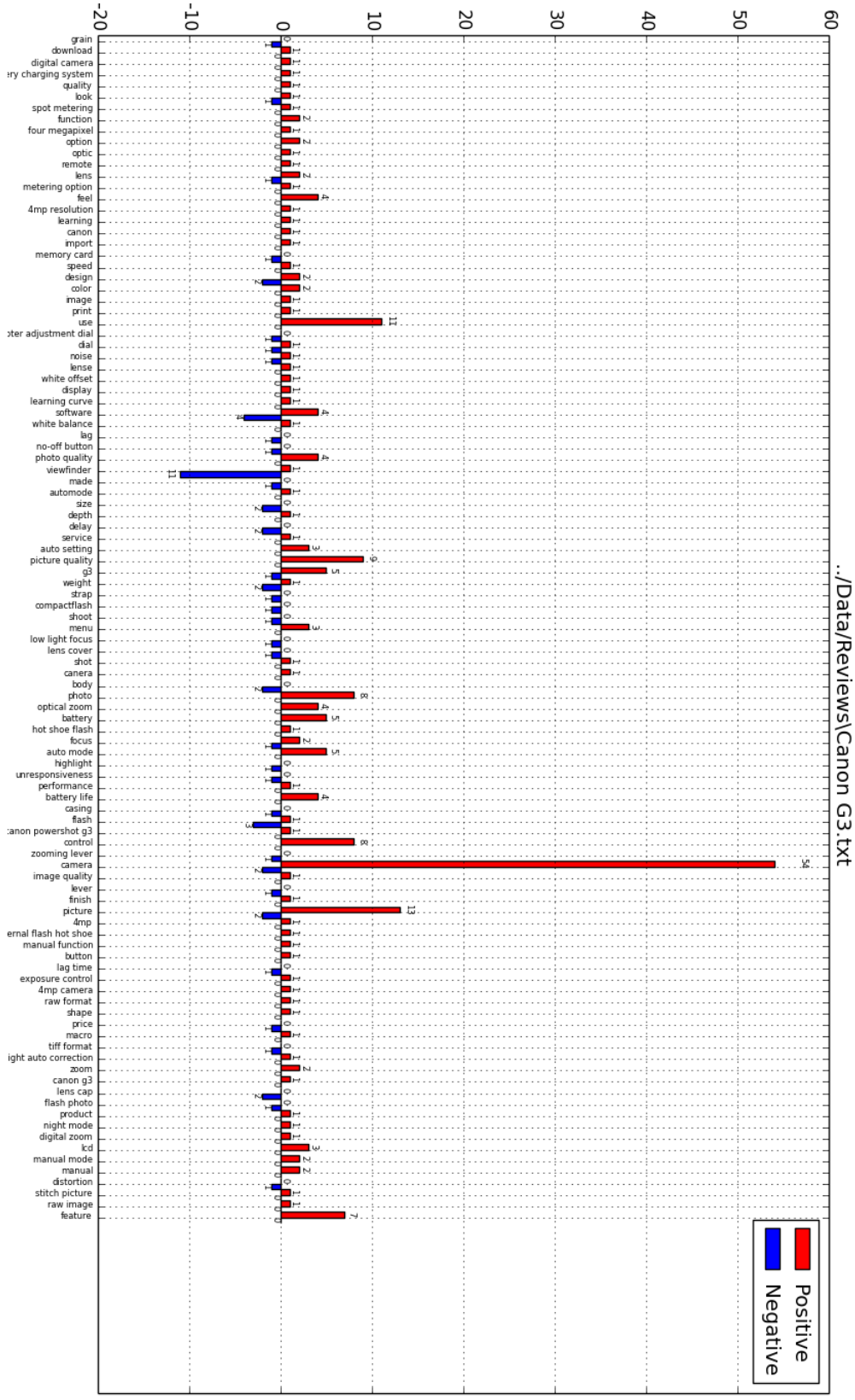


Figure 4.3.: Canon Camera structured summary

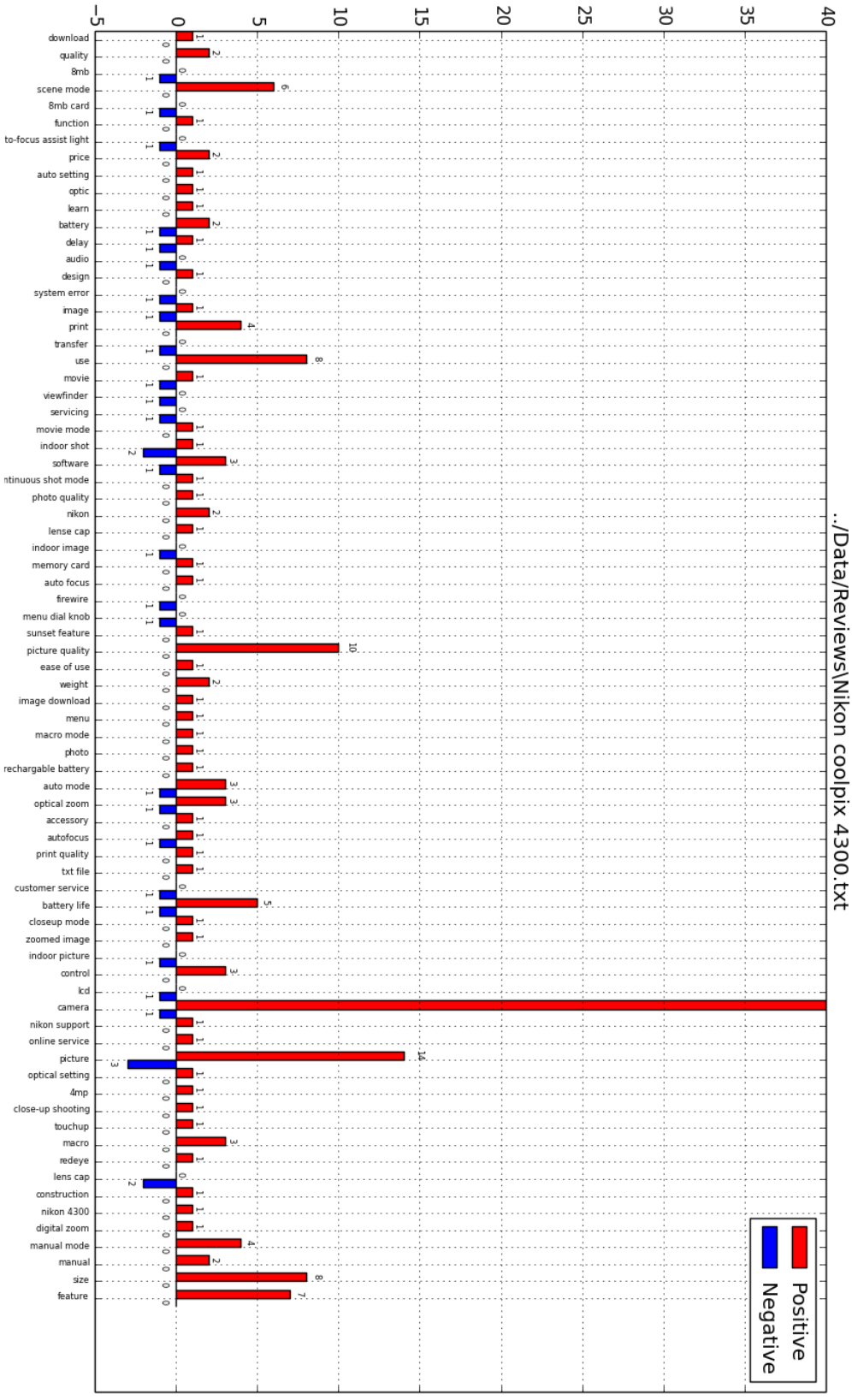


Figure 4.4.: Nikon Camera structured summary

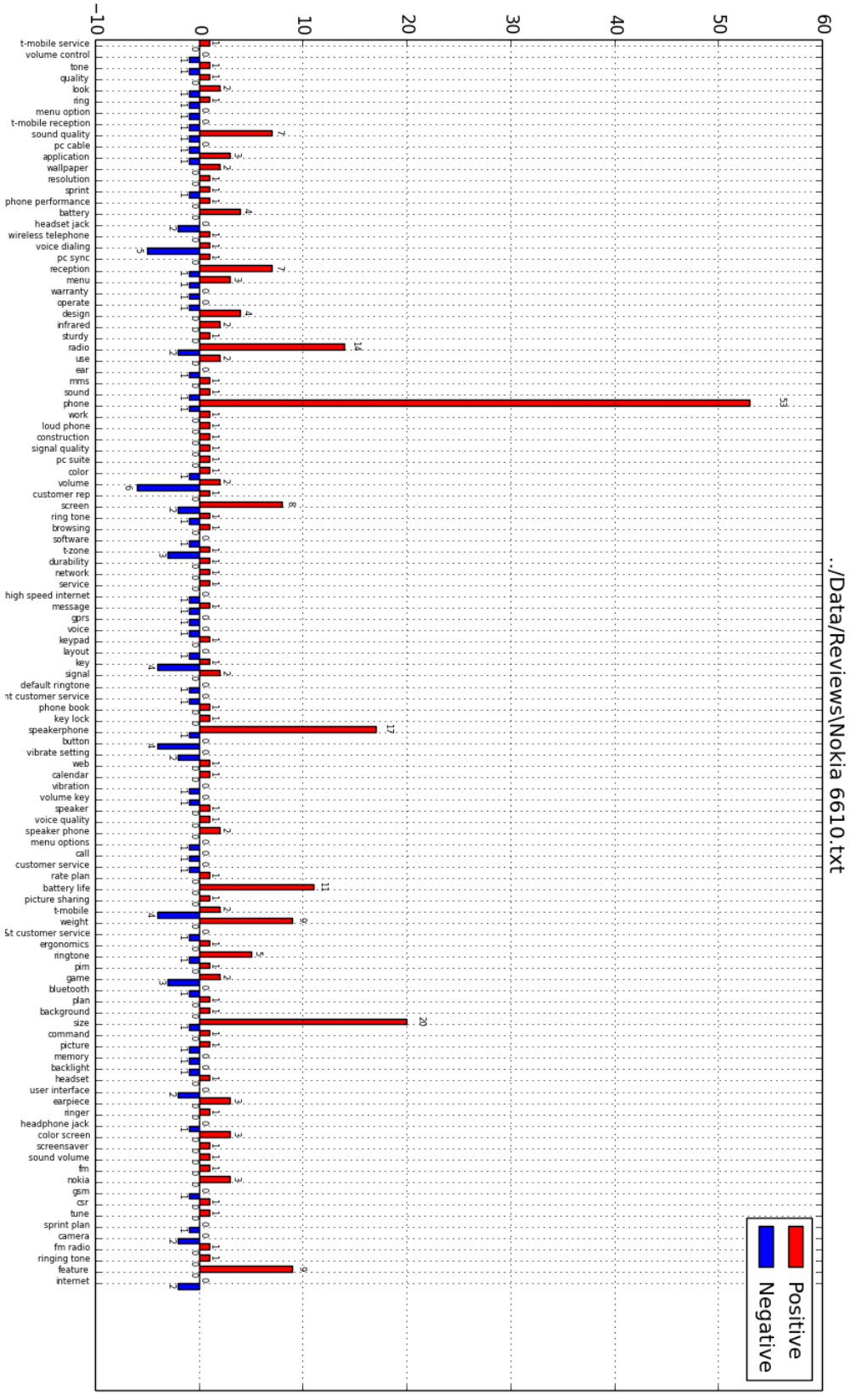


Figure 4.5.: Nokia Cellphone structured summary

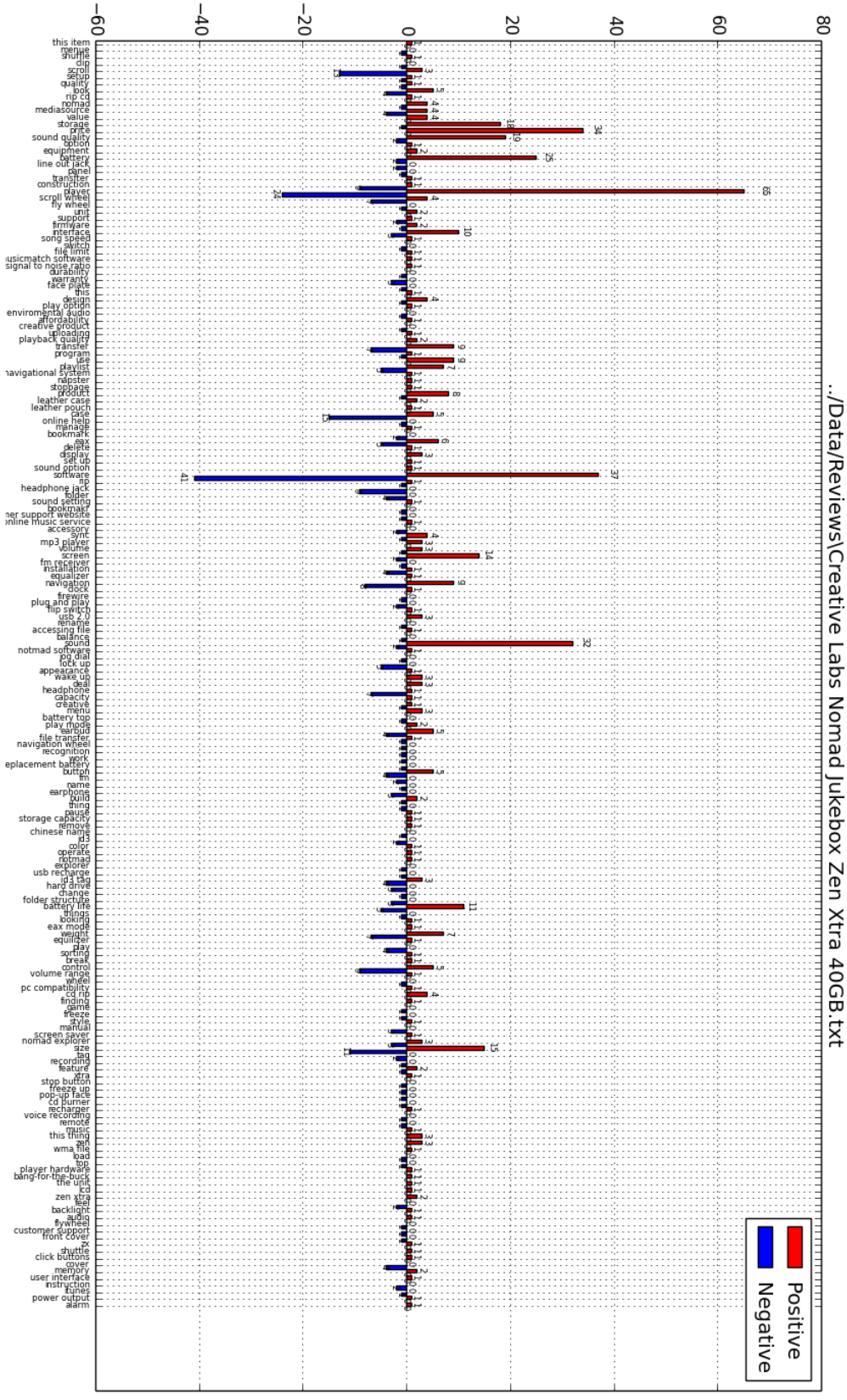


Figure 4.6.: Nomad MP3 player structured summary

Conclusions

This research work presented a novel framework for Polarity Opinion Summarization. It described the phenomena involved in this task along with the models proposed. It also described the methods used and their implementation.

This research work fulfill the objectives stated in sec. 1.2 since new methods for opinion summarization, and the tasks it involves, were created. Furthermore the state-of-the-art in opinion mining was expanded by introducing new and better methods for the tasks involved.

This work introduced Implicit Aspect Indicators (IAI) and a novel approach for IAI extraction. This approach proved to be the state-of-the-art method. It was also presented a comparative performance evaluation of the proposed approach with three baselines. Furthermore, and as part of this effort, the first corpus for IAI extraction and implicit aspect extraction was created.

Moreover, this work also introduced a novel implicit aspect method using distributed word representations. The semantic relatedness between implicit aspects and IAI was calculated using vector space of words, which was computed using a neural network language model. The semantic relatedness was used to extract implicit aspects using IAI. It was shown that this method outperforms the current approaches based on semantic relatedness by presenting a detailed performance analysis comparing this method and the baselines proposed.

