Aspect-based Sentiment Analysis

A Scalable System, a Condition Miner, and an Evaluation Dataset

Fernando O. Gallego

University of Sevilla, Spain

Doctoral Dissertation
Supervised by Dr. Rafael Corchuelo

December, 2018

First published in December 2018 by
The Distributed Group
ETSI Informática
Avda. de la Reina Mercedes, s/n
Sevilla, E-41012. SPAIN

http://www.tdg-seville.info
contact@tdg-seville.info

Classification (ACM 1998): H.1.2 [User/Machine Systems] Human information processing; H.3.4 [Systems and Software] Distributed systems; H.3.5 [On-line Information Services] Web-based services; I.2.7 [Natural Language Processing] Text analysis; I.5.1 [Models] Neural networks.

Support: Supported the Spanish R&D&I programme (grants TIN2013-40848-R and TIN2016-75394-R) and Opileak.

University of Sevilla, Spain

The committee in charge of evaluating the dissertation presented by Fernando O. Gallego in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Software Engineering, hereby recommends _____ of this dissertation and awards the author the grade _____.

_____

Dr. Miguel Toro
Full Professor
University of Sevilla

_____ _____

Dr. Juan Pavón
Full Professor
Complutense Univ. of Madrid

Dr. Juan M. Corchado
Full Professor
University of Salamanca

_____ _____

Dr. José R. Villar
Associate Professor
University of Oviedo

Dr. Paula Montoto
Associate Professor
University of Coruña

To put record where necessary, we sign minutes in _____, _____.

Aspect-based sentiment analysis, by Adriana, aged twelve.

Mining conditions, by Lucía and Daniela, aged two and one, respectively.

Dedicated to my family, chiefly to my daughter Daniela, who gave me the encouragement to go on improving myself as a professional.

Contents

List of figures

List of tables

Acknowledgements

Auliq-Ice said: “the single greatest cause of happiness is gratitude”. Whilst writing these lines I am feeling happy to get to the finish line of my PhD and I wish to express my gratitude to the people who, to a greater or a lesser extent, positively contributed to this stage of my life.

First of all, I would like to thank my parents for their entire life of support and love. I also thank Fátima, for not allowing me to fall down in my worst moments. I really appreciate that she has always supported the decisions that I have made regarding my career. And of course, I cannot forget the biggest happiness in my life: my daughter, Daniela; her existence gave me a strong reason to never give up and to become a reference for her in terms of human values, responsibility, and effort. I cannot neglect my best friend, Fran, one of the best architects I know, who has specialised in architectural visualisation (franciscotirado.es). He is a role model to me. I thank him for our long talks regarding work, technology, business, video-games, films, TV shows, and many other topics.

Special thanks go to the TDG family and the many colleagues from the faculty, the department, and the conferences that I have attended for our fruitful discussions and the time that we have spent together.

I also thank the team at Dinamic Area, S.L. for their happiness and joy. It is always a pleasure to meet all of you. And special thanks go to my partners Roberto and Pablo, who have always trusted me; I can feel our partnership growing into a strong friendship.

Finally, I wish to thank my supervisor, Rafael Corchuelo. He has been the best mentor that I could have had during my PhD, I have always supported his point of view regarding teaching and researching and I will continue to do that. He supported me in my decision not to continue my academic career and return to industry. I am pretty sure that I will not regret that.

Abstract

Aspect-based sentiment analysis systems are a kind of text-mining systems that specialise in summarising the sentiment that a collection of reviews convey regarding some aspects of an item. There are many cases in which users write their reviews using conditional sentences; in such cases, mining the conditions so that they can be analysed is very important to improve the interpretation of the corresponding sentiment summaries. Unfortunately, current commercial systems or research systems neglect conditions; current frameworks and toolkits do not provide any components to mine them; furthermore, the proposals in the literature are insufficient because they are based on hand-crafted patterns that fall short regarding recall or machine learning models that are tightly bound with a specific language and require too much configuration.

In this dissertation, we introduce Torii, which is an aspect-based sentiment analysis system whose most salient feature is that it can mine conditions; we also introduce Kami, which provides two deep learning proposals to mine conditions; and we also present Norito, which is the first publicly available dataset of conditions. Our experimental results prove our proposals to mine conditions are similar to the state of the art in terms of precision, but improve recall enough to beat them in terms of F_1 score. Finally, it is worth mentioning that this dissertation would not have been possible without the collaboration of Opileak, which backs up the industrial applicability of our work.

Resumen

Los sistemas de análisis de sentimiento basados en aspectos son un tipo de sistemas de minería de texto que están especializados en resumir el sentimiento que una colección de revisiones transmite respecto a diversos aspectos de un item. En muchas ocasiones, los usuarios escriben sus revisiones utilizando condiciones; minarlas de manera que puedan ser analizadas es muy importante para mejorar la interpretación de los correspondientes sumarios de sentimiento. Por desgracia, los sistemas comerciales y los sistemas académicos existentes ignoran las condiciones; los frameworks y bibliotecas existentes no proporcionan ningún componente para minarlas; además, las propuestas de la bibliografía son insuficientes ya que están basadas en patrones diseñados manualmente que no ofrecen suficiente cobertura o modelos de aprendizaje automático que están muy ligados a un idioma concreto y requieren de demasiada configuración específica.

En esta tesis presentamos Torii, que es un sistema de análisis de sentimiento basado en aspectos cuya característica más destacada es que puede minar condiciones; también presentamos Kami, que proporciona dos propuestas de aprendizaje profundo para minar condiciones; finalmente presentamos Norito, que es el primer dataset de condiciones disponible públicamente. Nuestros resultados experimentales prueban que nuestras propuestas de minería de condiciones son similares al estado del arte en términos de precisión, pero mejoran la cobertura suficientemente como para batirlos en términos de F_1. Finalmente, es digno de mención que esta tesis no habría sido posible sin la colaboración de Opileak, que garantiza la aplicabilidad industrial de nuestro trabajo.

Chapter 1

Introduction

This chapter introduces our PhD work. It is organised as follows: Section §1.1 introduces the context of our research work; Section §1.2 presents an overview of the related work; Section §1.3 presents the hypothesis that has motivated our work and states our thesis; Section §1.4 summarises our main contributions; Section §1.5 sketches the collaborations that we have conducted throughout the development of this dissertation; finally, Section §1.6 describes the structure of this document.

1.1 Research context

The Web is growing at an ever increasing pace, which confirms that we are in the age of data [4]. Furthermore, it has changed the lives of most people forever. There are tones of new data and services that sprout out everyday [36, 60]. Consequently, many related research fields have grown quickly in recent years, including business intelligence, big data, data science, or text mining.

The goal of text mining is to process the text in a collection of documents in order to produce structured information that can be used to feed business processes [12]. It has a set of features that clearly makes it different from regular data analysis or data mining, for instance: text is sparse and highly-dimensional and it can be analysed at different levels of representation [3]. Sentiment analysis is an important text-mining task [50, 52]. It consists in computing whether the sentiment that a piece of text conveys about an item or any of its aspects is positive, negative, or neutral. Sentiment analysis has found its way in the industry because it helps make many business decisions [1, 28]. For instance, analysing a collection of reviews like "it's a pity that they don't provide a virtual credit card when you register at their offices" unveils a negative sentiment that may help a bank discover a new market need; analysing a collection of reviews like "my first impression was that I didn't feel too excited about the new box design" also unveils a negative sentiment that may help a sushi company forecast the impact of a business decision; and analysing a collection of reviews like "the check-out page is very responsive on desktop computers" unveils a positive sentiment that may help a retailer understand their strengths.

Conditions are important in sentiment analysis [42]. Note that the sentiment about the credit cards and the new box design cannot be generally flagged as negative due to the influence of conditions "when you register at their offices" and "my first impression"; it is the fact that no virtual credit card is provided when someone registers at the a bank's office what is considered negative; similarly, the initial negative sentiment about the new box does not necessarily entail that the customer feels unhappy with it now; analogously, condition "on desktop computers" entails that the positive sentiment about the check-out page is valid on desktop computers, not necessarily on tablets or mobile phones. Mining conditions clearly enhances the results of text mining tasks since it helps interpret the semantics of the original text better.

Unfortunately, mining conditions is not easy at all. Our experience proves that traditional machine-learning approaches do not work well, including Random Forests [9], Support Vector Machines [11], Conditional Random Fields [33], Bayesian Networks [48], or even instance-based learning [2]. Currently, deep learning is gaining impetus thanks to its ability to learn feature-based representations of the input data that facilitate learning classifiers or regressors [34]. They typically build on non-linear transformations that are organised in layers so that the outputs of a layer constitute the inputs of the succeeding one. They have achieved relevant results in computer vision [56, 58] and natural language processing (NLP) [55, 62]. In the case of NLP, it is necessary to transform the input text into vectors using so-called word embedders [5, 39, 49].

Most deep learning approaches build on neural networks [51]. There are two models that are very appropriate in NLP, namely: recurrent neural networks (RNN) and convolutional neural networks (CNN). An RNN is a neural network in which the connections between its units form a directed graph across a sequence [29], which makes them particularly well-suited to deal with sequences of data in which each element depends on the previous ones. Bi-directional recurrent neural networks (BiRNN) [53] are a particular class of recurrent neural networks that can take both the past and the future elements of a sequence into account. Unfortunately, both RNNs and BiRNNs suffer from the so-called exploding and vanishing gradient problems [7, 47], which can be addressed by controlling the data that is passed on to the next training epoch by means of gated recurrent units (GRUs) [29] or bi-directional gated recurrent units (BiGRUs) [44]. A CNN [15, 32] is a class of neural network that is composed of convolution layers, which consist of computation units whose purpose is to transform some regions of the input vectors by means of non-linear functions, and pooling layers, which consists of filters that are intended to subsample the output of the previous convolutions in an attempt to reduce the number parameters that need to be computed in the network.

1.2 Related work

There are many commercial text-mining services, e.g., Lithium, Sprout Social, Lexalytics, Brand Watch, Sysomos, or Opileak. They all can analyse text from sources like blogs, e.g., Word Press or Blog Spot, social networks, e.g., Twitter or Facebook, customer review sites, e.g., Le Guide or Kelkoo, or news sites, e.g., New York Times or NBC News, just to mention a few. Their focus is on extracting topics, measuring the impact of some contents, performing aspect-based sentiment analysis, or computing volumetric insights. Unfortunately, none of them can mine conditions.

There are also several research text-mining systems, for instance: entity-relation extractors [17, 40] mine entities and relations that are used to populate a knowledge base; aspect-based sentiment analysis systems [45, 61] analyse the sentiment that people express in social media regarding some aspects of an item; recommenders [13, 43] make suggestions building on a user's search history, his or her previous orders, or the messages that he or she has exchanged with other people. Unfortunately, none of them can mine conditions.

Regarding text-mining frameworks, UIMA [19] and GATE [14] range amongst the most popular ones. They both help design and implement pipelines in which each stage gets an input message and the annotations produced by the previous stages and computes further annotations. The annotations range from the word stems or part-of-speech tags of the input words to their dependency tags and sentiment scores, to mention a few. They both found their way into the text-mining field since they provide many off-the-shelf components, there are many third-party components, and they are open to integrate custom in-house components. They both support multiple languages and can read text from a variety of sources, including plain text, HTML documents, PDF documents, or databases. There are also many toolkits that provide components to implement specific natural-language-processing tasks, e.g., Stanford Core NLP [35], Freeling [46], and NLTK [8]. Recently, toolkits like OpenNLP [59] and spaCy [30] have gained popularity thanks to their focus on efficiency without sacrificing effectiveness. Unfortunately, none of the previous frameworks or toolkits provide any components to mine conditions.

In the literature, Narayanan and others [42] highlighted the problems of not dealing with conditions in the field of opinion mining. Their proposal is a machine-learning model to compute the sentiment of a conditional sentence, but they did not report on a proposal to mine the conditions; they assumed that the sentences were previously labelled so as to make the conditions explicit. Recently, Skeppstedt and others [54] presented a complementary proposal that can automatically classify a sentence as speculative, contrast, or conditional, but neither was their goal to mine conditions. The naivest proposals to mine conditions are based on searching for user-defined patterns that rely on syntactic anchors. Mausam and others [38] studied the problem in the field of entity-relation extraction; they realised that many usual conditions can be identified by locating adverbial clauses whose first word is one of the sixteen one-word condition connectives in English; unfortunately, they did not report on the effectiveness of their proposal to mine conditions, only on the overall effectiveness of their pro-

posal for entity-relation extraction. Their system was updated recently with new features [37], but their proposal to mine conditions was not. Chikersal and others [10] reported on a similar, but simpler proposal: search for sequences of words in between connectives "if", "unless", "until", and "in case" and the first occurrence of word "then" or a comma. Unfortunately, the previous proposals are not appealing because there are many unusual ways to introduce conditions, which makes hand-crafting patterns with high recall very difficult; furthermore, it is not straightforward to adapt them to other languages in which common connectives are multi-word or there is not a unique, context-agnostic translation for some English connectives. We confirmed the previous claims with our experimental analysis. The only existing machine-learning proposal was introduced by Nakayama and Fujii [41], who worked in the field of opinion mining in Japanese. They devised a model that is based on features that are computed by means of a syntactic parser and a semantic analyser. The former identifies so-called bunsetus, which are Japanese syntactic units that consists of one independent word and one or more ancillary words, as well as their inter-dependencies; the latter identifies opinion expressions, which requires to provide some specific-purpose dictionaries, taxonomies, and heuristics. They used Conditional Random Fields and Support Vector Machines to learn classifiers that make bunsetus that can be considered conditions apart from the others. Unfortunately, their proposal was only evaluated on a small dataset with 3 155 sentences from hotel reviews and the best F_1 score attained was 0.58.

1.3 Research rationale

The information that customer reviews convey is very valuable for companies. According to Forrester [18], sentiment analysis is one of the most important text mining tasks since it is the key to understanding the opinion that customers convey in their reviews [16]. Furthermore, Tractica [6] forecast that the worldwide revenue from sentiment analysis will increase from $123 million in 2017 to $3.80 billion in 2025. According to our own estimations, roughly 10% of the sentences in our evaluation dataset contain conditions that must be mined so that sentiment summaries can be interpreted appropriately. This argumentation leads to the following hypothesis:

> *There is a growing industrial interest in analysing customer reviews and conditions must not be neglected.*

If the previous hypothesis makes sense, then our analysis of the related work reveals that the existing text-mining systems, frameworks, or

toolkits cannot mine conditions, and the few proposals in the literature are insufficient. This clearly justifies working on mining conditions. But there is an additional requirement: the solution must be able to work at web scale. This argumentation leads to the following thesis, which we prove in this dissertation:

> *It is possible to construct an aspect-based sentiment analysis system that can work at web scale and mine conditions. We conjecture that micro services and deep learning are the key.*

1.4 Summary of contributions

Our main contributions are the following:

Torii: it is an aspect-based sentiment analysis system whose most salient feature is that it is the only system that can mine conditions. It can easily scale due to its micro-service architecture. Torii uses Kami to mine conditions. Torii has significatively improved the services that Opileak provide to some of their customers. We have two conference papers [20, 21] and a journal article [27] regarding Torii.

Kami: it consists of two proposals to mine conditions, namely: the first one relies on a deep-learning regressor and the second one relies on a deep neural encoder-decoder. They do not require any user-defined patterns, any specific-purpose dictionaries, taxonomies, or heuristics, can mine conditions in both factual and opinion sentences, and rely on readily-available components. We have two conference papers [23, 26] and a journal article [25] regarding Kami.

Norito: it is a dataset of conditions. It consists of 4.7M sentences in English, Spanish, French, and Italian that were gathered from Ciao.com between April 2017 and May 2017. They were classified into 16 categories according to their sources, namely: adults, baby care, beauty, books, cameras, computers, films, headsets, hotels, music, ovens, pets, phones, TV sets, and video games. We have a conference paper [24] and a poster [22] regarding Norito.