Finally, this work introduced a novel framework for polarity opinion summarization. This framework consist in two opinion summary approches: structured opinion summarization and textual opinion summarization. The proposed textual

opinion summarization method tries to generate a summary that covers all the aspects present in the input. It also tries to maximize the informativeness of the summary (defined as the entropy of the data) and to minimize the size of the summary. It was shown that the presented textual opinion summarization approach outperforms the baselines.

A. Probabilistic Graphical Models

A.1. Overview

This appendix gives a brief description on Probabilistic Graphical Models. It is out of the scope of this work a complete description of this framework.

A.2. Introduction

Probabilistic Graphical Models is a mathematical framework for representation and inference in multivariate probabilistic distributions. The key concept is that a probability distribution over many variables can be represented as product of local functions or factors, and these functions depend on a small set of variables. This factors can make the inference of a multivariate probabilistic distribution tractable.

Factors are just functions with a given scope. A factor could be a function p that represents the probability distribution in the scope of two independent random variables X and Y, i.e $p(X, Y)$.

The chain rule for probabilistic distributions is an example on how a probabilistic distribution can be represented as a product of factor . This rule allows the calculation of any member of the joint distribution of a set of random variables using conditional probabilities. The equation A.1 gives an example of this chain rule applied to some probability distribution over the random variables A_1, A_2, A_3 and A_4.

$$p(A_1, A_2, A_3, A_4) = p(A_4|A_3, A_2, A_1) \cdot p(A_3|A_2, A_1) \cdot p(A_2|A_1) \cdot p(A_1) \quad \text{(A.1)}$$

The chain rule representation of a joint distribution can be drastically compressed knowing conditional dependencies between variables. If we consider that every random variable in equation A.1 is independent of other variables, then the chain rule would be represented as in equation A.2.

$$P(A_1, A_2, A_3, A_4) = P(A_4) \cdot P(A_3) \cdot P(A_2) \cdot P(A_1) \quad \text{(A.2)}$$

The graphical part of this frameworks comes from the possibility of representing the conditional dependencies of a probability distribution in a graph. As an example, the Fig. A.1 represents the probabilistic graphical model for a Naive Bayes model. In such model it is assumed that every input variable is conditionally independent between each other. Each node is a random variable in the probability distribution. Each directed edge represents the conditional dependency between variables. In this case, the probability of x_1, x_2, x_3 depend on a class y and there are not conditional dependencies between input variables (the naive assumption of this model, hence the name).

This type of graphs representing conditional interdependencies are called *Bayesian Networks.* They are represented with directed graphs.

There are other type of probabilistic graphical models. These are called *Markov Networks* and they are represented by an undirected graph. The nodes also represent the random variables of a probability distribution. The difference are the edges, which represent the *affinity* rate between variables. The concept of affinity is not a concept easy to explain since it depends on the phenomena modeled, but in simple terms affinity can be described as the degree of acceptance or agreement between variables. An intuitive example would be a word represented by a variable x and its POS Tag represented as a variable y. In a POS tagging task the affinity

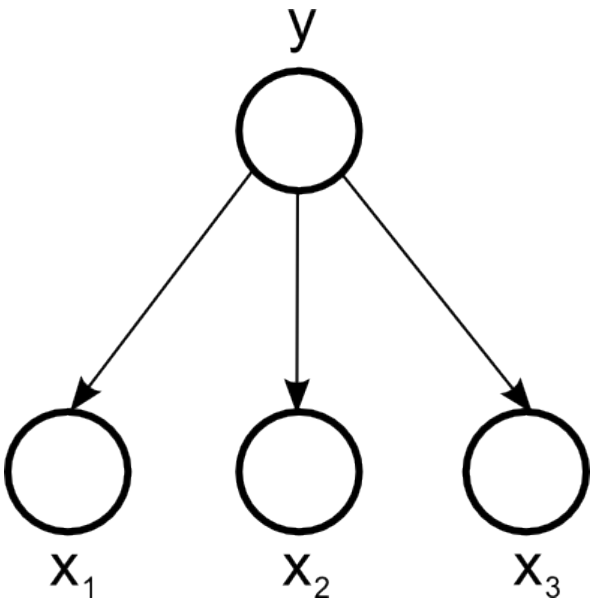


Figure A.1.: Probabilistic Graphical representation of a Naive Bayes Classification scheme

of x being the word "dog" and the label y being the tag "NOUN" is higher than x being "dog" and y being "VERB".

One could argue that the affinity concept is the same as probability. The main difference is that affinity allows to represent the phenomena in a more general way. The affinity could be computed as a number between 0 and 10 and it is not restricted as a probability distribution representation. This makes Markov Networks more flexible and powerful than Bayesian Networks. Conditional Random Fields is a type of Markov Network.

The formal definition for these Markov Networks is given in the equation A.3. It describes the conditional probability distribution as a product of n factors Ψ in the scope of an input vector $\vec{x}$ and a label vector $\vec{y}$. The $Z(\vec{x})$ function is a *normalization factor.* This is because the affinity between random variables could be represented in different ways (e.g. a integer number), and what it is wanted at the end is the conditional probability distribution.

$$p(\vec{y}|\vec{x}) = \left(\frac{1}{Z(\vec{x})} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y}) \right) \tag{A.3}$$

The equation A.4 describes the normalization factor. It consist on the summation of factor products for all the possible combinations of the class vector $\vec{y}$. The complexity to calculate this normalization factor is exponential but there are several techniques to compute this factor efficiently.

$$Z(\vec{x}) = \sum_{\vec{y'}} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y}) \tag{A.4}$$

A.3. Conditional Random Fields Definition

Conditional Random Fields, or CRF is a probabilistic graphical framework for building probabilistic models to segment and label sequences of data. It takes a discriminative approach. More generally, a CRF is a log-linear model that defines a probability distribution over sequences of data given a particular observation sequence. Lafferty et al. (2001) defined a CRF on a set of observations X and a set of labels sequences Y as follows: Let $G = (V, E)$ be a graph such that $Y = (Y_v)_{v \epsilon V}$ so that Y is indexed by the vertices of G, then (X, Y) is a conditional random field in case, when conditioned on X the random variables Y_v obey the Markov property with respect to the graph:

$$p(Y_v|X, Y_u, u \neq v) = p(Y_v|X, Y_u, u \sim v) \tag{A.5}$$

where $u \sim v$ means that w and v are neighbors in G. The joint distribution over the label sequences Y given X has the form:

$$p_\theta(y|x) \propto exp\left(\sum_{e \epsilon E,k} \lambda_k f_k(y|_e, x) + \sum_{v \epsilon V,k} \mu_k g_k(y|_v, x) \right) \tag{A.6}$$

where x is the data sequence, y is a label sequence, $y|_S$ is the set of components of y associated with the vertices in subgraph S and θ is the set weight parameters $\theta = (\lambda_1, \lambda_2, \lambda_3, ...; \mu_1, \mu_2, \mu_3, ...)$. We assume that the features f_k and g_k are given and fixed. They are usually Boolean and crafted by hand. For example, a vertex feature f_k might be true if the word X_i is upper case and the tag Y_i is a proper noun.

The parameter estimation problem is to determine the parameters θ from training data $D = (x^{(i)}, y^{(i)})_{i=1}^{N}$ with empirical distribution $\vec{p}(x, y)$. Usually gradient descent and its variations are used for this. This will be discussed further in sec. A.6.

A.4. Features Functions

The equation A.6 describes the conditional distribution in function of some feature functions. The intuitive explanation is that the data is represented as a series of questions. Each question is represented by these function features and they use the input value, the label of the class assigned, the value of the previous input, etc., to compute a score. If the data fits the model proposed, the score will be higher.

An example would be the application of CRF in Name Entity Recognition. This is a task in Natural Language Processing of identifying the entities like PERSON, LOCATION or COMPANY in a document. A Name Entity Extractor would label "Mister Smith" as PERSON in the phrase "There goes Mister Smith singing". The features that the extractor could use are shown in the Fig. A.2. For simplicity these functions only uses the current input value x. The function f_1 tries to model the fact that words like "Mr.", "Mister" or "Miss" usually refers to a PERSON, so the function will return the value of 1 indicating that there is strong evidence the word must be tagged as PERSON. The function f_2 models the fact that entities are always nouns. So if the Part of Speech Tag of the input sentence is a noun then the function returns 1, so the probability of y being PERSON for the input x is higher. The function f_3 models the fact that the first letter of an entity word is usually uppercase.

$$f_1(x) = \begin{cases} 1 & if\, x = Mister, Miss, Ms, Mr \\ 0 & Otherwise \end{cases}$$

$$f_2(x) = \begin{cases} 1 & if\, Tag(x) = NOUN \\ 0 & Otherwise \end{cases}$$

$$f_3(x) = \begin{cases} 1 & if\, Start(x)\, is\, uppercase \\ 0 & Otherwise \end{cases}$$

Figure A.2.: Feature Functions Example

The term $f(y|_e, x)$ of the equation A.6 is the general feature function representation. Its value will depend on the edges $y|_e$ for every input variable x and each feature function is weighted by the parameter λ_k. The term $\mu_k g_k(y|_v, x)$ also describes the feature functions but these are parametrized by vertices.

A.5. Linear Chain Conditional Random Fields

CRF is a general framework for discriminative modeling rather than a specific technique. A special CRF case for sequence label modeling is Linear Chain CRF. This CRF case is structured as a linear chain, and it models the output variables as a sequence. The Fig. A.3 shows the graphical representation of a Linear Chain CRF. This graph tries to model the relations present in a input x and a sequence of labels $y_t, ..., y_{t+k}$. In this case it is assumed that there is a relation between the input variable x and its label y_t. Moreover the graph models a relation between x and labels near to its own label and also it is assumed that there is a relation between labels.

The factor definition for a Linear Chain CRF is given in the equation A.7. The features are now in function of the input x and a k-sized label window y. The value j is the position of x in the sequence.

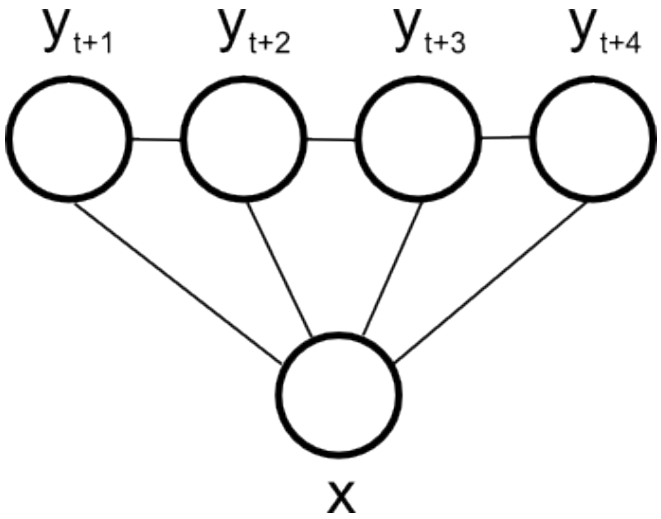


Figure A.3.: Linear Chain CRF

$$\Psi_j(\vec{x}, \vec{y}) = exp\left(\sum_{i=1}^{m} \lambda_k f_k(y_{j-k,} y_{j-k-1}, ..., y_j, \vec{x}, j)\right) \tag{A.7}$$

The equation A.8 is the formal definition of the Linear Chain CRF:

$$p_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}} \cdot exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_k f_k(y_{j-k,} y_{j-k-1}, ..., y_j, \vec{x}, j)\right) \tag{A.8}$$

The normalization factor is described in equation A.9:

$$Z_{\vec{\lambda}}(\vec{x}) = \sum_{\vec{y} \epsilon Y} exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_k f_k(y_{j-k,} y_{j-k-1}, ..., y_j, \vec{x}, j)\right) \tag{A.9}$$

The k parameter in the features functions indicates the label window size. E.g. with $k = 2$ the model just takes as input for sequence labeling the label associated to the current input and the label associated to the previous input. The equation A.10 shows this.

$$\lambda_k f_k(, y_{j-1}, y_j, \vec{x}, j) \tag{A.10}$$

By restricting the features to depend on only the current and previous labels, rather than arbitrary labels throughout the sentence, it is built a special case of a linear-chain CRF.

The Fig. A.4 shows some examples of 2-label-sized feature functions and how these functions try to model POS Tagging. For example, if the weight λ_1 associated with the feature f_1 is large and positive, then this feature models that words ending in -ly tend to be labeled as ADVERBS. Another example is if the weight λ_2 associated with feature f_2 is large and positive, then labels VERB assigned to the first word in a question (e.g "Is this a sentence beginning with a verb?") are preferred.

$$f_1(y_{j-1}, y_j, \vec{x}, j) = \begin{cases} 1 & if\, y_j = ADVERB;\, end(\vec{x}) = "ly" \\ 0 & Otherwise \end{cases}$$

$$f_2(y_{j-1}, y_j, \vec{x}, j) = \begin{cases} 1 & if\, j = 1;\, y_j = VERB;\, end(\vec{x}) = "?" \\ 0 & Otherwise \end{cases}$$

$$f_3(y_{j-1}, y_j, \vec{x}, j) = \begin{cases} 1 & if\, y_{j-1} = ADJETIVE;\, y_j = NOUN \\ 0 & Otherwise \end{cases}$$

$$f_3(y_{j-1}, y_j, \vec{x}, j) = \begin{cases} 1 & if\, y_{j-1} = PREPOSITION;\, y_j = PREPOSITION \\ 0 & Otherwise \end{cases}$$

Figure A.4.: Examples of feature functions for POS Tagging with $k = 2$.

A.6. Parameter Estimation in Linear Chain CRF

The method for estimation of the set of weights θ is as follows: Assuming there is a set of training examples (sentences and associated class labels):

- Randomly initialize the set of weights θ of our CRF model.
- To shift these randomly initialized weights to the correct ones, for each training example:
  - Go through each feature function f_i, and calculate the gradient of the log probability of the training example with respect to θ_i as shown in the equation A.11:

$$\frac{\partial}{\partial \theta_i} log(y|\vec{x}) = \sum_{j=1}^{m} f_k(, y_{j-1}, y_j, \vec{x}, j) - \sum_{y'} p(y'|\vec{x}) \sum_{j=1}^{m} f_k(, y_{j-1}, y_j, \vec{x}, j) \tag{A.11}$$

  - Move λ_i in the direction of the gradient:

$$\lambda_i = \lambda_i + \alpha \left[\sum_{j=1}^{m} f_k(, y_{j-1}, y_j, \vec{x}, j) - \sum_{y'} p(y'|\vec{x}) \sum_{j=1}^{m} f_k(, y_{j-1}, y_j, \vec{x}, j) \right] \tag{A.12}$$

    where α is some learning rate.
  - Repeat the previous steps until some stopping condition is reached (e.g the updates fall under some threshold).

B. Neural Networks

B.1. Overview

This appendix gives a brief description on the Neural Networks framework. It is out of the scope of this research work a full description of neural networks.

B.2. Supervised classification with Neural Networks

Consider a supervised learning problem where we have access to labeled training examples $(x^{(i)}, y^{(i)})$. Neural networks give a way of defining a complex, non-linear form of hypotheses $h_{W,b}(x)$, with parameters W, b that one can fit to the data.

The best example to start to describe neural networks is the simplest possible neural network, one which comprises a single "neuron." The Fig. B.1 shows a diagram that denote a single neuron.

This neuron is a computational unit that takes as input x_1, x_2, x_3 (and a $+1$ intercept term), and outputs $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$, where $f : \Re \mapsto \Re$ is called the activation function. In this research work, we will choose $f(\cdot)$ to be the sigmoid function:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{B.1}$$

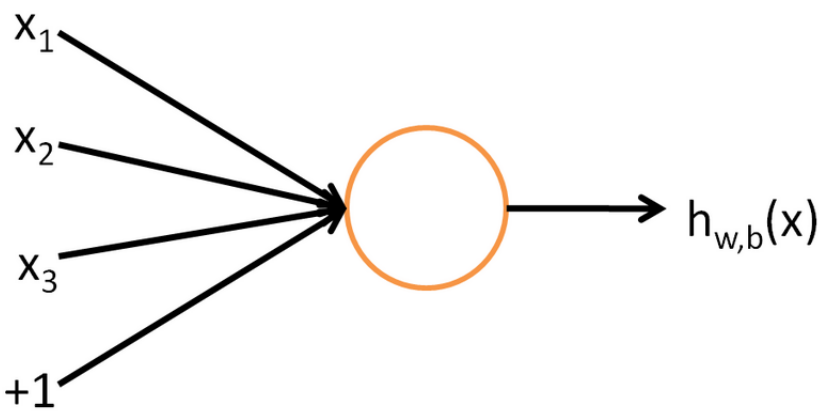


Figure B.1.: A single neuron

Thus, this single neuron corresponds exactly to the input-output mapping defined by logistic regression. Another common choice for f is the hyperbolic tangent, or *tanh*, function:

$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{B.2}$$

The plots of the sigmoid function and the *tanh* function are shown in the Fig. B.2.