1.5 Collaborations

The motivation to work in the topic of this dissertation comes from a close collaboration with Opileak, a product line of Dinamic Area, S.L. whose main

service is a social media analysis system. Dinamic Area, S.L. is a small company with 20 employees whose the annual turnover is about €350K. Fernando O. Gallego is a shareholder of Dinamic Area, S.L. and a co-founder of Opileak, where he led the development of the system as a CTO. Opileak helped us validate our proposal in two real-world case studies.

The first case study was in the context of a pilot €10K project with the engineering department of a power utility company. Their projects revolve around green energy, power infrastructures, consulting, technology, and chemicals. One of their projects consists in analysing power distribution data that comes from their distribution centres and target consumers. Frequently, they have to design and implement new dashboards to compare those data to a variety of factors, including the power intake of smart cities or the evolution of registered customers. Social media data caught their eye. They started to pay attention to new measures like the number of tweets regarding their company or how many likes their last Facebook publication got. They hired Opileak's opinion mining services and integrated them into their dashboards. The integration was performed by means of the REST API that Opileak provides to retrieve the data. They were interested in exploring the opinion of their customers regarding topics like smart cities, public lights, or the company itself in social media like Twitter or Facebook. This customer used the data to display statistics about the topics in which they were interested. Our proposal helps improve the results by means of a fine-grained analysis at the aspect level that takes conditions into account.

The second case study was in the context of a €1.78M project with the town hall of a well-known city in Spain that is visited by thousands of tourists every year. The town hall has a tourism division that manages guided tours, attractions, monuments, and tourist information offices. They had several sites related to tourism, and they wanted to integrate them all, to improve the front-end design, to improve the availability of their services, and to implement new business rules. Furthermore, they wished to integrate a friendly assistant to help people find their tourism services by means of interactions in natural language. The town hall is also interested in learning about the opinion that people cast in social media about their services. The project is currently ongoing. The town hall will benefit from our results thanks to the finer-grained sentiment summaries that our aspect-based approach provides and the fact that conditions are taken into account. But it is also expected that our deep neural proposals to mine conditions be integrated in a friendly virtual assistant so that it can recognise conditional sentences properly.

Finally, it is worth mentioning that the development of this dissertation has involved a comprehensive study of several approaches to text mining

and the depths of deep learning. This is the reason why Fernando O. Gallego caught Emagine GmbH's eye to work as an NLP engineer for their R&D department in a project in the automotive industry. This company has a yearly turnover of €53M.

1.6 Structure of this dissertation

This dissertation is organised as follows:

- The introduction comprises this chapter, in which we motivate our research work and conclude that there is a need to devise an aspect-based sentiment analysis system that takes conditions into account.
- Chapter §2 reports on our aspect-based sentiment analysis system. We describe the details of our system taking special attention to the way that we implement aspect identification, sentiment analysis, and condition mining in a micro-services architecture.
- Chapter §3 describes our two proposals to mine conditions, namely: a candidate-ranking regressor and an encoder-decoder network. Furthermore, it provides a comprehensive experimental evaluation to support that we have advanced the state of the art not only conceptually, but also empirically.
- Chapter §4 presents our dataset of conditions, which is used to perform our experimental analysis.
- Chapter §5 concludes this dissertation. It summarises our key findings and sketches some future work.

Chapter 2

Torii: the aspect-based sentiment analysis system

Torii is described in this chapter, which is organised as follows: Section §2.1 introduces our system; Section §2.2 presents our data storage component; Section §2.3 describes our data processors component; Section §2.4 describes our client application component; finally, Section §2.5 summarises our conclusions.

2.1 Introduction

In this chapter, we describe the architecture of our system. It relies on three components to which we refer to as Data Storage, Data Processors, and Client Application. The Data Storage component is responsible for persisting and indexing reviews, the related entities, and some meta-data that allows the system to work properly. The Data Processors component helps load reviews, mine conditions, discover new aspects, and summarise the sentiment. The Client Application component provides user interfaces to interact with the system, both as an administrator and a user to fine tune its configuration parameters and visualise its results, respectively.

Micro services have become one of the most suitable approaches to design software architectures whose requirements include high scalability, reliability, and inter-operability. Since they can be easily distributed across a network, message queuing is then the ideal approach to communicate them. This paradigm revolves around the following concepts: producers, consumers, messages, and queues. A producer is a micro service that produces a message and sends it to a queue; a consumer is a micro service that receives messages from a queue; a message encapsulates the data exchanges between a producer and a consumer; a queue is an intermediate cache that helps producers and consumers work as asynchronously as possible. This paradigm has a number of advantages, including that it facilitates decoupling micro services, it ensures the reliability of messages by means of acknowledgements and timeouts, and it allows systems to easily scale out. (Since there is no room for confusion, in the sequel, we refer to our micro services as simply services.)

We implemented our components using a micro service approach with the technologies provided by Spring Boot 2.1.0; the deployment was performed on a cluster of Tomcat 7.0 application servers; the queuing system was implemented using RabbitMQ 3.7.

2.2 The Data Storage component

The Data Storage component is responsible for persisting and querying the data with which our system works. The domain model provides the classes and associations that model them. We use two custom stereotypes in this model, namely: «persistent» and «transient». The difference is that the «persistent» stereotype implies that the objects of the corresponding class are

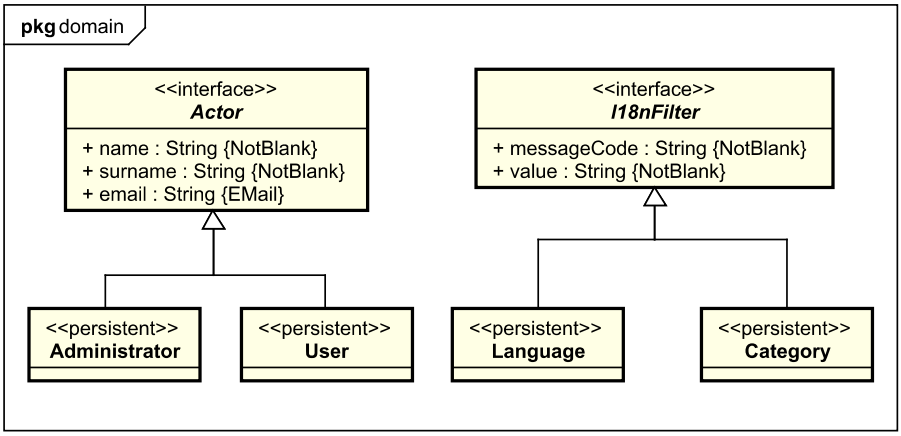


Figure 2.1: *Domain model: actors and filters.*

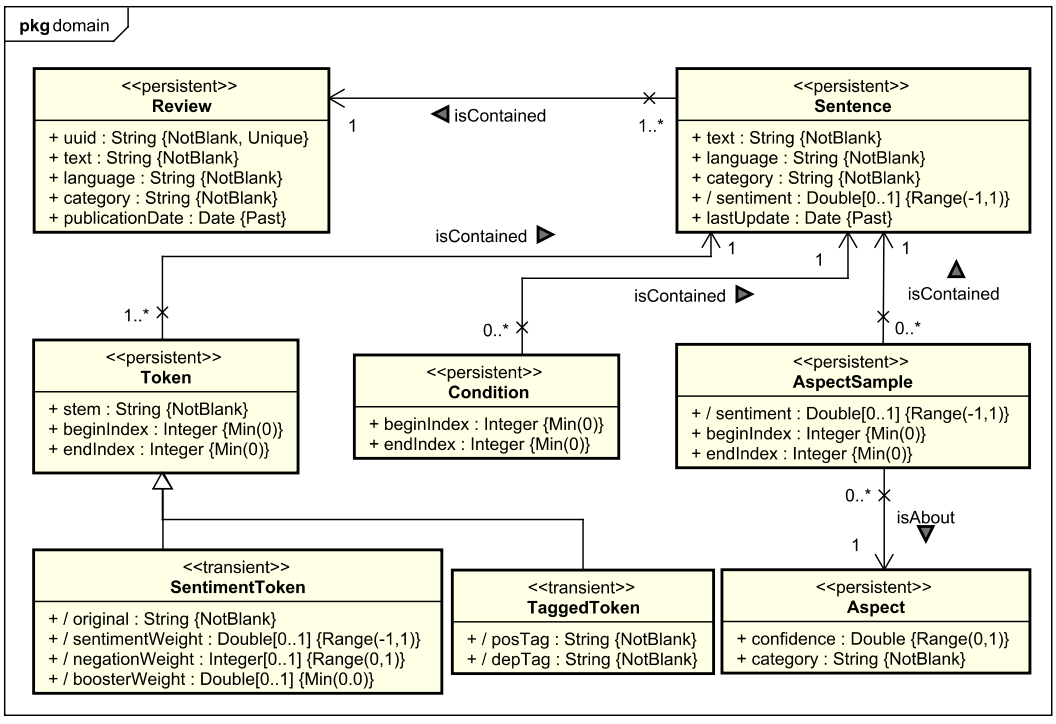


Figure 2.2: *Domain model: reviews.*

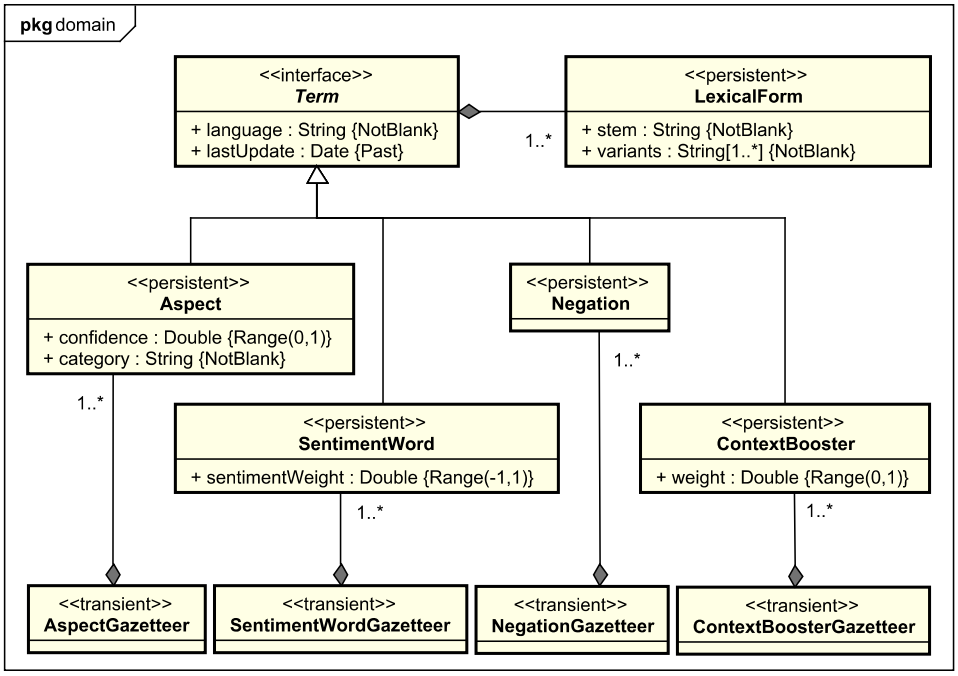


Figure 2.3: *Domain model: terms and gazetteers.*

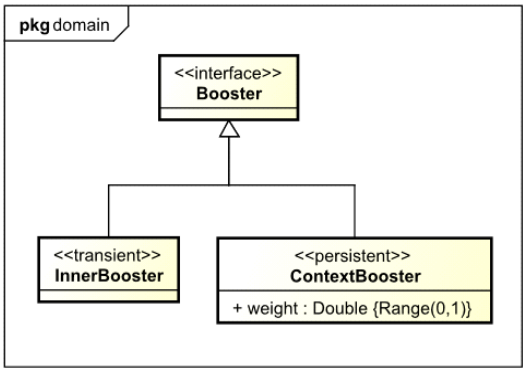


Figure 2.4: *Domain model: boosters.*

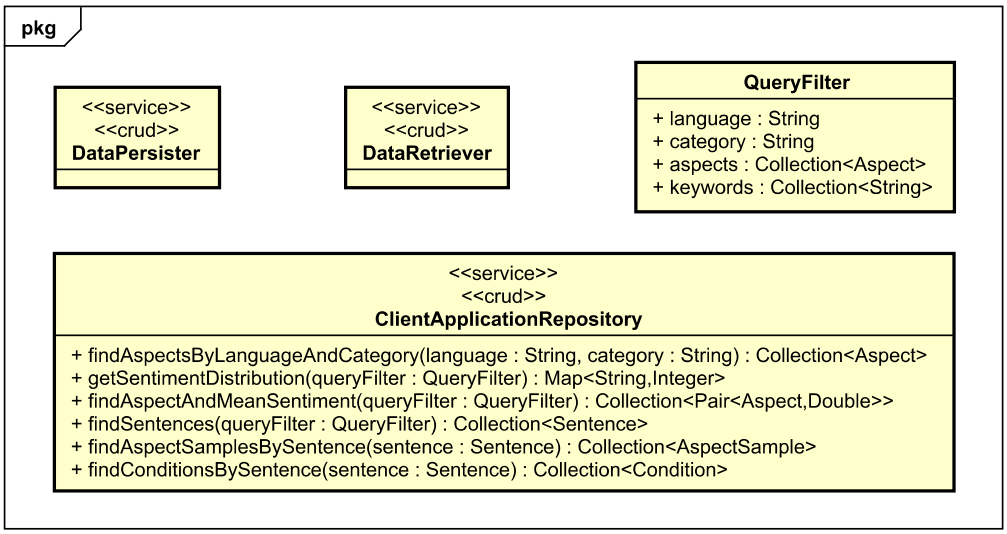


Figure 2.5: *Services: data storage.*

persisted in our database, whereas the objects whose class has the «transient» stereotype are not; that is, they are computed and consumed on the fly.

Figure §2.1 describes the domain model regarding the actors and the internationalised filters. There are two kinds of actors, namely: Administrator, which is responsible for managing the configuration of the system, and User, which can visualise the results. There are two kinds of internationalised filters, namely: Language, which represents the language in which a review can be written, and Category, which represents the category of the item in a review.

Figure §2.2 describes the domain model regarding the reviews. A review is a piece of text that provides an assessment regarding some aspects of an item. Reviews contain one or more sentences. A sentence is a piece of text that is typically delimited by punctuation symbols and expresses a complete though. A sentence contains one or more tokens. A token is the minimum linguistic unit with a syntactical function. A sentiment token is a specialisation of a token that is necessary to compute sentiment. A tagged token is a token that contains some linguistic tags. A sentence may contain one or more conditions. A condition is a clause that conveys a circumstance, a state, or a situation that must be true so that the whole sentence holds. A sentence may also contain one or more aspect samples. An aspect is a feature of

an item and an aspect sample is an instantiation of an aspect in a sentence.

Figure §2.3 describes our term model. A term is kind of an annotation that we use to characterise the tokens of which a sentence is composed. Every term has, at least, a lexical form. A lexical form has a stem that results from lowercasing a word and removing gender, number, and/or case prefixes and/or suffixes in the case of nouns and adjectives or gender, number, and tense forms in the case of verbs. Terms can be either aspects, sentiment words, negations, or context boosters. An aspect is a term that represents a feature of an item. For each aspect, there is a confidence score in range $[0.00, +1.00]$ that determines how likely it is an actual aspect. Each aspect may be assessed by one or more sentiment words. A sentiment word is a term that represents an assessment that is made explicit by means of a sentiment weight in range -1.00 (very negative) up to $+1.00$ (very positive). A negation is a term that changes the sign of the sentiment weight regarding an aspect. A context booster is a term that boosts the sentiment weight regarding an aspect. A gazetteer represents a collection of samples of a specific kind of term. There are four gazetteers, namely: AspectGazetteer, which represents a collection of aspects, SentimentWordGazetteer, which represents a collection of SentimentWord, NegationGazetteer, which represents a collection of negations, and ContextBoosterGazetteer, which represents a collection of context boosters. Note that these gazetteers are readily-available resources for most common languages, except for the gazetteer of aspects. To build the gazetteer of aspects, we start with an initial set of aspects that can be grown by means of an aspect identification process that we describe later.

Figure §2.4 represents the booster model. Generally speaking, a booster is a word that increases the sentiment weight regarding an aspect. They can be classified as context boosters, which are terms, and inner-boosters, which are writing styles.

Note that our model was optimised in a number of ways. The first optimisation is regarding the orientation of the associations; note that they are oriented from the many endpoint to the one endpoint since this can be implemented very efficiently using foreign keys. We have also added attribute lastUpdate to class Sentence and interface Term, which helps determine if a sentence or a term needs to be re-analysed in cases in which a gazetteer is updated. An additional optimisation consists in using attributes beginIndex and endIndex in classes Token, Condition, and AspectSample, which provide their offsets in the sentences in which they are contained. Note, too, that classes SentimentWord, Booster, and Negation are transient because their objects can be computed on the fly very easily and they are only used in the process that

computes sentiment scores, at the sentence or the aspect level, or the sentiment weight, the negation weight, and the booster weight of a token; we provide additional details on that process in the following section. Note, too, that the gazetteer classes are transient since they are used as entry points to the corresponding collections of terms only at run time; so, there is no need to persist them explicitly. Last, but not least, we created an ElasticSearch index to improve performance when retrieving objects of class Sentence.

Figure §2.5 presents the services that the DataStorage component provides, their public methods, and an ancillary class QueryFilter that summarises the parameters of some of those methods. Note that the classes that model services are annotated with the «service» stereotype; some of them are also annotated with the «crud» stereotype, which indicates that they provide create, retrieve, update, and delete methods.

The DataStorage component has three services, namely: DataPersister, which is responsible for persisting the data from the services of the DataProcessors component, DataRetriever, which is responsible for retrieving the data of the services of the DataProcessors component, and ClientApplicationRepository, which is responsible for managing the queries issued by the ClientApplication component. The ClientApplicationRepository service provides some additional methods, namely: find the aspects for a given language and category; find the number of positive, negative, and neutral sentences in a given language and category that comment on some given aspects and contain some given key words; find the aspects and their average sentiment in the aspect samples of the sentences in a given language and category that comment on some given aspects and contain some given key words; find the sentences in a given language and category that comment on some given aspects and contain some given key words; and find the aspect samples and the conditions in the sentences that result from the previous query.

The DataStorageComponent was implemented using the following technologies: we used the Spring Data component provided by Spring Boot 2.1.0 to implement the repositories that handle writing data to the database and querying them, MySQL 5.5 to persist our data, and ElasticSearch 6.3.0 to implement full-text indices.

2.3 The Data Processors component

The Data Processors component is responsible for loading reviews, preprocessing them, and running the processes to mine conditions, discover new

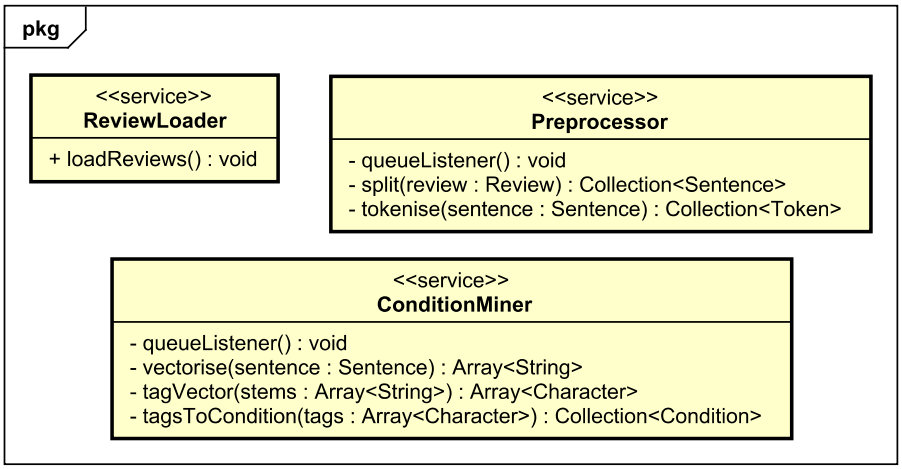


Figure 2.6: *Services: loading reviews.*

aspects, and summarise the sentiment. Figures §2.6 and §2.7 present the services, their relevant attributes, if any, their public methods, which can be called by external services synchronously, their listener methods, which are subscribed to queues, and their private methods, which are used to execute their corresponding algorithms.

Figure §2.8 shows the communications involved in loading new reviews into the DataStorage component. External resources are annotated with the «resource» stereotype.

We assume that there is an external service that helps search for a collection of reviews on the Web and download them to a CSV file with the following columns: unique identifier, text, language, category, and publication date. The CSV file is loaded into our system (1.1) by means of the ReviewLoader service, which forwards them to the Preprocessor service (1.2), which is responsible for splitting the input reviews into sentences, the sentences into tokens, and stemming them. The results are persisted to the Data Storage component (2.1) and they are also forwarded to the ConditionMiner service (2.2), which is responsible for mining conditions and persisting them to the Data Storage component (3). Note that the previous services run in a pipeline every time that an administrator loads a new collection of reviews into the system.

Example 1: Figure §2.10 shows an example of review, which we assume was retrieved from a web site and stored in a CSV file. After reading from it, the

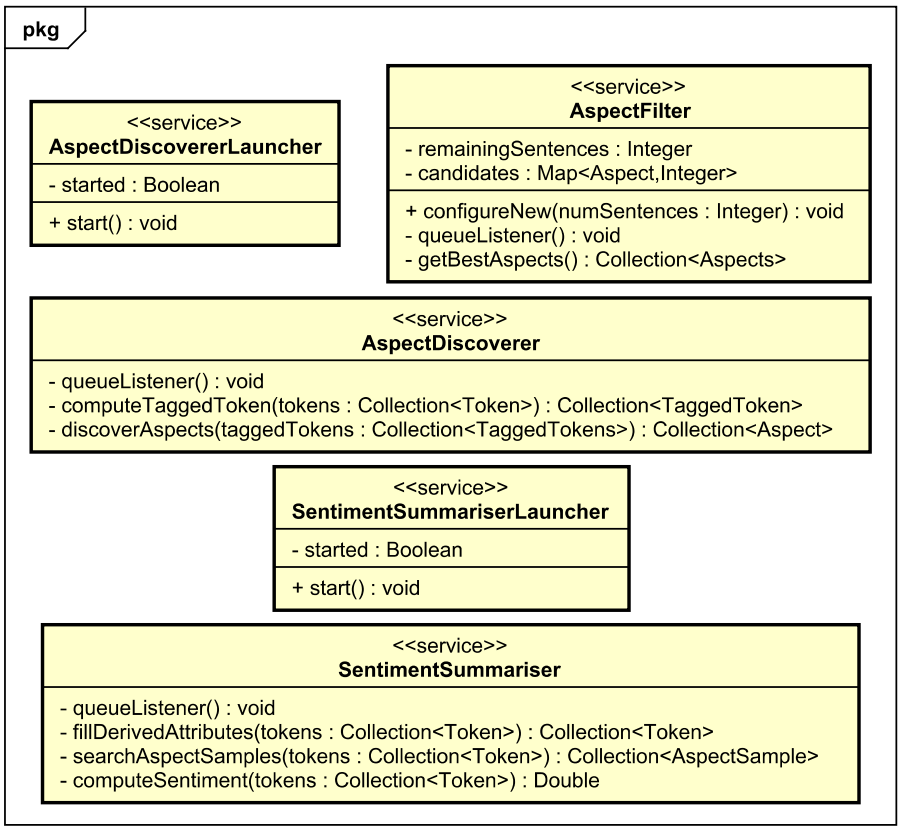


Figure 2.7: *Services: discovering aspects and summarising sentiment.*

ReviewLoader produces the Review object (1). Then, the Preprocessor service splits the review text into sentences (2). Finally, for each sentence, a sequence of tokens is produced. Figure §2.11 shows the tokens of the first sentence. □

The proposals that can be used in the ConditionMiner service are detailed in the following chapter. Given that we present two proposals, we implemented this service as independently as possible from them; that is, we just pass the sentence to the model and it returns the Condition objects from the conditions of the sentence.

Figure §2.9 shows the communications involved in discovering new aspects and summarising sentiment.

There is a service called AspectDiscovererLauncher that initiates the process. It first uses the DataRetriever service to retrieve the sentences to be anal-

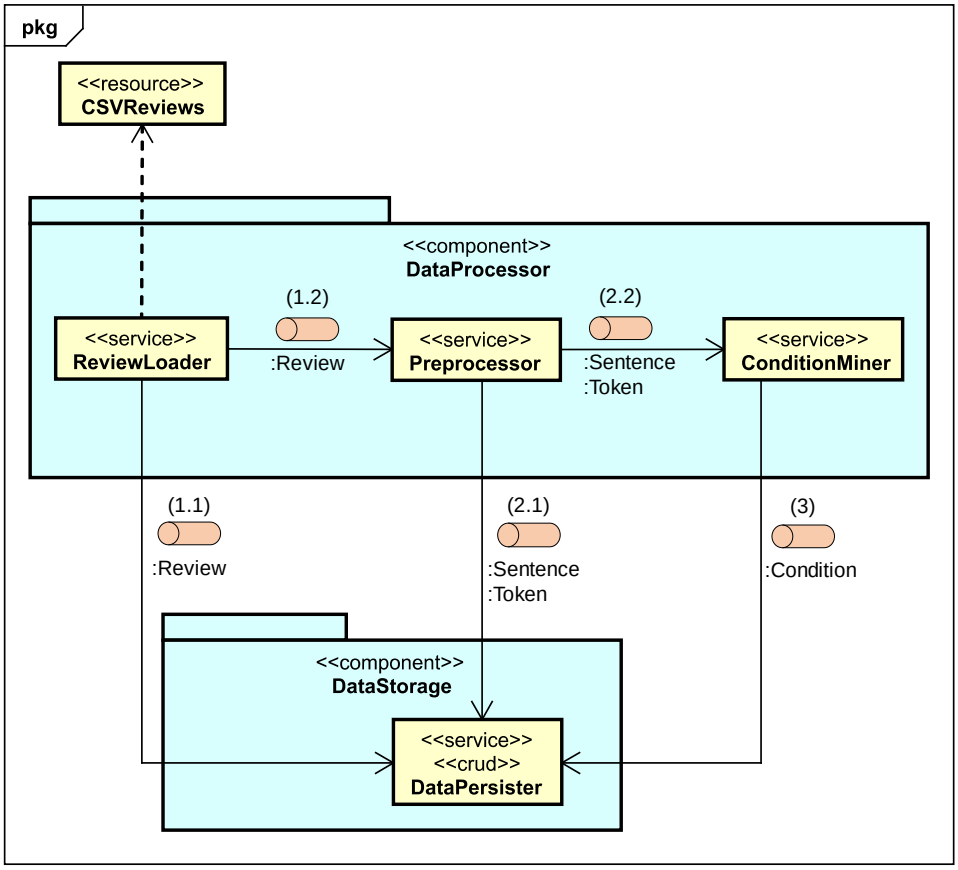


Figure 2.8: *Communications: loading reviews.*

ysed (1.1). It then informs the AspectFilter service of the number of sentences (1.2). It then starts passing the sentences on to the AspectDiscoverer service (1.3), which discovers as many new aspects as possible. Every time the AspectDiscoverer service finishes working on a sentence, it passes them on to the AspectFilter service (1.4), which retains the most representative ones and stores them in the database by means of the DataPersister service (1.5).

The process to discover new aspects was inspired by Hu and Liu's [31] results, namely: given an input sentence, it iterates over the tokens of the sentence, creates the corresponding tagged tokens, and initialises their part-of-the-speech tags and their dependency tags. Then, it selects the nouns and noun-phrases that are non-stop words and occur in, at least, one percent of the sentences; this results in a set of candidate aspects for which we

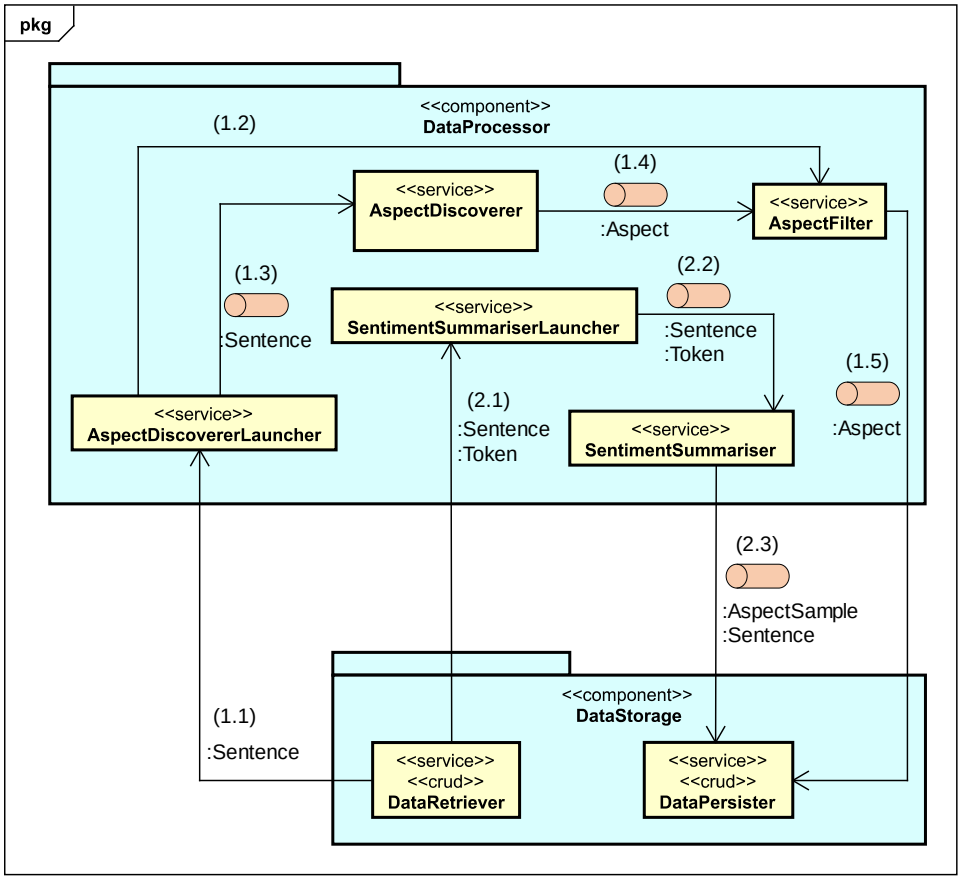


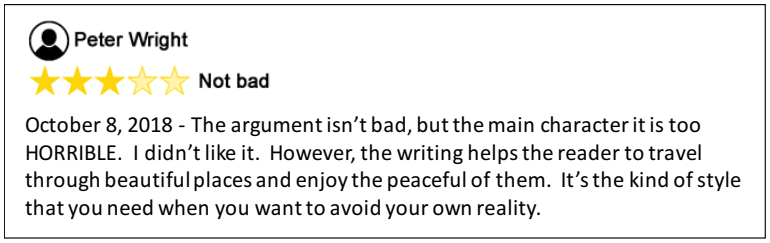
Figure 2.9: *Communications: discovering aspects and summarising sentiment.*

compute a confidence score that normalises the aspect frequency as follows:

$$c_k = \frac{f_k - \min\{f_i,...,f_n\}}{\max\{f_i,...,f_n\} - \min\{f_i,...,f_n\}}$$

where c_k represents the confidence score of aspect k, f_k represents its frequency in the sentences that have been analysed, $\min\{f_1,\ldots,f_n\}$ represents the minimum frequency of all aspects, and $\max\{f_1,\ldots,f_n\}$ represents the maximum frequency of all aspects. It is not difficult to realise that the previous computation results in a score in range $[0.00,+1.00]$, so that the closer to zero the less confidence that the corresponding aspect is an actual aspect and the closer to one the more confidence that it is an actual aspect.