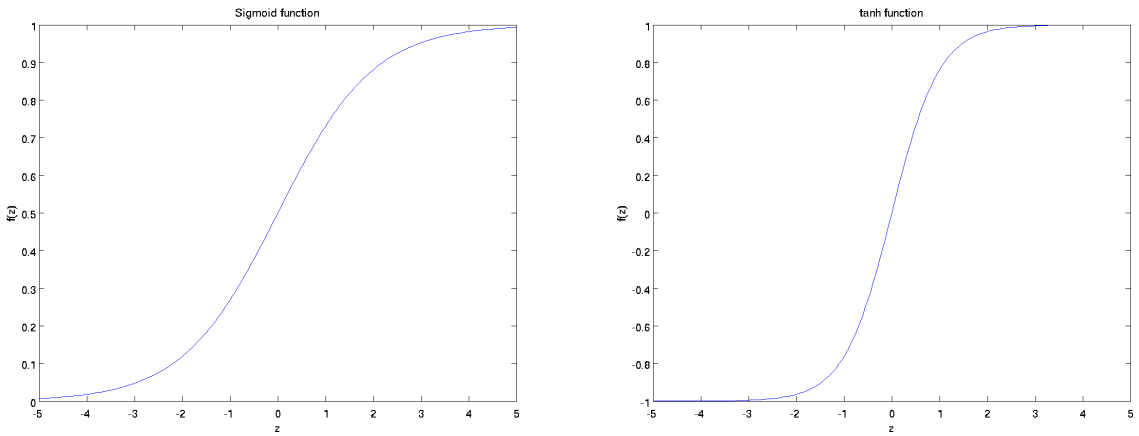


Figure B.2.: Sigmoid and tanh functions plots

The $tanh(z)$ function is a rescaled version of the sigmoid, and its output range is $[-1, 1]$ instead of $[0, 1]$. If $f(z) = 1/(1 + exp(-z))$ is the sigmoid function, then its derivative is given by $f'(z) = f(z)(1-f(z))$. (If f is the $tanh$ function, then its derivative is given by $f'(z) = 1-(f(z))^2$).

A neural network is put together by hooking together many simple neurons, so that the output of a neuron can be the input of another. The Fig. B.3 shows an example of a small neural network.

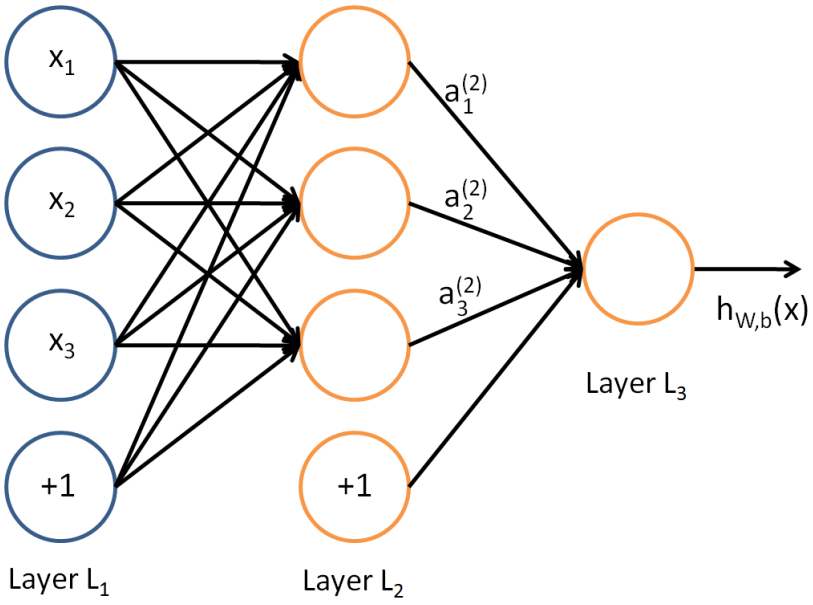


Figure B.3.: A small neural network

The inputs to the network are also denoted as circles. The circles labeled "+1" are called bias units, and correspond to the intercept term. The leftmost layer of the network is called the input layer, and the rightmost layer the output layer (which, in the Fig. B.3, has only one node). The middle layer of nodes is called the hidden layer, because its values are not observed in the training set. This neural network has 3 input units (not counting the bias unit), 3 hidden units, and 1 output unit.

Let n_l denote the number of layers in a network; thus $n_l = 3$ in the Fig. B.3. Each layer l is labelled as L_l, so layer L_1 is the input layer, and layer L_{n_l} the output layer. The neural network has parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, Let $W_{ij}^{(l)}$

be the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$. Also, $b_i^{(l)}$ is the bias associated with unit i in layer $l+1$. Thus, in the Fig. B.3, $W^{(1)} \in \Re^{3\times 3}$, and $W^{(2)} \in \Re^{1\times 3}$. Note that bias units don't have inputs or connections going into them, since they always output the value +1. Let s_l denote the number of nodes in layer l (not counting the bias unit).

B.3. Forward Propagation

Let $a_i^{(l)}$ to denote the activation (meaning output value) of unit i in layer l. For $l = 1$, let $a_i^{(1)} = x_i$ to denote the i-th input. Given a fixed setting of the parameters W, b, the neural network defines a hypothesis $h_{W,b}(x)$ that outputs a real number. Specifically, the computation that this neural network represents is given by:

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \tag{B.3}$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \tag{B.4}$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \tag{B.5}$$
$$h_{W,b}(x) = a_1^{(3)} = f(W_{11}^{(2)}x_1 + W_{12}^{(2)}x_2 + W_{13}^{(2)}x_3 + b_1^{(2)}) \tag{B.6}$$

The computation of $h_{W,b}(x)$ is called *forward propagation.*

B.4. Backpropagation

The *backpropagation* algorithm is used to learn the parameters W, b from a training set. Let $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ be a fixed training set of m training examples. A neural network can be trained using batch gradient descent. In detail, for a single training example (x, y), let the cost function with respect to that single example to be:

$$J(W, b; x, y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 \tag{B.7}$$

The equation B.7 is a one-half square error cost function. Given a training set of m examples, the overall cost function is:

$$\begin{aligned} J(W, b) &= \left[\frac{1}{m} \sum_{i=1}^{m} J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l - 1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} \left(W_{ji}^{(l)} \right)^2 \\ &= \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l - 1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} \left(W_{ji}^{(l)} \right)^2 \end{aligned} \tag{B.8}$$

The first term in the definition of $J(W, b)$ is an average sum-of-squares error term. The second term is a regularization term (also called a weight decay term) that tends to decrease the magnitude of the weights, and helps prevent overfitting. The weight decay parameter λ controls the relative importance of the two terms.

The objective is to minimize $J(W, b)$ as a function of W and b. To train a neural network, one must initialize each parameter $W_{ij}^{(l)}$ and each $b_i^{(l)}$ to a small random value near zero (say according to a $\Gamma(0, \epsilon^2)$ distribution for some small ϵ), and then apply an optimization algorithm such as batch gradient descent. Since $J(W, b)$ is a non-convex function, gradient descent is susceptible to local optima; however, in practice gradient descent usually works fairly well. It is important to initialize the parameters randomly, rather than to all 0's. If all the parameters start off at identical values, then all the hidden layer units will end up learning the same function of the input (more formally, $W_{ij}^{(1)}$ will be the same for all values of i, so that $a_1^{(2)} = a_2^{(2)} = a_3^{(2)} = \ldots$ for any input x). The random initialization serves the purpose of symmetry breaking.