Example 2: Figure §2.12 shows the four sentences of our running exam-



(1)

UUID	Text	Language	Category	Publication Date
4c4b4f66-b21c-47a4-a4ff-5be1fdca6bd6	The argument isn't bad, but the main character is too HORRIBLE. I didn't like it. However, the writing helps the...	English	Books	2018/10/08

(2)

Text	Language	Category	Sentiment	Last Update
The argument isn't bad,...	English	Books	0.0	2018/12/17
I didn't like it.	English	Books	0.0	2018/12/17
However, the writing...	English	Books	0.0	2018/12/17
It's the kind of style that...	English	Books	0.0	2018/12/17

Figure 2.10: *Review loading and preprocessing: example (1/2).*

ple in the first column. The second column represents the aspect samples that are discovered by the AspectDiscoverer service, namely: it discovers the aspect samples argument and main character from the first sentence; it cannot discover any aspect samples from the second sentence; it discovers the aspect samples writing, reader, places, and peaceful from the third sentence; and it discovers the aspect samples style and reality from the last sentence. Realise that all of the aspect samples correspond to noun or noun phrases in the original sentences. □

There is another service called SentimentSummariserLauncher that initiates the process to summarise sentiment. It first uses the DataRetriever service to retrieve the sentences to be analysed (2.1). It then passes them on to

Stem	Begin Index	End Index
the	0	3
argument	4	12
be	13	15
not	15	18
bad	19	22
but	23	26
the	27	30
main	31	35
character	36	45
be	46	48
too	49	52
horrible	53	62

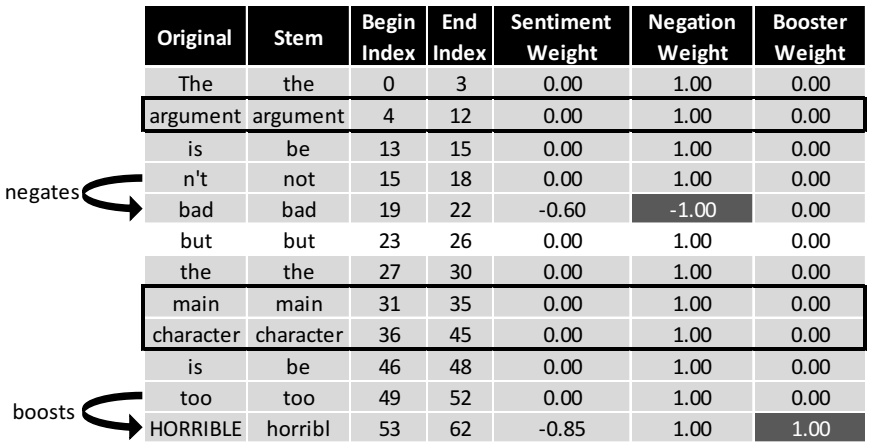
Figure 2.11: *Review loading and preprocessing: example (2/2).*

	Sentences	Aspect Samples
(1)	The argument isn't bad, but the main character is too HORRIBLE.	argument, main character
(2)	I didn't like it	-
(3)	However, the writing helps the reader to travel through beautiful places and enjoy the peaceful of them	writing, reader, places, peaceful
(4)	It's the kind of style that you need when you want to avoid your own reality	style, reality

Figure 2.12: *Aspect discovering: example.*

the SentimentSummariser service (2.2), which actually summarises the sentiment on a per-aspect basis and stores the results by means of the methods offered by the DataPersister service.

The process to summarise the sentiment was inspired by Taboada and others's [57] results, namely: it first iterates over the tokens of the input sentence, next creates their corresponding sentiment tokens, and then initialises their derived attributes as follows: attribute sentimentWeight is initialised to the



Original	Stem	Begin Index	End Index	Sentiment Weight	Negation Weight	Booster Weight
The	the	0	3	0.00	1.00	0.00
argument	argument	4	12	0.00	1.00	0.00
is	be	13	15	0.00	1.00	0.00
n't	not	15	18	0.00	1.00	0.00
bad	bad	19	22	-0.60	-1.00	0.00
but	but	23	26	0.00	1.00	0.00
the	the	27	30	0.00	1.00	0.00
main	main	31	35	0.00	1.00	0.00
character	character	36	45	0.00	1.00	0.00
is	be	46	48	0.00	1.00	0.00
too	too	49	52	0.00	1.00	0.00
HORRIBLE	horribl	53	62	-0.85	1.00	1.00

Figure 2.13: *Sentiment analysis: example.*

sentiment weight of the corresponding word, which is looked up in the sentiment word gazetteer; attribute negationWeight is initialised to -1.00 if there is a nearby negation word, which are provided by the negation gazetteer; and attribute boosterWeight is initialised to a value that accounts for the context boosters and the inner boosters that are found in the nearby tokens and the tokens themselves; then it searches for samples of the aspects that were discovered by the AspectDiscoverer service; for each aspect sample, its derived sentiment attribute is computed as the sum of the most negative and the most positive overall sentiment weight of the tokens of the sample (the overall sentiment weight of a token is the sum of its sentiment weight and booster weight multiplied by the negation weight). The process also computes the sentiment at the sentence level using the same procedure on the whole sentence instead of the aspect samples.

Example 3: Figure §2.13 shows the sequence of SentimentTokens of sentence "The argument isn't harsh, but the main character is too HORRIBLE". Note that there are two updates to the tuples after the computation of their initial values. Our method focuses on the two sentiment words in our example, namely: "harsh", which changes its negation weight due to the nearby "n't", and "HORRIBLE", which changes its booster weight due to the context booster "too" and the inner booster that consists in writing the whole word with capital letters. Then, the sentence is split into two sequences of tokens that correspond to aspects "argument" and "main character"; token "but" does not belong to any sequences since it is just a nexus between them. The sentiment

for the first aspect is 0.60 and −1.00 for the second aspect. The sentence sentiment is the sum of those scores, that is, $0.60 + (-1.00) = -0.40$. □

The DataProcessors component was implemented using the general technologies provided by Spring Boot 2.1.0 and the following specific technologies: the Preprocessor service was implemented using Stanford Core NLP 3.8.0 to split and tokenise sentences; the ConditionMiner service was implemented using Gensim 2.3.0 to compute word embedders using a Word2Vec implementation, Keras 2.0.8 with Theano 1.0.0 to build and train our deep neural network, and DeepLearning4j 1.0.0-beta3 to bridge it to Java; and the AspectDiscoverer service was implemented using Stanford Core NLP 3.8.0 to compute the part-of-speech tags and generate dependency trees. The deep neural network on which the ConditionMiner relies was developed in-house; we provide the many additional details behind the scene in the following chapter.

2.4 The Client Application component

The Client Application component provides the user interfaces used to interact with the system depending on the role of the user.

The administrator role can use the following functionalities: managing the gazetteer of aspects, managing the gazetteer of sentiment words, managing the review loading process, managing the sentiment summarisation process, and managing the aspect discovery process. (Note that the negation and the booster gazetteers need not be managed, since they are readily-available resources in most common languages.)

Figure §2.14 shows the page in which an administrator can manage the gazetteer of aspects. To add a new aspect, an administrator must write, at least, one stem and one variant form. When an administrator adds a new aspect, the confidence value is 1.00, so they are the perfect seeds to discover new aspects. Furthermore, the list of current aspects is shown as a table with their stems and their corresponding confidence scores. Figure §2.15 shows the page in which an administrator can manage the gazetteer of sentiment words. To add a new sentiment word, an administrator must write, at least, one stem, one variant form, and the sentiment weight.

The user role can display a dashboard with the following widgets: a set of filters, a sentiment chart, a list of aspects with sentiment, and a list of sentences with sentiment. Figure §2.16 shows the filters and the sentiment chart.

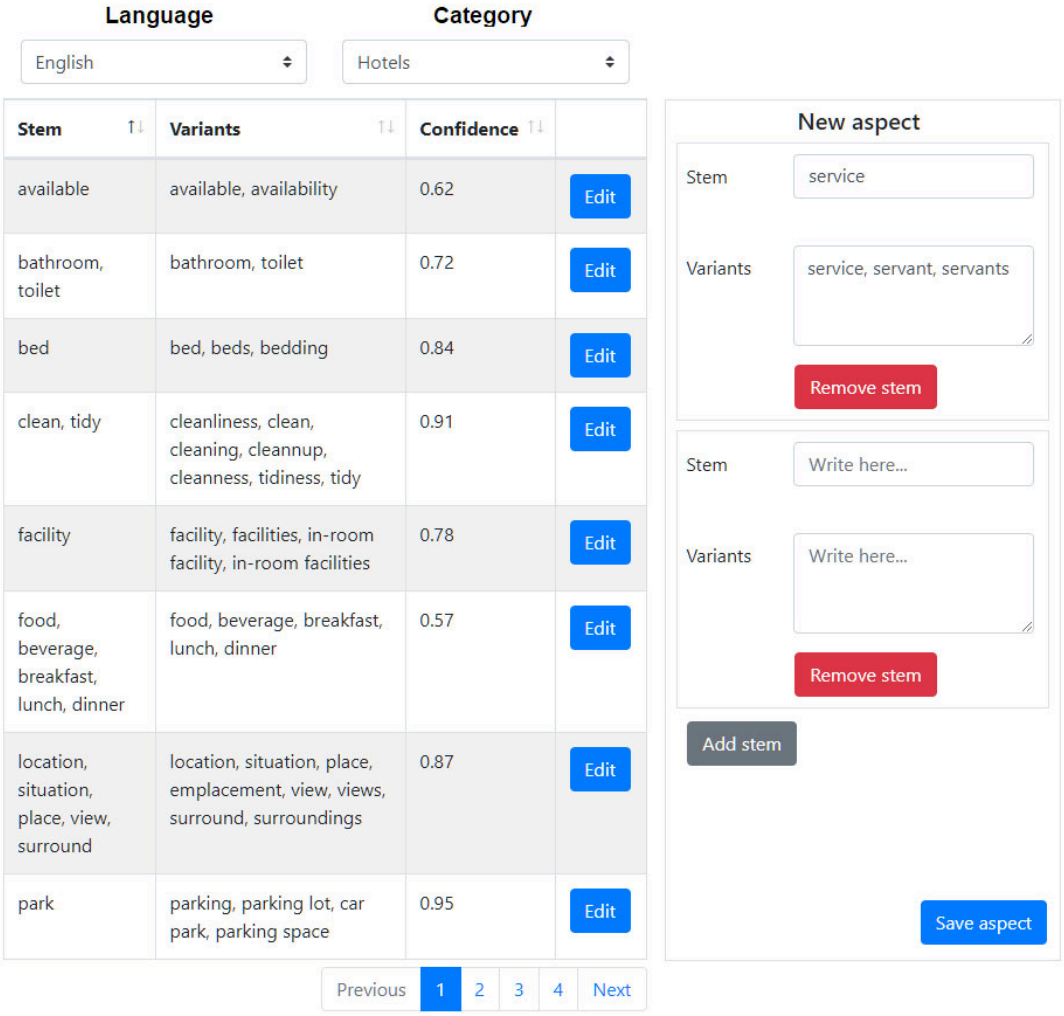


Figure 2.14: *Administrator interface: aspect gazetteer.*

There are three types of filters, namely: main filters, which allow to filter the results by language and category; aspect filters, which allow to filter the results by the sentences whose aspect samples are related to them; and text filters, which allow to filter the results by key word. A user must always select a language and a category to display the results. Regarding the aspects, the dashboard shows a paginated list with their stems in the filtered reviews and allows the user to select zero, one, or more aspects to filter the results. Regarding the key words, the user can write a string to filter the results. This filter performs an exact-match search. The sentiment chart represents the dis-

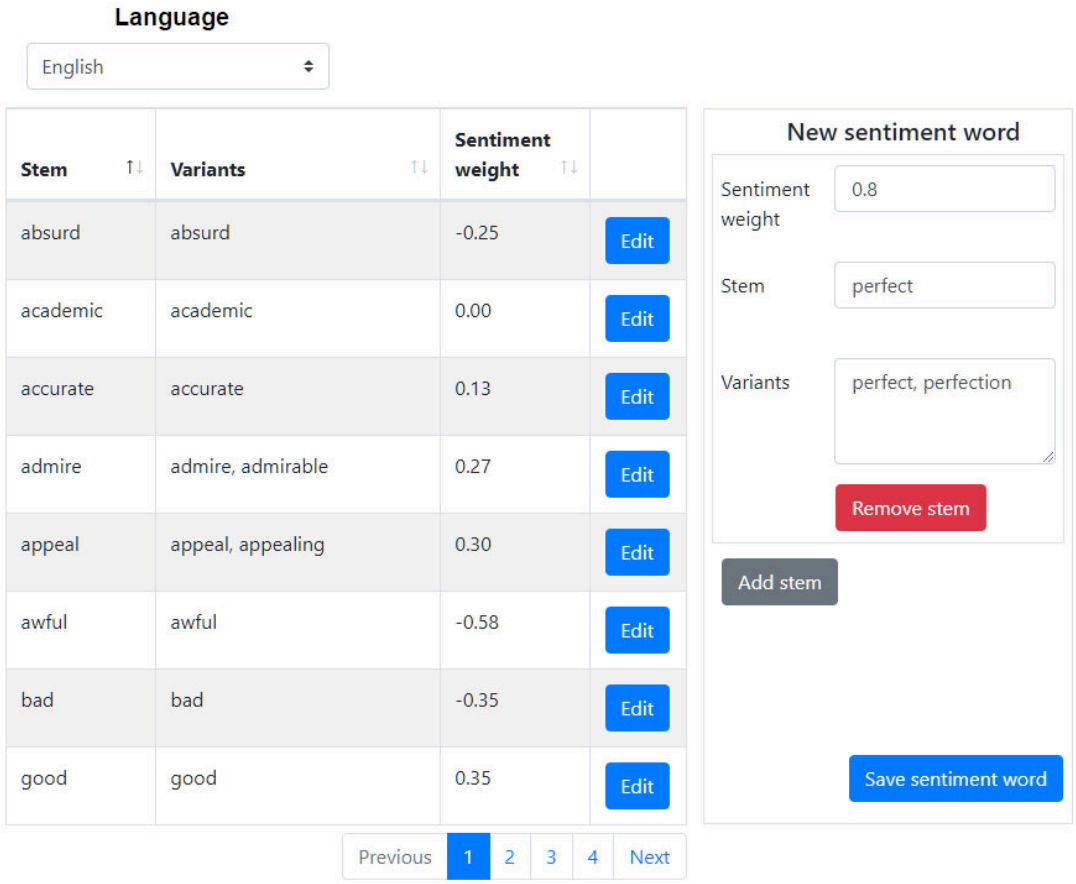


Figure 2.15: *Administrator interface: sentiment word gazetteer.*

tribution of sentiment across the three sentiment classes, namely: positive, neutral, and negative. This distribution is computed as the sum of reviews whose sentence-level sentiment score is above, approximately, or below 0.00, respectively. Figure §2.17 shows a paginated list of aspects with sentiment. It is a table with two columns, namely: aspect and sentiment. the aspect column represents the stem of the aspects and the sentiment column represents the averaged sentiment of the aspect for all reviews. Figure §2.18 shows the paginated list of sentences with sentiment. It is a table with two columns, namely: the text of the sentence and its corresponding sentiment. The text of the sentence represents the original text of the sentence, without any preprocessing. Furthermore, aspect samples and conditions are highlighted so that the user can spot them more easily. The sentiment column represents the

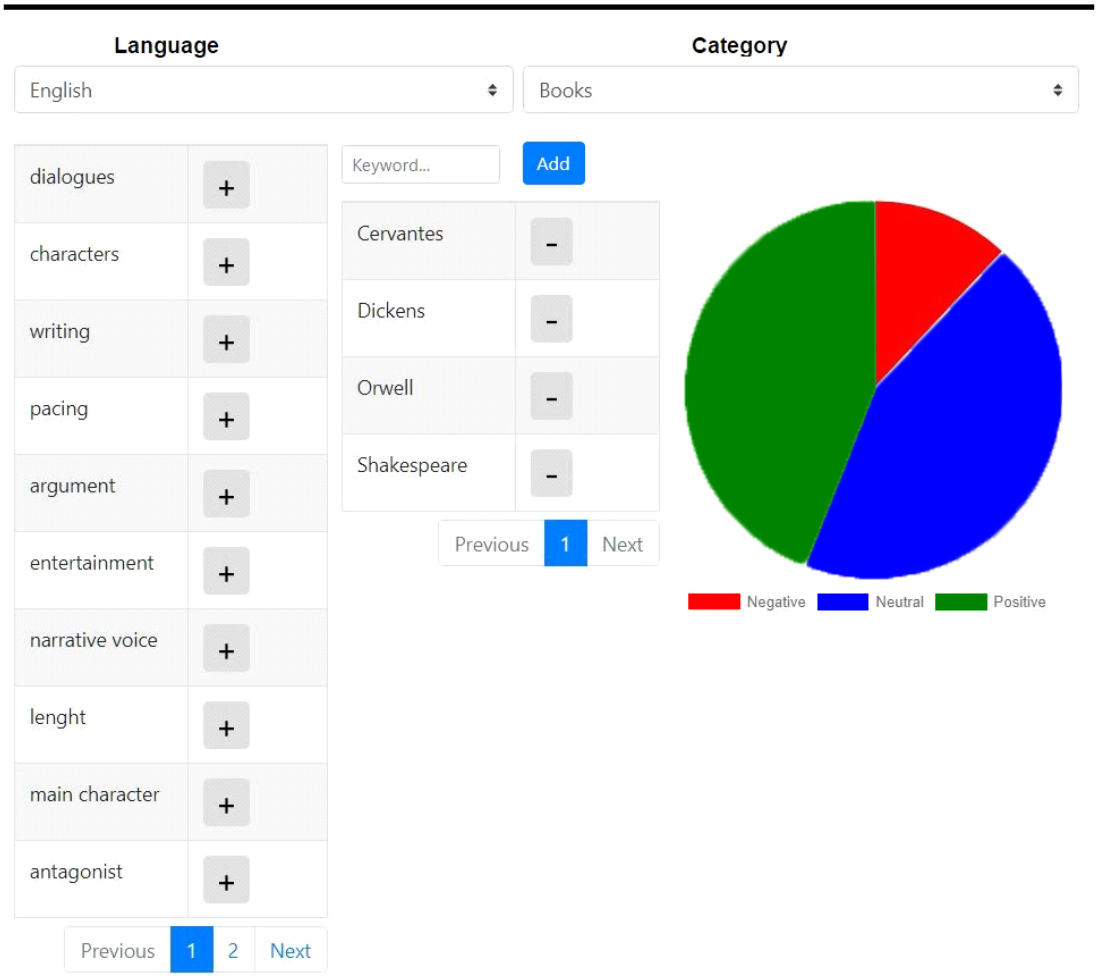


Figure 2.16: *User interface: filters and sentiment chart.*

sentiment sentence-level that we have computed for each sentence.

This component was implemented using the following technologies: Spring Boot 2.1.0 to develop the application as a whole, Spring Security 5.0 to secure the application, Thymeleaf 3.0.4 to create application views, JavaScript 1.8 to implement some client-side functionalities, and Bootstrap 4.0.0 with CSS3 to apply some style templates and personalise them, respectively.

Aspect	Sentiment
main character	-0.39
secondary stories	-0.97
dialogues	0.14
pacing	-0.24
writing	-0.04

Previous 1 Next

Figure 2.17: *User interface: table of aspects with sentiment.*

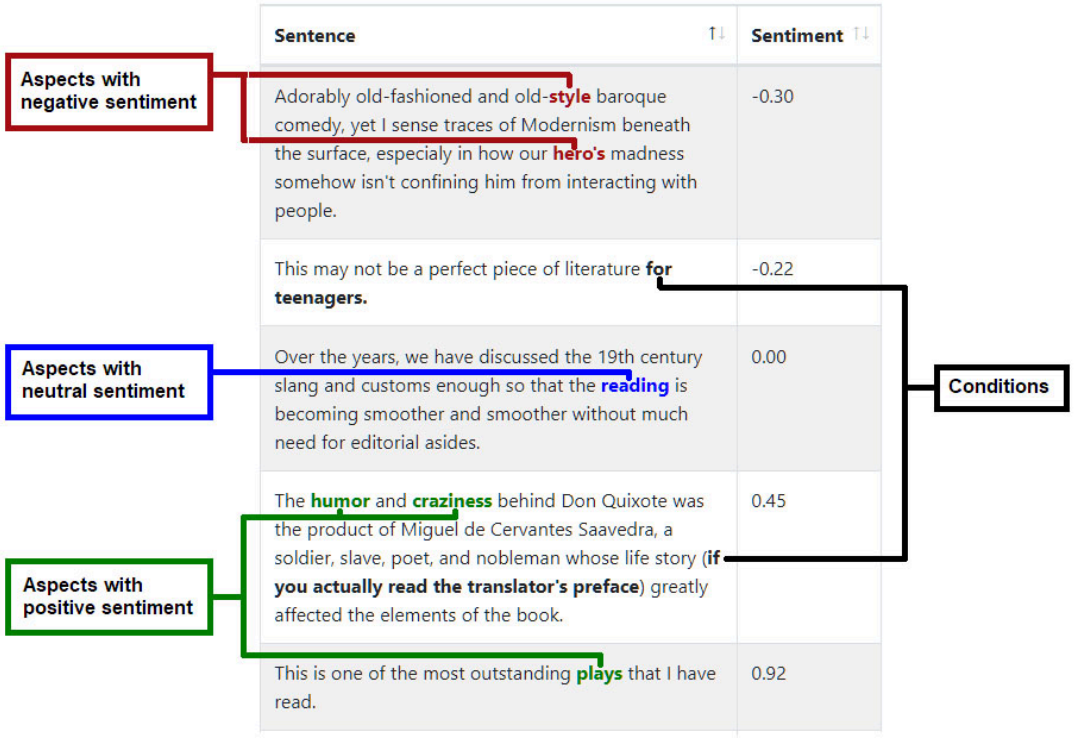


Figure 2.18: *User interface: table of sentences with sentiment.*

2.5 Summary

In this chapter, we have presented Torii, which is the first aspect-based sentiment analysis system that can mine conditions. It relies on three components, namely: the Data Storage component, which is responsible for persisting and indexing reviews; the Data Processors component, which provides some micro services to load reviews, plus some micro services to mine conditions, discover new aspects, and summarise the sentiment; and the Client Application component, which provides user interfaces to interact with the system, both as an administrator and a user. It can scale out due to its micro service architecture. Furthermore, it supports working on different languages and domains so that it is flexible enough.

Chapter 3

Kami: the proposals to mine conditions

Kami provides two proposals to mine conditions that are described in this chapter. It is organised as follows: Section §3.1 introduces our proposals; Section §3.2 describes our candidate-ranking proposal; Section §3.3 describes our encoder-decoder proposal; Section §3.4 presents our experimental analysis; finally, Section §3.5 summarises our conclusions.

3.1 Introduction

In this chapter, we present two deep learning proposals to mine conditions that overcome the drawbacks of the state-of-the-art.

A conditional sentence, or conditional for short, is composed of two clauses, namely: a condition and a consequent. The condition describes a state, a factor, or a circumstance that must hold so that the consequent holds. Usual conditionals are expressed by means of grammatical patterns that are used to make a difference amongst zero-conditionals, which convey general truths, first conditionals, which convey possible conditions and their likely results, second conditionals, which convey hypothetical conditions and their likely results, and third conditionals, which convey unreal past conditions and their likely results in the past. The connectives and the verb tenses that are used to introduce these conditionals are well-known in the literature. Typical connectives include "if", "when", "even if", "only if", "as long as", "providing", "provided", "supposed", "supposing", "what if", "unless", or "in case". For instance, "if you heat ice, then it melts" is a zero conditional sentence; "unless I don't pass my exams, I will get my degree" is a first conditional sentence; "I'd quit my job as long as I weren't in debt" is a second conditional sentence; and "we'd have attended the show even if she hadn't bought the tickets" is a third conditional sentence. Unusual conditionals do not fit the patterns that characterise the previous types of conditionals. There is not a standard set of connectives or verb tenses to introduce their conditions. For instance, in sentence "put a grain in the tank and the engine will break", the condition "put a grain in the tank" describes an action that is sufficient to break an engine. In sentence "people who like The Beatles will enjoy The Kinks", condition "who like The Beatles" expresses a taste that characterises a group of people that will likely enjoy "The Kinks". In sentence "after you book Aladdin, you'll get a discount for a DVD", condition "after you book Aladdin" expresses a specific time after which one can get a discount. In sentence "it smells terribly bad near the main gate", the condition "near the main gate" expresses a place where it does not smell well. In sentence "you must show your ID due to law restrictions", the condition "due to law restrictions" expresses the reason why an ID is mandatory.

Our proposals are based on deep learning, whose focus is on learning feature-based representations of the input data that facilitate learning classifiers or regressors. They typically build on non-linear transformations that are organised in layers so that the outputs of a layer constitute the inputs of the succeeding one. It is worth mentioning that we also experimented

```
method train(ds) returns r
  T := ∅
  for each (s, L) ∈ ds do
    C := generateCandidates(s)
    for each c ∈ C do
      z := computeScore(c, L)
      T := T ∪ {(c, z)}
    end
  end
  r := learnRegressor(T)
end
```

```
method apply(s, r, θ) returns R
  C := generateCandidates(s)
  R := ∅
  for each c ∈ C do
    z := apply r to c
    if z > θ then
      R := R ∪ {(c, z)}
    end
  end
  R := removeOverlaps(R)
end
```

Figure 3.1: *Candidate ranking: main methods.*

with a variety of approaches, including Random Forests, Support Vector Machines, Conditional Random Fields, Bayesian Networks, or even instance-based learning. Unfortunately, none of them were good enough. We have performed a comprehensive experimental analysis on Norito, a dataset of conditions that is presented in the following chapter. Our results prove that our proposals are comparable to others in terms of precision, but they improve recall enough to beat them in terms of F_1 score.

3.2 A candidate-ranking proposal

In this section, we describe our first proposal to mine conditions, which consists in learning a regressor that assesses how likely a candidate condition is an actual condition[†1]. Hereinafter, we refer to condition candidates as candidates and to actual conditions as conditions since there is no room for confusion. We first describe the main methods and then provide an insight into the ancillary methods.

3.2.1 Main methods

Figure §3.1 shows the main methods of our proposal, namely: `train`, which learns a regressor that assesses candidates, and `apply`, which computes the best candidates in a sentence.

Method `train` takes a dataset `ds` as input and returns a regressor `r`. The input dataset is of the form $\{(s^{(i)}, L^{(i)})\}_{i=1}^{n}$, where each $s^{(i)}$ denotes a sentence and

[†1]The implementation is available at https://github.com/FernanOrtega/candidate-ranking.

each $L^{(i)}$ denotes a set of labels that identify the conditions in that sentence ($n \geq 0$). The output regressor is a function that given a candidate returns a score that assesses how likely it is an actual condition. The method first initialises training set T to the empty set and then loops over dataset ds; for each sentence s and set of labels L in ds, it first computes a set of candidates; then, for each candidate c, it computes a score z and stores a tuple of the form (c, z) in training set T. When the main loop finishes, it learns a regressor from T using a deep-learning approach.

Method *apply* takes a sentence s, a regressor r, and a threshold θ as input and returns a set R of tuples of the form $\{(c^{(i)}, z^{(i)})\}_{i=1}^{n}$, where each $c^{(i)}$ denotes a candidate and $z^{(i)}$ its corresponding score, which must be equal to or greater than threshold θ ($n \geq 0$). The method first generates the candidates in s, stores them in set C, and initialises R to an empty set; it then iterates over set C; for each candidate c in set C, it first computes its score by applying regressor r to it; if it is equal to or greater than threshold θ, then candidate c is added to the result set. When the main loop finishes, R provides a collection of candidates and scores; before returning it, we must remove the candidates that overlap others with a higher score. The candidates in R are considered the actual conditions in the input sentence s.

3.2.2 Generating candidates

Our first ancillary method is *generateCandidates*, which takes a sentence as input and returns a set of candidates. A naive approach would simply generate as many sub-strings as possible, but it would be very inefficient because a sentence with n words has $O(n^2)$ such sub-strings. In order to reduce the candidate space, we use a non-overlapping sequence of blocks that created from a dependency tree.

Method *generateCandidates* first computes the dependency tree of the input sentence, lowercases the words in its nodes, and stemmises them. It then computes a non-overlapped sequence of blocks, which are sequences of tokens of the form (w, d, p), where w denotes a stem, d the dependency tag that links its node in the dependency tree to its parent, if any, and p its position in the sentence. To compute a non-overlapped sequence of blocks, we first recursively select the sub-trees whose depth is exactly two because it helps select syntactical units like noun-phrases, adjective-phrases, or verb-phrases; then, we repeat the procedure for the nodes whose depth is equal to one because it helps to select smaller syntactical units; and, finally, we select the nodes that have only one token. The depth of a sub-tree is the distance from its root node to its deepest leaf node. Note that we need to ensure

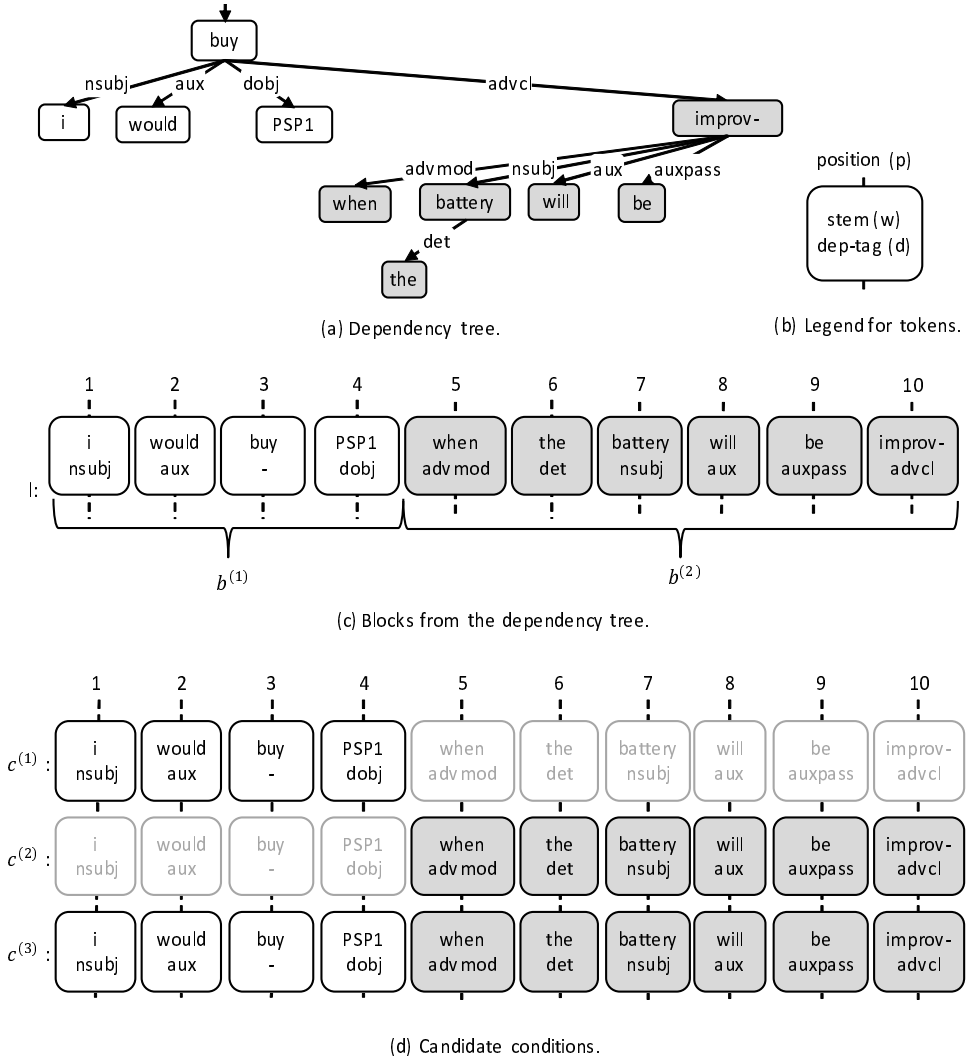


Figure 3.2: *Candidate ranking: sample candidate generation.*

that the tokens in a block are consecutive. For that reason, we split every block that contains holes. Finally, we generate the candidates as the sequences of tokens from all of the sequential combinations of the blocks.