One iteration of gradient descent updates the parameters W, b as follows:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{B.9}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{B.10}$$

where α is the learning rate. The backpropagation algorithm computes the partial derivatives shown in the equations B.9 and B.10. The intuition behind the backpropagation algorithm is as follows. Given a training example (x, y), we will first run a "forward pass" to compute all the activations throughout the network, including the output value of the hypothesis $h_{W,b}(x)$. Then, for each node i in layer l, we would like to compute an "error term" $\delta_i^{(l)}$ that measures how much that node was "responsible" for any errors in our output. For an output node, we can directly measure the difference between the network's activation and the true target value, and use that to define $\delta_i^{(n_l)}$ (where layer n_l is the output layer). For the hidden layer, we will compute $\delta_i^{(l)}$ based on a weighted average of the error terms of the nodes that uses $a_i^{(l)}$ as an input.

B.5. Softmax Regression in Neural Networks

If a neural network is used in a multiclass problem, then the softmax regression model is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables.

In the softmax regression setting, the goal is multiclass classification (as opposed to only binary classification), and so the label y can take on k different values, rather than only two. Thus, in the training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, we now have that $y^{(i)} \in \{1, 2, \ldots, k\}$. (Note the convention will be to index the classes starting from 1, rather than from 0.) The softmax function is defined in the equation B.11.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{B.11}$$

In softmax regression, the input to the function is the result of K distinct linear functions, and the predicted probability for the j class given a sample vector $x^{(i)}$ is given in the equation B.12.

$$p(y^{(i)} = j|x^{(i)}; \theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{k=1}^{K} e^{\theta_k^T x^{(i)}}} \tag{B.12}$$

Given a test input x, the objective of an hypothesis is to estimate the probability that $p(y = j|x)$ for each value of $j = 1, \ldots, k$. I.e., the objective is to estimate the probability of the class label taking on each of the k different possible values. Thus, the hypothesis will output a k dimensional vector (whose elements sum to 1) giving us our k estimated probabilities. Concretely, the equation B.13 describes the the hypothesis $h_\theta(x)$.

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)}; \theta) \\ p(y^{(i)} = 2|x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \tag{B.13}$$

The softmax regression cost function is defined in the equation B.14. The $1\{\cdot\}$ expression is the indicator function, so that $1\{\text{a true statement}\} = 1$, and $1\{\text{a false statement}\} = 0$. For example, $1\{2 + 2 = 4\}$ evaluates to 1; whereas $1\{2 + 2 = 5\}$ evaluates to 0.

$$\begin{aligned} J(\theta) &= -\frac{1}{m} \left[\sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \right] \\ &= -\frac{1}{m} \left[\sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} log(p(y^{(i)} = j|x^{(i)}; \theta)) \right] \end{aligned} \tag{B.14}$$

There is no known closed-form way to solve for the minimum of $J(\theta)$, and thus as usual one must resort to an iterative optimization algorithm such as gradient

descent or L-BFGS. The gradient of $J(\theta)$ is shown in the

$$\frac{\partial}{\partial \theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[x^{(i)} (1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta)) \right] \tag{B.15}$$

When a softmax regression model is integrated to a neural network, there is some changes in how this network is trained. Concretely, let l be the layer in a neural network that is constrained to the softmax function as activation function rather than the sigmoid or tanh function. Therefore the activation function gradients used by backpropagation change. Let k be the number of units in this layer. Since we have n activation functions parametrized by θ, to compute the gradients one must compute the Jacobian matrix. The equation B.16 describes the Jacobian matrix based on the probability of a class j described in equation B.15.

$$[J]_{ij} = p(y^{(i)} = i | x^{(i)}; \theta)(\delta_{ij} - p(y^{(i)} = j | x^{(i)}; \theta)) \tag{B.16}$$

where δ_{ij} is the Kronecker Delta (and not the term error δ from the backpropagation algorithm.) So the Jacobian matrix is a square matrix of size $n \times n$.

To compute the error term δ_l multiply the vector of error terms of the next layer by the Jacobian matrix as described in the equation B.17.

$$\delta_l = [J]\, \delta_{l+1} \tag{B.17}$$

If the softmax layer is the output layer, then the equation B.17 is the same as a regular neural network with the sigmoid or tanh function, as described in the equation B.18.

$$\delta_l = h_\theta(x^{(i)}) - y^{(i)} \tag{B.18}$$

C. Linear Programming and Integer Linear Programming

C.1. Overview

This appendix gives a brief description on Linear Programming (LP) and Integer Linear Programming (ILP). It is out of the scope of this research work to give a full description of LP and ILP.

C.2. Linear Programming

Linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. Its feasible region is a convex polyhedron, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued affine function defined on this polyhedron. A linear programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

Linear programs are problems that can be expressed in canonical form described by the expression C.1:

$$\begin{aligned} &\text{maximize } c^{\mathrm{T}}x \\ &\text{subject to } Ax \leq b \\ &\qquad \text{and } x \geq 0 \end{aligned} \tag{C.1}$$

where x represents the vector of variables to be determined, c is a vector of known coefficients usually associated to a *cost* of each variable, b is a known vector of coefficient associated to the *constraints* of the problem, A is a known matrix of coefficients and $(\cdot)^{\mathrm{T}}$ is the *matrix transpose*. The expression to be maximized or minimized is called the objective function. The inequalities $Ax \leq b$ and $x \geq 0$ are the constraints which specify a convex polytope over which the objective function is to be optimized. In this context, two vectors are comparable when they have the same dimensions. If every entry in the first is less-than or equal-to the corresponding entry in the second then we can say the first vector is less-than or equal-to the second vector.

The Fig. C.1 shows the feasible region for an LP problem of maximizing the objective function $z = x_1 + x_2$ with the constraints given by the inequalities $x_1 + 2x_2 \leq 4$, $4x_1 + 2x_2 \leq 12$, $-x_1 + x_2 \leq 1$, $x_1 \geq 0$ and $x_2 \geq 0$. The optimal value is given by the point $(8/3, 2/3)$, which is the intersection of the planes $x_1 + 2x_2 \leq 4$ and $4x_1 + 2x_2 \leq 12$.

The *standard form* is the usual and most intuitive form of describing a linear programming problem. It consists of the following three parts:

- A linear function to be maximized. The equation C.2 shows an example of this:

$$f(x_1, x_2) = c_1x_1 + c_2x_2 \tag{C.2}$$

- The problem constraints. The equations in the expression C.3 describe the

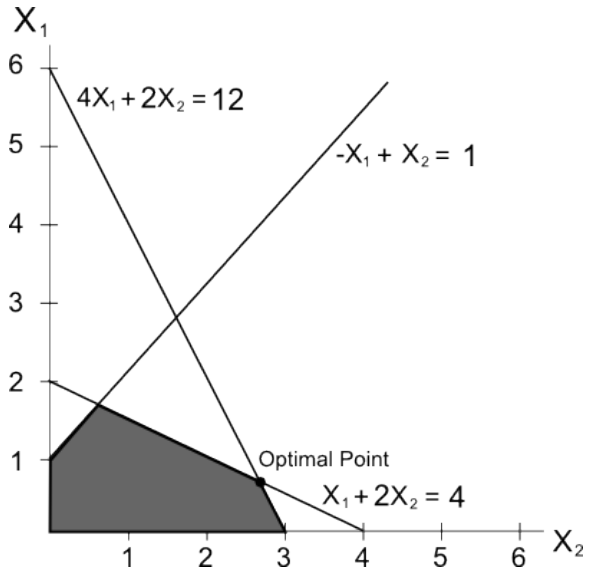


Figure C.1.: An LP feasible region

constrains of the problem defined in the equation C.2:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 &\leq b_2 \\ a_{31}x_1 + a_{32}x_2 &\leq b_3 \end{aligned} \tag{C.3}$$

- The non-negative variables constraints. The expression C.4 describe this:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned} \tag{C.4}$$

Other forms, such as minimization problems, problems with constraints on alternative forms, as well as problems involving negative variables can always be rewritten into an equivalent problem in standard form.

C.3. The Simplex Algorithm

The simplex algorithm is a method for linear programming. This method computes the optimal solution (if there is any) to an LP problem.