Example 4: Figure §3.2.a shows the dependency tree of sentence "I would buy PSP1 when the battery will be improved". The nodes that correspond to the condition are highlighted in grey. Figure §3.2.b shows a legend that helps understand how a token $(w, \mathrm{d}, \mathrm{p})$ is mapped onto our graphical notation. Fig-

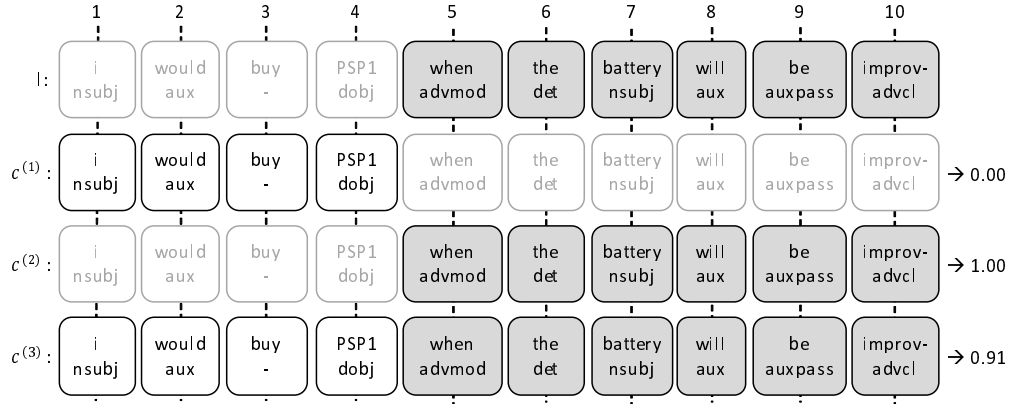


Figure 3.3: *Candidate ranking: sample matching scores.*

ure §3.2.c shows the blocks that are generated from the previous dependency tree. Figure §3.2.d shows the candidates that are generated from the previous blocks. Candidate $c^{(1)}$ is generated from block $b^{(1)}$, candidate $c^{(2)}$ is generated from block $b^{(2)}$, and candidate $c^{(3)}$, which corresponds to the whole sentence, is generated from blocks $b^{(1)}$ and $b^{(2)}$. □

3.2.3 Computing matching scores

Our second ancillary method is $\mathtt{computeScore}$, which takes a candidate c and a set of labels L as input and returns its corresponding score. A naive approach would simply return 0.00 if c does not exactly match any of the labels in L and 1.00 otherwise, but that is too crisp. An approach in which a candidate gets a score in range $[0.00, 1.00]$ better captures the chances that it is an actual condition.

Our idea was to use an approach that is based on the well-known F_1 score in order to balance the precision and the recall of a candidate. The F_1 score is computed as $\frac{2\,tp}{(tp+fp)+(tp+fn)}$, where tp, fp, and fn denote, respectively, the number of true positives, false positives, and false negatives. Given a candidate c and a label l, it makes sense to interpret the tokens that they have in common as true positive tokens, the tokens in c that are not in l as false positive tokens, and the tokens in l that are not in c as false negative tokens. We also realised that the first few tokens in a condition typically provide an anchor that characterises it by means of a connective. So, we decided to measure the matching between a candidate and a label as follows:

$$match(c, l) = \sum_{i=1}^{|l|} \left\{ \begin{array}{ll} 1/i & \text{if } l_i \in c \\ 0 & \text{otherwise} \end{array} \right\}$$

Simply put: let l_i denote the i-th token in the label ($i = 1 \,..\, |l|$); if l_i is in the candidate, we then add $1/i$ to the score and zero otherwise. This way, the first few tokens in the label contribute much more to the score than the remaining ones. That is, given a candidate c and a label l, $match(c, l)$ may be interpreted as a measure of the number of true positive tokens in c.

Given the previous definition, the maximum matching for a candidate or a label x is defined as follows:

$$match^*(x) = \sum_{i=1}^{|x|} 1/i$$

Realise that given a candidate c, $match^*(c)$ is a measure of the number of true positive tokens (the tokens that belong to both the candidate and the label) plus the false positive tokens (the tokens that belong to the candidate, but not to the label); similarly, given a label l, $match^*(l)$ is a measure of the number of true positive tokens (the tokens that belong to both the label and the candidate) plus the number of false negative tokens (the tokens that belong to the label, but not to the candidate).

Our proposal to compute the matching score of candidate c with respect to the set of labels L is then as follows:

$$score(c, L) = \max_{l \in L} \frac{2\, match(c, l)}{match^*(c) + match^*(l)}$$

Note the similarity to the F_1 score since $match(c, l)$, $match^*(c)$, and $match^*(l)$ are measures of tp, $tp + fp$, and $tp + fn$, respectively.

Example 5: Figure §3.3 shows label l, which corresponds to the condition in our running example, and how the candidates match it. Candidate $c^{(1)}$ does not match any true positive tokens, but four false positive tokens and six false negative tokens, which results in a matching score of 0.00. Candidate $c^{(2)}$ matches the label perfectly, i.e., it matches six true positive tokens and no false positive or false negative token, which results in a matching score of 1.00. Candidate $c^{(3)}$ represents the whole input sentence, which obviously contains the label, i.e., six true positive tokens, but also four false positive tokens, which results in a matching score of 0.91. □

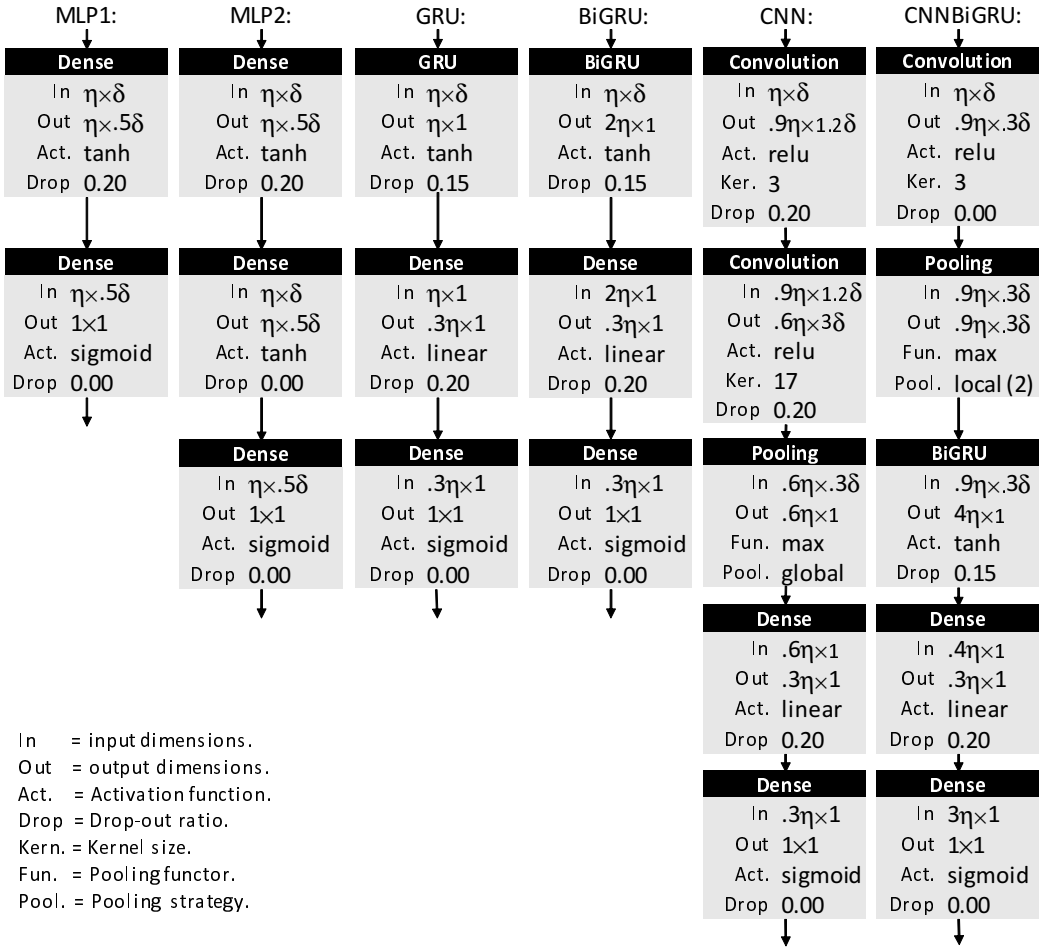


Figure 3.4: *Candidate ranking: neural network architectures.*

3.2.4 Learning a regressor

Our third ancillary method is learnRegressor, which takes a training set T as input and returns a regressor r.

Prior to learning a regressor, the candidates in the training set must be vectorised; note that it is necessary to put a limit to the maximum candidate length, to which we refer to as η, and that padding must be used when analysing shorter candidates.

Given a candidate of the form $\langle (w_i, d_i) \rangle_{i=1}^{n}$, we transform it into a sequence of the form $\langle \overline{w_i} \oplus \overline{d_i} \rangle_{i=1}^{\eta}$, where $\overline{w_i}$ denotes the word embedding of stem w using Word2vec, $\overline{d}$ denotes the vectorisation of dependency tag d using one-hot encoding, $\overline{w} \oplus \overline{d}$ the vector that results from catenating the previous ones, and η denotes the size of the longest possible condition; padding tokens need to be added if the original condition is shorter than η. Note that the vectorisation of a condition can then be interpreted as a matrix with η rows and δ columns, where δ denotes the dimensionality of the word embedding plus the dimensionality of the one-hot encoding.

Figure §3.4 summarises some of the neural network architectures that we have devised. We used the Mean Squared Error as the loss function and we trained them using Stochastic Gradient Descent Optimisation with batch size equal to 32. In order to prevent over-fitting as much as possible, we used some drop-out regularisations and early stopping when the loss did not improve significatively after 10 epochs. We did not apply a decay momentum because we observed that the loss always converges smoothly.

Our baseline architectures are the following: a shallow multi-layer perceptron (MLP1), which has an input layer and an output layer, and a deep multi-layer perceptron (MLP2), which has an input layer, a hidden layer, and an output layer. We devised a dozen more architectures with several combinations of Dense Networks, Gated Recurrent Units, Bi-directional Gated Recurrent Units, and Convolutionary Network layers and different configurations of activation functions, loss functions, regularisation parameters, and optimisation methods. In the sequel, we focus on the best architectures, namely: the one with gated recurrent units (GRU), the one with bi-directional gated recurrent units (BiGRU), the one with convolutional neural networks (CNN), and a hybrid approach that combines convolutional neural networks and bi-directional gated recurrent units (CNNBiGRU). The CNN network includes two convolution layers and a pooling layer. Our proposal is to use a convolution layer with a large number of filters in order to create a wide range of first-level features, but a smaller number of filters in the second convolution layer to obtain a more specific range of second-level features that combine the first ones. Finally, the pooling layer combines the previous deep features using a global maximum function as the global pooling strategy since our experiments prove that it performs better than others. The CNNBiGRU network uses a convolution with a number of filters similar to the input length, and then applies a local pooling that captures the most relevant features only. We then apply a BiGRU layer that takes the dependencies between tokens into account, from both the beginning to the end of the sentences and vice versa.

3.2.5 Removing overlaps

The fourth ancillary method is $removeOverlaps$, which takes a set of tuples of the form (c, z) as input, where c denotes a candidate and z its corresponding score, and filters those whose candidates overlap a candidate with a higher score. In other words, given the input set of tuples R, it computes the following subset:

$$\{(c, z) \in R \mid \nexists (c', z') : (c', z') \in R \wedge c' \cap c \neq \emptyset \wedge z' > z\}$$

We do not provide any additional details since this method can be implemented very straightforwardly.

Example 6: Assume that the threshold to select the best candidates is set to $\theta = 0.50$. In our running example, method $apply$ would return candidates $c^{(2)}$ and $c^{(3)}$ since they are the only whose scores exceed the threshold, cf. Figure §3.3. Note that both candidates overlap, so the one with the lowest score is filtered out. In this case, method $apply$ would then return candidate $c^{(2)}$ only, which, indeed, represents the condition in our running example. □

3.3 An encoder-decoder proposal

In this section, we describe our second proposal to mine conditions, which consists of an encoder-decoder model that is based on recurrent and bi-directional recurrent neural networks[†2].

Figures §3.5, §3.6, §3.7 and §3.8 sketch its architecture. The input are sentences are encoded as vectors of the form $(x_\lambda, x_{\lambda-1}, \ldots, x_1)$, where each x_i represents the corresponding lowercased, stemmed word (term) in the sentence ($i \in [1, \lambda], \lambda \geq 1$); note that λ must be set a priori to a large enough length and that padding must be used when analysing shorter sentences. The input vectors are first fed into an embedding layer that transforms each term into its corresponding word embedding vector E_i, which is assumed to preserve some similarity to the vectors that correspond to semantically similar terms ($E_i \in \mathbb{R}^t$, where t denotes the dimensionality of the word embedding). In order to improve efficiency without a significant impact on effectiveness, we replaced numbers, email addresses, URLs, and words whose

[†2]The implementation is available at https://github.com/FernanOrtega/encoder-decoder.

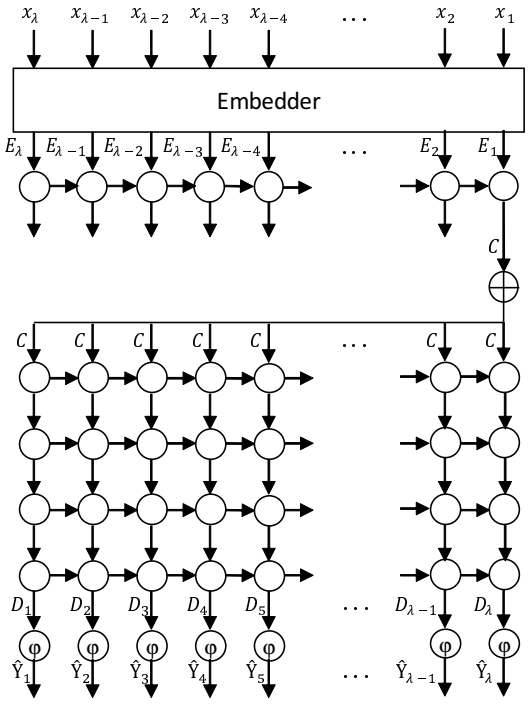


Figure 3.5: *Encoder-decoder: GRU-GRU model.*

frequency is equal or smaller than five by class words "NUMBER", "EMAIL", "URL", and "UNK", respectively. Note that the input vectors encode the reversed input sentences because Sutskever and others's [55] suggested that this approach works better with bi-directional recurrent neural networks and does not have a negative impact on regular recurrent neural networks. Furthermore, in order to prevent over-fitting, we used drop-out regularisations and early stopping when the loss did not improve significatively after 10 epochs. We used the Adam method with batch size 32 as the optimiser.

3.3.1 The encoder

We decided to implement the encoder using a single-layer neural network, for which we tried two alternatives, namely: a recurrent neural network (RNN) and a bi-directional recurrent neural network (BiRNN). The reason is that these networks are particularly well-suited to dealing with natural language because of their inherent ability to process varying-length inputs (even if they must be encoded using fixed-sized vectors with padding,

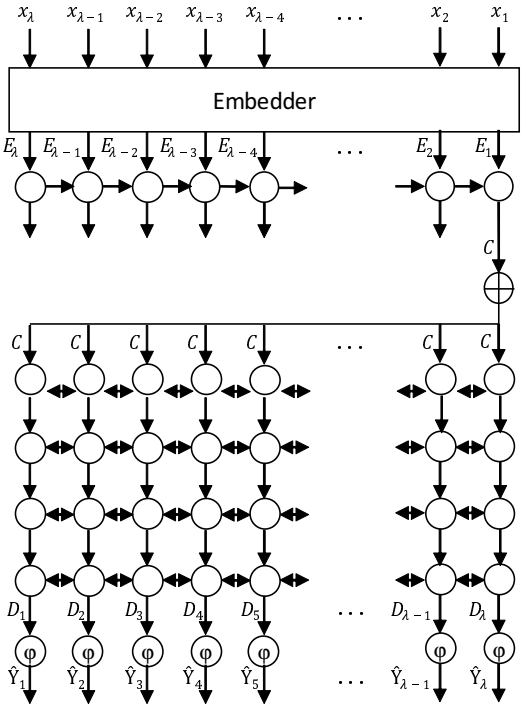


Figure 3.6: *Encoder-decoder: GRU-BiGRU model.*

like in our problem). The difference is that RNNs cannot take future elements of the input into account, whereas BiRNNs can.

Unfortunately, such networks suffer from the so-called exploding and vanishing gradient problems. These problems can be addressed by using long-short-term-memory units (LSTM) or gated recurrent units (GRU), which basically help control the data that is passed on to the next training epoch. Our decision was to use GRU units because they seem to be more efficient because they do not have a separate memory cell like LSTM units.

Hereinafter, we refer to the alternative in which we use an RNN with GRU units as GRU and to the alternative in which we use a BiRNN with GRU units as BiGRU. The encoder returns a context vector C that captures global semantic and syntactic features of the input sentences. In the case of the GRU alternative, it is computed as the output of the last GRU unit, that is, $C \in \mathbb{R}^t$; in the case of the BiGRU alternative, it is computed from the last right-to-left GRU unit and the last left-to-right GRU unit, that is, $C \in \mathbb{R}^{2t}$.

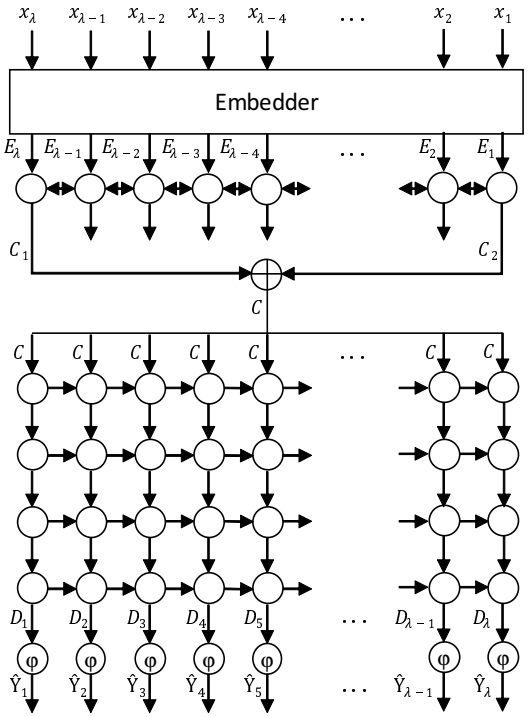


Figure 3.7: *Encoder-decoder: BiGRU-GRU model.*

3.3.2 The decoder

We decided to implement the decoder using a recurrent neural network with four layers since our preliminary experiments proved that this approach was better than using a single layer. We implemented two alternatives, namely: one in which we used recurrent neural networks with gated recurrent units (GRU) and another in which we used bi-directional recurrent neural networks with gated recurrent units (BiGRU).

The decoder gets a context vector from each recurrent unit of the first layer. Then, it computes an output vector D from each recurrent unit of the last layer. Since the number of recurrent units for each layer is λ, the components of the output vector D indicate whether the corresponding input term belongs to a condition or not using the well-known IOB tags, namely: I, which indicates that a term is inside a condition, O, which indicates that it is

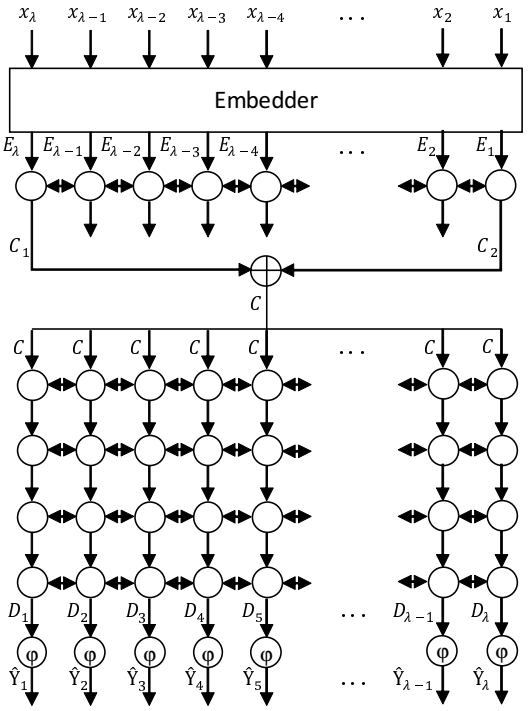


Figure 3.8: *Encoder-decoder: BiGRU-BiGRU model.*

outside a condition, and B, which indicates that it is the beginning of a condition. The individual components of the output vector are then passed onto a collection of perceptrons that compute the output of our system as follows:

$$\widehat{Y}_i = \varphi(W D_i + b)$$

where φ is an activation function, W is a weight matrix, D_i is the output of the decoder, and b is a bias vector. $\widehat{Y}_i$ represents the probability distribution of the IOB tags as 3-dimensional vectors. We decided to implement the activation function φ using either the Softmax function or the Sigmoid function, since the preliminary experiments that we carried out proved that other choices resulted in worse results.

To reconstruct the conditions from this output, we simply take the tag with the highest probability and then return all of the sub-sequences of words in the original sentence that start with a term with tag B that is followed by one, two, or more terms with tag I.

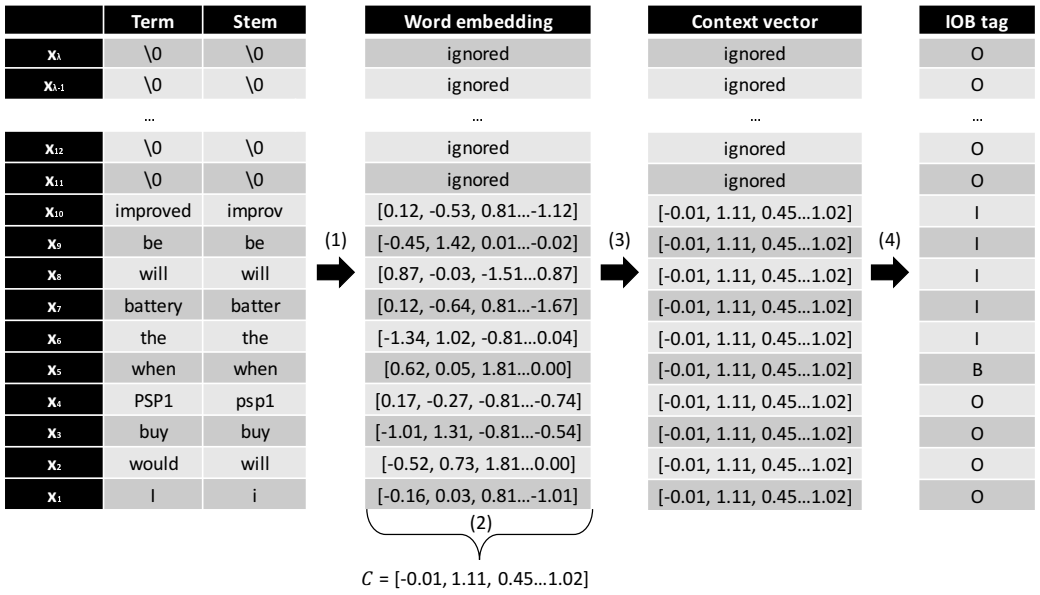


Figure 3.9: *Encoder-decoder: example.*

Example 7: Figure §3.9 shows our running example as an input of our encoder-decoder model. First, it converts the input tokens into its lower-cased stems. (We use "\0" as a padding symbol that is ignored across the entire network). Next, it computes their word embeddings (1). Now, the encoder network produces a vector C. It is then used as input for every first-layer unit of the decoder (3). Finally, the decoder returns the most likely IOB tag for each token. In our example, the condition is then "when the battery will be improved". □

3.4 Experimental analysis

In this section, we first describe our experimental setup, then comment on our results, and finally present our statistical analysis.

3.4.1 Experimental setup

We run our experiments on a virtual computer that was equipped with one Intel Xeon E5-2690 core at 2.60 GHz, 2 GiB of RAM, and an Nvidia Tesla K10 GPU accelerator with 2 GK-104 GPUs at 745 MHz with 3.5 GiB of RAM

each; the operating system on top of which we run our experiments was CentOS Linux 7.3.0.

We used the proposals by Chikersal and others [10] and Mausam and others [38] as baselines. The proposal by Nakayama and Fujii [41] was not taken into account because it is bound to the Japanese language and it is not clear how it can be extended to deal with other languages; neither could we find an implementation.

Regarding our candidate-ranking proposal, we evaluated our five alternatives using the following values for the threshold: $\theta = 0.25$, $\theta = 0.50$, or $\theta = 0.75$. For the sake of readability, we refer to them using their names and the threshold as subscripts, namely: MLP1_θ, MLP2_θ, GRU_θ, BiGRU_θ, CNN_θ, and CNNBiGRU_θ.

Regarding our encoder-decoder proposal, we evaluated the eight alternatives that result from combining the two alternatives to implement the encoder (GRU or BiGRU), with the two alternatives to implement the decoder (GRU or BiGRU), using the following values for the activation function: $\varphi = \texttt{soft}$, to mean the Softmax function, and $\varphi = \texttt{sig}$, to mean the Sigmoid function. For the sake of readability we refer to them using the names of the alternatives and the activation function as subscripts, namely: GRU-GRU_φ, GRU-BiGRU_φ, BiGRU-GRU_φ, and $\text{BiGRU-BiGRU}_\varphi$.

3.4.2 Experimental results

We evaluated the baselines and our alternatives on our dataset using 4-fold cross-validation. We measured the standard performance measures, namely: precision, recall, and the F_1 score. Table §3.1 presents the results of our experiments. The conclusion is that the state-of-the-art baselines can attain relatively good precision; Mausam and others's [38] proposal attains a recall that is similar to its precision, but Chikersal and others's [10] proposal falls short regarding recall. Our baseline performs slightly worse than our alternatives in most situations. Most of our alternatives beat the baselines regarding recall because they can learn complex patterns that a person cannot easily spot. Note that the improvement regarding recall is enough for the F_1 score to improve the baselines.

3.4.3 Statistical analysis

To make a decision regarding which of the alternatives performs the best according to their F_1 score, we used a stratified strategy that builds on Hommel's statistical comparison test at the standard significance level ($\alpha = 0.05$).

Proposal	English			Spanish			French			Italian		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Mausam et al.	0.63	0.61	0.62	0.67	0.53	0.59	0.65	0.67	0.66	0.49	0.50	0.49
Chikersal et al.	0.80	0.46	0.59	0.80	0.44	0.57	0.88	0.58	0.70	0.50	0.48	0.49

(a) Baselines.

Alternatives	English			Spanish			French			Italian		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
$MLP1_{0.25}$	0.51	0.80	0.62	0.48	0.81	0.61	0.66	0.87	0.75	0.57	0.75	0.65
$MLP1_{0.50}$	0.58	0.69	0.63	0.56	0.66	0.60	0.71	0.85	0.78	0.63	0.64	0.63
$MLP1_{0.75}$	0.60	0.58	0.59	0.59	0.53	0.56	0.73	0.76	0.74	0.65	0.54	0.59
$MLP2_{0.25}$	0.49	0.82	0.61	0.44	0.84	0.58	0.69	0.87	0.77	0.58	0.80	0.67
$MLP2_{0.50}$	0.56	0.74	0.63	0.52	0.75	0.62	0.75	0.86	0.80	0.64	0.68	0.66
$MLP2_{0.75}$	0.61	0.64	0.63	0.59	0.65	0.61	0.77	0.83	0.80	0.72	0.59	0.65
$GRU_{0.25}$	0.55	0.80	0.65	0.53	0.82	0.64	0.74	0.88	0.80	0.69	0.55	0.62
$GRU_{0.50}$	0.58	0.74	0.65	0.57	0.77	0.66	0.81	0.78	0.79	0.74	0.55	0.63
$GRU_{0.75}$	0.61	0.66	0.63	0.63	0.67	0.65	0.75	0.85	0.80	1.00	0.48	0.65
$BiGRU_{0.25}$	0.55	0.80	0.65	0.53	0.82	0.64	0.74	0.88	0.80	0.93	0.48	0.63
$BiGRU_{0.50}$	0.59	0.74	0.65	0.59	0.76	0.66	0.80	0.67	0.73	0.87	0.48	0.62
$BiGRU_{0.75}$	0.62	0.68	0.65	0.63	0.66	0.64	0.76	0.84	0.79	1.00	0.48	0.65
$CNN_{0.25}$	0.62	0.68	0.65	0.54	0.78	0.64	1.00	0.48	0.65	1.00	0.48	0.65
$CNN_{0.50}$	0.92	0.49	0.64	0.65	0.68	0.67	1.00	0.48	0.65	1.00	0.48	0.65
$CNN_{0.75}$	1.00	0.44	0.61	0.71	0.58	0.64	1.00	0.48	0.65	1.00	0.48	0.65
$CNNBiGRU_{0.25}$	0.55	0.78	0.65	0.53	0.78	0.64	0.76	0.88	0.81	0.82	0.62	0.71
$CNNBiGRU_{0.50}$	0.59	0.73	0.65	0.60	0.73	0.66	0.77	0.88	0.82	0.83	0.58	0.68
$CNNBiGRU_{0.75}$	0.62	0.67	0.64	0.65	0.65	0.65	0.78	0.87	0.82	0.77	0.64	0.70

(b) Candidate-ranking alternatives.

Alternatives	English			Spanish			French			Italian		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
GRU-GRU_{sig}	0.65	0.62	0.63	0.61	0.56	0.59	0.87	0.57	0.69	0.58	0.59	0.59
GRU-GRU_{soft}	0.68	0.61	0.64	0.63	0.57	0.60	0.81	0.77	0.79	0.65	0.62	0.63
GRU-$BiGRU_{sig}$	0.65	0.63	0.64	0.67	0.57	0.62	0.82	0.77	0.79	0.63	0.57	0.60
GRU-$BiGRU_{soft}$	0.68	0.62	0.65	0.64	0.58	0.61	0.81	0.79	0.80	0.61	0.60	0.61
BiGRU-GRU_{sig}	0.63	0.61	0.62	0.60	0.56	0.58	0.86	0.65	0.74	0.91	0.50	0.64
BiGRU-GRU_{soft}	0.65	0.62	0.63	0.65	0.56	0.60	0.80	0.78	0.79	0.65	0.59	0.62
BiGRU-$BiGRU_{sig}$	0.67	0.62	0.64	0.63	0.59	0.61	0.82	0.77	0.80	0.60	0.57	0.58
BiGRU-$BiGRU_{soft}$	0.69	0.59	0.64	0.64	0.58	0.61	0.81	0.77	0.79	0.61	0.57	0.59

(d) Encoder-decoder alternatives.