In geometric terms, the feasible region $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x_i} \geq \mathbf{0}$ is a (possibly unbounded) convex polytope. There is a simple characterization of the extreme points or vertices of this polytope, namely $x = (x_1, \ldots, x_n)$ is an extreme point if and only if the subset of column vectors A_i corresponding to the nonzero entries of $x(x_i \neq 0)$ are linearly independent. In this context such a point is known as a basic feasible solution (BFS).

It can be shown that for a linear program in standard form, if the objective function has a minimum value on the feasible region then it has this value on (at least) one of the extreme points. This in itself reduces the problem to a finite computation since there is a finite number of extreme points, but the number of extreme points is unmanageably large for all but the smallest linear programs.

It can also be shown that if an extreme point is not a minimum point of the objective function then there is an edge containing the point so that the objective function is strictly decreasing on the edge moving away from the point. If the edge is finite then the edge connects to another extreme point where the objective function has a smaller value, otherwise the objective function is unbounded below on the edge and the linear program has no solution. The simplex algorithm applies this insight by walking along edges of the polytope to extreme points with lower and lower objective values. This continues until the minimum value is reached or an unbounded edge is visited, concluding that the problem has no solution. The algorithm always terminates because the number of vertices in the polytope is finite; moreover since we jump between vertices always in the same direction (that of the objective function), we hope that the number of vertices visited will be small.

The solution of a linear program is accomplished in two steps. In the first step, known as Phase I, a starting extreme point is found. Depending on the nature

of the program this may be trivial, but in general it can be solved by applying the simplex algorithm to a modified version of the original program. The possible results of Phase I are either a basic feasible solution is found or that the feasible region is empty. In the latter case the linear program is called infeasible. In the second step, Phase II, the simplex algorithm is applied using the basic feasible solution found in Phase I as a starting point. The possible results from Phase II are either an optimum basic feasible solution or an infinite edge on which the objective function is unbounded below.

C.3.1. Standard Augmented Form

Linear programming problems must be converted into the standard augmented form before being solved by the simplex algorithm. This form introduces non-negative *slack variables* to replace inequalities with equalities in the constraints. This is done as follows:

- Let z be the objective function $c^{\mathrm{T}}x$. Therefore $z = c^{\mathrm{T}}x$. If the problem is $\min z$, convert it to $\max z$.
- If a constraint is $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i$ convert it into an equality constraint by adding a non-negative slack variable s_i. The resulting constraint is $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n + s_i = b_i$ where $s_i \geq 0$.
- If a constraint is $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq b_i$ convert it into an equality constraint by subtracting a non-negative *surplus* variable s_i. The resulting constraint is $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n - s_i = b_i$ where $s_i \geq 0$.
- If some variable x_j is unrestricted in sign, replace it everywhere in the formulation by $x_j^{'} - x_j^{''}$ where $x_j^{'} \geq 0$ and $x_j^{''} \geq 0$.

C.3.2. Simplex Tableaux

A linear program in standard form can be represented as a tableau of the form shown in the expression C.5:

$$\begin{bmatrix} 1 & -\mathbf{c}^T & 0 \\ 0 & \mathbf{A} & \mathbf{b} \end{bmatrix} \tag{C.5}$$

The first row defines the objective function and the remaining rows specify the constraints. If the columns of A can be rearranged so that it contains the identity matrix of order p (the number of rows in A) then the tableau is said to be in canonical form. The variables corresponding to the columns of the identity matrix are called basic variables while the remaining variables are called nonbasic or free variables. If the nonbasic variables are assumed to be 0, then the values of the basic variables are easily obtained as entries in b and this solution is a basic feasible solution.

Conversely, given a basic feasible solution, the columns corresponding to the nonzero variables can be expanded to a nonsingular matrix. If the corresponding tableau is multiplied by the inverse of this matrix then the result is a tableau in canonical form.

C.3.3. Pivot operations

The geometrical operation of moving from a basic feasible solution to an adjacent basic feasible solution is implemented as a pivot operation. First, a nonzero pivot element is selected in a nonbasic column. The row containing this element is multiplied by its reciprocal to change this element to 1, and then multiples of the row are added to the other rows to change the other entries in the column to 0. The result is that, if the pivot element is in row r, then the column becomes the r-th column of the identity matrix. The variable for this column is now a basic variable, replacing the variable which corresponded to the r-th column of the identity matrix before the operation. In effect, the variable corresponding to the pivot column enters the set of basic variables and is called the entering variable, and the variable being replaced leaves the set of basic variables and is called the leaving variable. The tableau is still in canonical form but with the set of basic variables changed by one element.

C.3.4. Algorithm

Let a linear program be given by a canonical tableau. The simplex algorithm proceeds by performing successive pivot operations which each give an improved basic feasible solution; the choice of pivot element at each step is largely determined by the requirement that this pivot improve the solution.

C.3.4.1. Entering variable selection

Since the entering variable will, in general, increase from 0 to a positive number, the value of the objective function will decrease if the derivative of the objective function with respect to this variable is negative. Equivalently, the value of the objective function is decreased if the pivot column is selected so that the corresponding entry in the objective row of the tableau is positive.

If there is more than one column so that the entry in the objective row is positive then the choice of which one to add to the set of basic variables is somewhat arbitrary and several entering variable choice rules have been developed.

If all the entries in the objective row are less than or equal to 0 then no choice of entering variable can be made and the solution is in fact optimal. It is easily seen to be optimal since the objective row now corresponds to an equation of the form shown in the equation C.6:

$$\mathrm{z}(x) = z_B + NB \tag{C.6}$$

where NB is the non-negative terms corresponding to non-basic variables.

C.3.4.2. Leaving variable selection

Once the pivot column has been selected, the choice of pivot row is largely determined by the requirement that the resulting solution be feasible. First, only

positive entries in the pivot column are considered since this guarantees that the value of the entering variable will be non-negative. If there are no positive entries in the pivot column then the entering variable can take any non-negative value with the solution remaining feasible. In this case the objective function is unbounded below and there is no minimum.

Next, the pivot row must be selected so that all the other basic variables remain positive. A calculation shows that this occurs when the resulting value of the entering variable is at a minimum. In other words, if the pivot column is c, then the pivot row r is chosen so that b_r/a_{cr} is the minimum over all $a_{cr} > 0$. This is called the *minimum ratio test*. If there is more than one row for which the minimum is achieved then a *dropping variable choice rule* can be used to make the determination.

C.4. Integer Linear Programming

An Integer Linear Programming (ILP) problem is an LP problem with the added constraint of all the variables must be integers. Integer programming is NP-hard. 0-1 integer programming or binary integer programming (BIP) is the special case of integer programming where variables are required to be 0 or 1 (rather than arbitrary integers). This problem is also classified as NP-hard.

An integer linear program in canonical form is expressed as:

$$\begin{aligned} &\text{maximize } c^{\mathrm{T}}x \\ &\text{subject to } Ax \leq b, \\ &\qquad x \geq 0 \\ &\text{and } x \in \mathbb{Z} \end{aligned} \tag{C.7}$$

and an ILP in standard form is expressed as:

$$\begin{aligned} &\text{maximize } c^{\mathrm{T}}x \\ &\text{subject to } Ax + s \leq b, \\ &\qquad x \geq 0 \\ &\text{and } x \in \mathbb{Z} \end{aligned} \tag{C.8}$$

C.4.1. Solving ILP problems

The naive way to solve an ILP is to remove the constraint $x \in \mathbb{Z}$, solve the corresponding LP and then round the entries of the solution. This method is called the LP relaxation. However, not only may this solution not be optimal, it may not even be feasible. That is, it may violate some constraint.

Since integer linear programming is NP-complete, many problem instances are intractable and so heuristic methods must be used instead. For example, tabu search can be used to search for solutions to ILP problems. To use tabu search, moves can be defined as incrementing or decrementing an integer constrained variable of a feasible solution, while keeping all other integer-constrained variables constant. The unrestricted variables are then solved for. Short term memory can consist of previous tried solutions while medium term memory can consist of values for the integer constrained variables that have resulted in high objective values (assuming the ILP is a maximization problem). Finally, long term memory can guide the search towards integer values that have not previously been tried.

Other heuristic methods that can be applied to ILP include hill climbing methods, simulated annealing, reactive search optimization, ant colony optimization and Hopfield neural networks.

There are also a variety of other problem-specific heuristics, such as the k-opt heuristic for the traveling salesman problem. Note that a disadvantage of heuristic methods is that if they fail to find a solution, it cannot be determined whether it is because there is no feasible solution or whether the algorithm simply was unable

to find one. Further, it is usually impossible to quantify how close to optimal a solution returned by these methods is.

Bibliography

M. Hu and B. Liu, "Mining and summarizing customer reviews." in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug 2004.

L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent twitter sentiment classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*, 2011.

E. Boiy and M.-F. Moens, "A machine learning approach to sentiment analysis in multilingual web texts." *Information retrieval*, vol. 12(5), pp. 526–558., 2009.

B. L. Ding, Xiaowen and L. Zhang, "Entity discovery and assignment for opinion mining applications." in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2009)*, 2009.

M. Ganapathibhotla and B. Liu, "Mining opinions in comparative sentences." in *Proceedings of International Conference on Computational Linguistics (COLING-2008).*, 2008.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).*, 2013.

B. L. Ding, Xiaowen and P. S. Yu, "A holistic lexicon-based approach to opinion mining," in *Proceedings of the Conference on Web Search and Web Data Mining (WSDM-2008)*, 2008.

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

J. S. Kessler and N. Nicolov, "Targeting sentiment expressions through supervised ranking of linguistic configurations." in *Proceedings of the Third International AAAI Conference on Weblogs and Social Media (ICWSM-2009).*, 2009.

A.-M. Popescu and O. Etzioni, "Extracting product features and opinions from reviews. in proceedings of conference on empirical methods," in *Natural Language Processing (EMNLP-2005)*, 2005.

L.-W. Ku, Y.-T. Liang, and H.-H. Chen, "Opinion extraction, summarization and tracking in news and blog corpora." in *Proceedings of AAAI-CAAW'06.*, 2006.

H. Guo, H. Zhu, Z. Guo, X. Zhang, and Z. Su, "Product feature categorization with multilevel latent semantic association." in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM-2009).*, 2009.

C. Long, J. Zhang, and X. Zhu., "A review selection approach for accurate feature rating estimation." in *Proceedings of Coling 2010: Poster Volume*, 2010.

N. Kobayashi, R. Iida, K. Inui, and Y. Matsumoto, "Opinion mining on the web by extracting subject-attribute-value relations." in *Proceedings of AAAI-CAAW'06.*, 2006.

S. Somasundaran, G. Namata, L. Getoor, and J. Wiebe, "Opinion graphs for polarity and discourse classification." in *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing.*, 2009.

L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition." in *Proceedings of the IEEE*, 1989.

J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: probabilistic models for segmenting and labeling sequence data." in *Proceedings of the 18th International Conference on Machine Learning*, M. K. Publishers, Ed., June 2001, pp. 282–289.

W. Jin and H. H. Ho, "A novel lexicalized hmm-based learning framework for web opinion mining." in *Proceedings of International Conference on Machine Learning (ICML-2009).*, 2009.

N. Jakob and I. Gurevych, "Extracting opinion targets in a singleand cross-domain setting with conditional random fields." in *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2010).*, 2010.

F. Li, C. Han, M. Huang, X. Zhu, Y.-J. Xia, S. Zhang, , and H. Yu, "Structure-aware review mining and summarization." in *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-2010).*, 2010.

Y. Choi and C. Cardie, "Hierarchical sequential learning for extracting opinions and their attributes." in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2010).*, 2010.

S. Huang, X. Liu, X. Peng, and Z. Niu, "Fine-grained product features extraction and categorization in reviews opinion mining." in *Proceedings of the IEEE 12th International Conference on Data Mining Workshops.*, 2012.

T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI-1999).*, 1999.

D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation." *The Journal of Machine Learning Research*, vol. 3, pp. 993– 1022., 2003.

Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai., "Topic sentiment mixture: modeling facets and opinions in weblogs." in *Proceedings of International Conference on World Wide Web (WWW- 2007).*, 2007.

C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis." in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM-2009).*, 2009.

S. Brody and N. Elhadad, "An unsupervised aspect-sentiment model for online reviews." in *Proceedings of The 2010 Annual Conference of the North American Chapter of the ACL.*, 2010.

F. Li, M. Huang, and X. Zhu., "Sentiment analysis with global topics and local dependency." in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-2010).*, 2010.

I. Titov and R. McDonald., "A joint model of text and aspect ratings for sentiment summarization." in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2008).*, 2008.

Q. Su, X. Xu, H. Guo, Z. Guo, X. Wu, X. Zhang, B. Swen, and Z. Su, "Hidden sentiment association in chinese web opinion mining," in *Proceedings of International Conference on World Wide Web (WWW-2008)*, 2008.

Z. Hai, K. Chang, and J. jae Kim, "Implicit feature identification via co-occurrence association rule mining," in *Computational Linguistics and Intelligent Text Processing*, 2011, pp. 393–404.

L. Zeng and F. Li, "A classification-based approach for implicit feature identification," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data Lecture Notes in Computer Science*, 2013, pp. 190–202.

B. Liu, M. Hu, and J. Cheng., "Opinion observer: Analyzing and comparing opinions on the web." in *Proceedings of International Conference on World Wide Web (WWW-2005).*, 2005.

R. N. Carenini, Giuseppe and A. Pauls, "Multi-document summarization of evaluative text." in *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL-2006).*, 2006.

S. Tata and B. D. Eugenio., "Generating fine-grained reviews of songs from album reviews." in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2010).*, 2010.

S. B.-G. Lerman, Kevin and R. McDonald, "Sentiment summarization: Evaluating and learning user preferences." in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2009).*, 2009.

H. Nishikawa, T. Hasegawa, Y. Matsuo, , and G. Kikui., "Optimizing informativeness and readability for sentiment summarization." in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2010).*, 2010.

H. Nishikawa, T. Hasegawa, Y. Matsuo, and G. Kikui., "Opinion summarization with integer linear programming formulation for sentence extraction and ordering." in *Proceedings of Coling 2010: Poster Volume.*, 2010a.

T. Pedersen, S. Patwardhan, and J. Michelizzi., "Wordnet:: Similarity: measuring the relatedness of concepts." in *Demonstration Papers at HLT-NAACL 2004.*, 2004.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch." *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

T. Mikolov, W. tau Yih, and G. Zweig., "Linguistic regularities in continuous space word representations." in *Proceedings of NAACL-HLT.*, 2013.

T. Mikolov, K. Chen, G. Corrado, and J. Dean., "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781*, 2013.

F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *Proceedings of the international workshop on artificial intelligence and statistics.*, 2005.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness." in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003, pp. 805–810.

S. Patwardhan, S. Banerjee, and T. Pedersen, "Using measures of semantic relatedness for word sense disambiguation." in *Proceedings of the Fourth International*

Conference on Intelligent Text Processing and Computational Linguistics, 2003, pp. 241–257.

Z. Wu and M. Palmer, "Verb semantics and lexical selection." in *Annual Meeting of the Association for Computational Linguistics.*, 1994, pp. 133–138.

E. Minkov, R. C. Wang, A. Tomasic, and W. W. Cohen, "NER systems that suit user's preferences: adjusting the recall-precision trade-off for entity extraction," in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, 2006, pp. 93–06.

W. Wei and J. A. Gulla, "Sentiment learning on product reviews via sentiment ontology tree," in *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2010)*, 2010.

Nomenclature

ASCII	American Standard Code for Information Interchange
BOW	Bag of Words
CRF	Conditional Random Fields
DVD	Digital Video Disc
HMM	Hidden Markov Models
IAI	Implicit Aspect Indicator
ILP	Integer Linear Programming
LDA	Latent Dirichlet Allocation
MP3	MPEG-2 Audio Layer III; an audio coding format
NB	Naive Bayes
NER	Name Entity Recognition
NLP	Natural Language Processing
NNLM	Neural Network Language Model

PLSA	Probabilistic Latent Semantic Analysis
PMI	Pointwise Mutual Information
POS	Part-of-Speech
RNTN	Recursive Neural Tensor Network
Skip-gram	Continuous Skip-gram Neural Network Language Model
TF-IDF	Term Frecuency - Inverse Document Frecuency
UTF-8	Universal Character Set + Transformation Format 8-bit