Table 3.1: *Experimental analysis: results.*

Alternative	Rank	z	p-value
$CNNBiGRU_{0.25}$	2.38	-	-
$BiGRU_{0.25}$	2.75	0.57	1.00
$GRU_{0.25}$	3.13	1.13	1.00
$CNN_{0.25}$	3.69	1.98	0.33
$MLP2_{0.25}$	4.31	2.93	0.03
$MLP1_{0.25}$	4.75	3.59	0.00

(a) Comparison with $\theta = 0.25$.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.50}$	2.56	-	-
$GRU_{0.50}$	2.88	0.47	1.00
$BiGRU_{0.50}$	3.25	1.04	1.00
$MLP2_{0.50}$	3.56	1.51	0.78
$CNN_{0.50}$	3.81	1.89	0.59
$MLP1_{0.50}$	4.94	3.59	0.00

(b) Comparison with $\theta = 0.50$.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.75}$	1.56	-	-
$BiGRU_{0.75}$	2.72	1.75	0.26
$GRU_{0.75}$	2.78	1.84	0.26
$MLP2_{0.75}$	3.81	3.40	0.00
$CNN_{0.75}$	4.38	4.25	0.00
$MLP1_{0.75}$	5.75	6.33	0.00

(c) Comparison with $\theta = 0.75$.

Alternative	Rank	z	p-value
$CNNBiGRU_{0.50}$	1.75	-	-
$CNNBiGRU_{0.75}$	1.88	0.35	0.72
$CNNBiGRU_{0.25}$	2.38	1.77	0.23

(d) Comparison of best alternatives.

Table 3.2: *Statistical analysis: candidate-ranking alternatives.*

Alternative	Rank	z	p-value
GRU-$BiGRU_{soft}$	2.00	-	-
GRU-GRU_{soft}	2.56	1.23	0.60
BiGRU-$BiGRU_{soft}$	2.69	1.51	0.60
BiGRU-GRU_{soft}	2.75	1.61	0.60

(a) Comparison with φ = *soft*.

Alternative	Rank	z	p-value
GRU-$BiGRU_{sig}$	1.75	-	-
BiGRU-$BiGRU_{sig}$	2.19	0.96	0.68
BiGRU-GRU_{sig}	3.00	2.74	0.02
GRU-GRU_{sig}	3.06	2.88	0.02

(b) Comparison with φ = *sig*.

Alternative	Rank	z	p-value
GRU-$BiGRU_{soft}$	1.50	-	-
GRU-$BiGRU_{sig}$	1.50	0.00	1.00

(c) Comparison of best alternatives.

Table 3.3: *Statistical analysis: encoder-decoder alternatives.*

The test computes the empirical ranking of every alternative; it then compares the best-ranked ones to the others by computing a z statistic and its corresponding p-value, which must be compared to the α significance level as follows: if the it is smaller than or equal to α, then the conclusion is that there is enough evidence in the experimental data to support the hypothesis that the difference between two alternatives is statistically signifi-

Proposal	Rank	z	p-value
CNNBiGRU$_{0.50}$	1.19	-	-
GRU-BiGRU$_{soft}$	2.06	1.92	0.11
Mausam et al.	3.06	4.11	0.00
Chikersal et al.	3.69	5.48	0.00

Table 3.4: *Statistical analysis: comparison to the baselines.*

cant; otherwise, the conclusion is that the experimental data cannot sustain that there is a statistically significant difference.

In Tables §3.2.a, §3.2.b, and §3.2.c, we report on the results of the statistical analysis regarding our alternatives. The best-ranked alternative is CNNBiGRU in every case, independently from the value of the threshold. Note that the difference in F_1 score is not significant with respect to the BiGRU, the GRU, or the CNN alternatives when $\theta = 0.25$, it is not significant with respect to the GRU, the BiGRU, the MLP2, or the CNN alternatives when $\theta = 0.50$, and it is not significant with respect to the BiGRU or the GRU alternatives when $\theta = 0.75$. Our conclusion is that the CNNBiGRU alternative is the best one in terms of F_1 score. In Table §3.2.d, we report on the results of comparing the CNNBiGRU alternative with the three values of θ. Note that the p-value is greater than the standard significance level, which means that the differences in F_1 score are not significant. Thus, we select the CNNBiGRU alternative with $\theta = 0.50$ as the best candidate-ranking alternative.

In Tables §3.3.a and §3.3.b, we report on the results of our statistical analysis regarding our encoder-decoder alternatives. The best-ranked alternative is GRU-BiGRU in every case, independently from the activation function. Note that the difference in F_1 score is not significant with regard to the other alternatives when $\varphi = soft$ and that it is not significant with regard to BiGRU-BiGRU when $\varphi = sig$. Our conclusion is that the GRU-BiGRU alternative can be considered the best of our encoder-decoder alternatives in terms of F_1 score. In Table §3.3.c, we report on the results of comparing the GRU-BiGRU alternatives with the two values of φ. Note that the p-value is greater than the standard significance level, which means that the differences in F_1 score are not significant. Thus, we select the GRU-BiGRU alternative with $\varphi = soft$ as the best encoder-decoder alternative.

In Table §3.4, we report on the results of comparing our best alternative to the baselines. The best-ranked alternative is the candidate-ranking

CNNBiGRU alternative with $\theta = 0.50$. The difference with our recurrent neural network baseline in terms of F_1 score is not significant from a statistical point of view since the p-value is greater than the standard significance level. Note, however, that the difference is very significant with respect to the baselines since the p-value is nearly zero. Thus, we select the CNNBiGRU alternative with $\theta = 0.50$ as our best proposal to mine conditions.

3.5 Summary

In this chapter, we have presented two proposals to mine conditions and our experimental analysis. The first proposal relies on a deep-learning regression approach to rank a set of candidates that are computed from a dependency tree. The second proposal relies on an encoder-decoder approach to perform sequence labelling. They do not rely on user-defined patterns, they do not require any specific-purpose dictionaries, taxonomies or heuristics, they can mine conditions in both factual and opinion sentences, and they are not bound to a specific language. Furthermore, we presented an experimental analysis that shows that our proposals can beat the state-of-art in terms of F_1. We confirmed this claim by means of statistical tests.

Chapter 4

Norito: a dataset of conditions

Norito is the dataset that we assembled to evaluate condition miners. The rest of the chapter is organised as follows: Section §4.1 introduces our dataset; Section §4.2 describes its main features; Section §4.3 presents our analysis of connective distribution; Section §4.4 presents our analysis of condition similarity; finally, Section §4.5 summarises our conclusions.

4.1 Introduction

In this chapter, we present Norito, which is a publicly available dataset[†1] to evaluate condition miners. It consists of 4.7M sentences on 16 common topics in English, Spanish, French, and Italian. As of the time of writing this dissertation, we have labelled roughly 45 000 sentences since we intend Norito to become a community effort by the researchers who are interested in condition mining. We also analyse the connective distribution and the similarity between conditions[†2].

The Web provides tones of data that can be used to create new datasets. For instance, it is easy to assemble a dataset to evaluate a statistical model that forecasts the weather in a specific area since there are a variety of web sites that provide historic data regarding outlooks, temperatures, precipitations, or wind speeds. The problem arises when the target class or function that we wish to compute or approximate is not publicly available.

This problem frequently appears when assembling new datasets to evaluate text mining tasks, for instance, condition mining. It is possible to gather many reviews from web sites like Amazon.com, shopping.google.com, kelkoo.com, or booking.com, but, unfortunately, it is unknown where the conditions are, if any. Neither works it the attempt to infer them from other text mining tasks like textual entitlement, question answering, part-of-speech tagging, or dependency parsing since they do not take conditions into account. Furthermore, none of the existing condition miners provides a public evaluation dataset. This was clearly our motivation to work on Norito.

4.2 Description

As of the time of writing this dissertation, our dataset consists of 4 671 533 sentences in English, Spanish, French, and Italian that were gathered from Ciao.com between April 2017 and May 2017. They were classified into 16 topics according to their sources, namely: adults, baby care, beauty, books, cameras, computers, films, headsets, hotels, music, ovens, pets, phones, TV sets, and video games.

[†1] The dataset is available at https://www.kaggle.com/fogallego/reviews-with-conditions.

[†2] The analyses were performed with a Kaggle kernel that is available at https://www.kaggle.com/fogallego/dataset-analysis.

English					
Domain	#Conds	#Sents	#Lab	#SwC	%SwC
adults	325	2 164	1 007	290	28.80%
babycare	26	28 282	1 008	22	2.18%
beauty	19	109 348	1 007	18	1.79%
books	3	49 653	1 002	3	0.30%
cameras	10	8 574	1 010	10	0.99%
films	3	45 547	1 024	3	0.29%
hotels	231	21 908	1 002	214	21.36%
music	113	24 218	1 000	106	10.60%
pets	284	34 878	1 002	251	25.05%
phones	172	2 052	1 027	162	15.77%
tvsets	175	3 537	1 007	159	15.79%
video games	176	111 439	1 006	159	15.81%

French					
Domain	#Conds	#Sents	#Lab	#SwC	%SwC
adults	42	7 586	1 012	42	4.15%
babycare	16	47 972	1 023	16	1.56%
beauty	32	74 739	1 004	30	2.99%
books	42	94 141	1 002	40	3.99%
cameras	61	3 641	1 008	54	5.36%
computers	59	2 792	1 006	56	5.57%
films	59	136 068	1 010	54	5.35%
hotels	46	12 941	1 000	46	4.60%
music	23	25 109	1 004	21	2.09%
pets	49	13 991	1 003	45	4.49%
phones	62	7 972	1 001	61	6.09%
tvsets	39	6 410	1 009	37	3.67%
video games	66	77 457	1 007	61	6.06%

Spanish					
Domain	#Conds	#Sents	#Lab	#SwC	%SwC
books	597	1 013 746	2 005	502	25.04%
computers	359	102 021	1 006	303	30.12%
films	210	209 810	1 004	185	18.43%
headsets	243	10 222	1 022	195	19.08%
hotels	173	317 177	1 028	162	15.76%
music	114	417 456	1 033	100	9.68%
oven	177	17 197	1 011	152	15.03%
pets	130	66 845	1 017	114	11.21%
phones	146	547 921	1 010	132	13.07%
tvsets	132	108 287	1 005	108	10.75%
video games	158	537 921	1 008	140	13.89%

Italian					
Domain	#Conds	#Sents	#Lab	#SwC	%SwC
adults	40	5 068	464	40	8.62%
books	5	5 890	554	5	0.90%
computers	2	3 403	430	2	0.47%
films	2	267 454	1 236	2	0.16%
headsets	139	4 703	1 020	129	12.65%
hotels	37	35 915	895	35	3.91%
music	77	46 029	1 170	68	5.81%
oven	48	2 049	1 041	44	4.23%

Table 4.1: *Description: summary of our dataset.*

We developed an application to label the dataset[†3]. There is an administrator role that can create labelling tasks and assign them to specific labellers; the administrator may also set a reward per sentence, which we have found a motivation for students. A labeller performs the task as follows: for each sentence, he or she must spot its conditions, if any; for each condition, he or she must highlight the piece of text that contains the connective; the application computes the indices automatically.

Our approach to label the sentences is to obtain as many labelled sentences as possible instead of labelling the same sentence several times in an

[†3]The application is available at http://conditionslabelling.fernanortega.es. You can log in as "labellerTest" / "M1entr45".

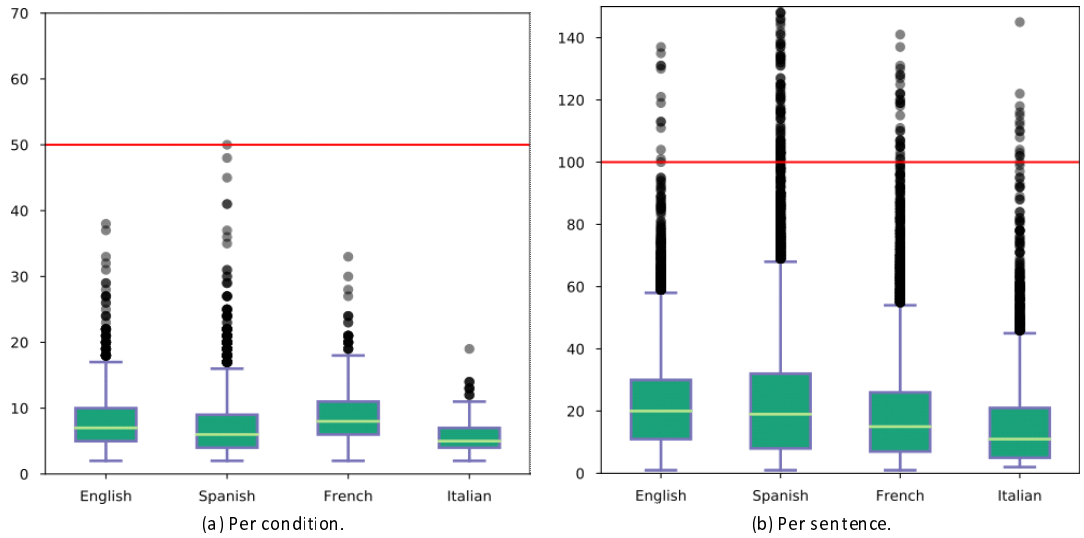


Figure 4.1: *Description: typical numbers of words.*

attempt to measure the inter-labeller agreement. We tried the following approach: first, we used the well-known IOB tags to label each individual token as a token inside a condition (tag I), outside it (tag O), or at the beginning of a condition (tag B); then, we computed some agreement measures on a per-token basis. Unfortunately, this approach did not work well because the classes are heavily unbalanced and it does not take overlapping into account. This motivates the need for further research on this issue, but this clearly falls out of the scope of this dissertation. The key is that current agreement measures assume that there are a number of data that the labellers must annotate using some pre-defined classes; in our problem, the labellers must spot the conditions, that is: there are not any pre-defined data to be labelled, they have to be found by the labellers.

In Table §4.1, we provide a summary of our dataset. The columns of the table denote the language (Lang), the domain (Domain), the number of conditions found (#Conds), the number of sentences (#Sents), the number of sentences that we have labelled as of the time of writing this article (#Lab), the number of sentences that contain at least one condition (#SwC), and the corresponding percentage (%SwC).

In Figures §4.1.a and §4.1.b, we present a box and whisker plot that represents the number of words per condition and sentence in our dataset. Note

Lang	Connective	Freq
en	if	349
	when	239
	for	100
	after	100
	before	64
	while	60

(a.1) Top.

Lang	Connective	Freq
en	every time	2
	over	2
	regardless of	2
	regardless	2
	prior	2
	but in	2

(a.2) 75-th percentile.

Lang	Connective	Freq
es	si	600
	cuando	235
	para	221
	en	119
	al	105
	a	94

(b.1) Top.

Lang	Connective	Freq
es	a medida que	3
	además de	3
	porque si	3
	incluso cuando	3
	a causa de	3
	viendo	3

(b.2) 75-th percentile.

Lang	Connective	Freq
fr	si	384
	même si	63
	pour	48
	mais si	15
	en cas de	11
	quand	9

(c.1) Top.

Lang	Connective	Freq
fr	sauf si	2
	avec	2
	même s'	2
	si bien qu'	1
	si on ne	1
	quant	1

(c.2) 75-th percentile.

Lang	Connective	Freq
it	se	170
	quando	108
	per	7
	anche	7
	in cui	4
	da quando	3

(d.1) Top.

Lang	Connective	Freq
it	el momento in cu	2
	in	2
	in cerca	2
	da	2
	soprattutto	2
	ogni volta	2

(d.2) 75-th percentile.

Table 4.2: *Connective distribution: connective samples.*

that 50 words for conditions and 100 words for sentences are sensible upper limits to their maximum lengths, which helps set an appropriate dimensionality regarding many problems, e.g., transforming the sentence into a vector to feed a neural network. Note that these thresholds do not miss any conditions, but some extremely long outlier sentences.

4.3 Connective distribution

Table §4.2 shows the frequency of the five most frequent connectives and the five around the 75-th percentile. Note that there are a few usual connectives that have high frequencies, whereas the others have frequencies that are very low.

The previous table makes it intuitively clear that the distribution of connectives might be a long-tail distribution. To confirm it, it is necessary to compare it to the Power-Law and the Log-Normal distributions, which are the standard long-tail distributions, and to the Exponential distribution, which is not long-tail by definition.

In Figure §4.2, we plot the Complementary Cumulative Distribution Functions (CCDF) of the previous distributions. Realise that the Power-Law distribution and the Log-Normal distribution are very similar to the connective distribution, whereas the Exponential one is not. We conducted

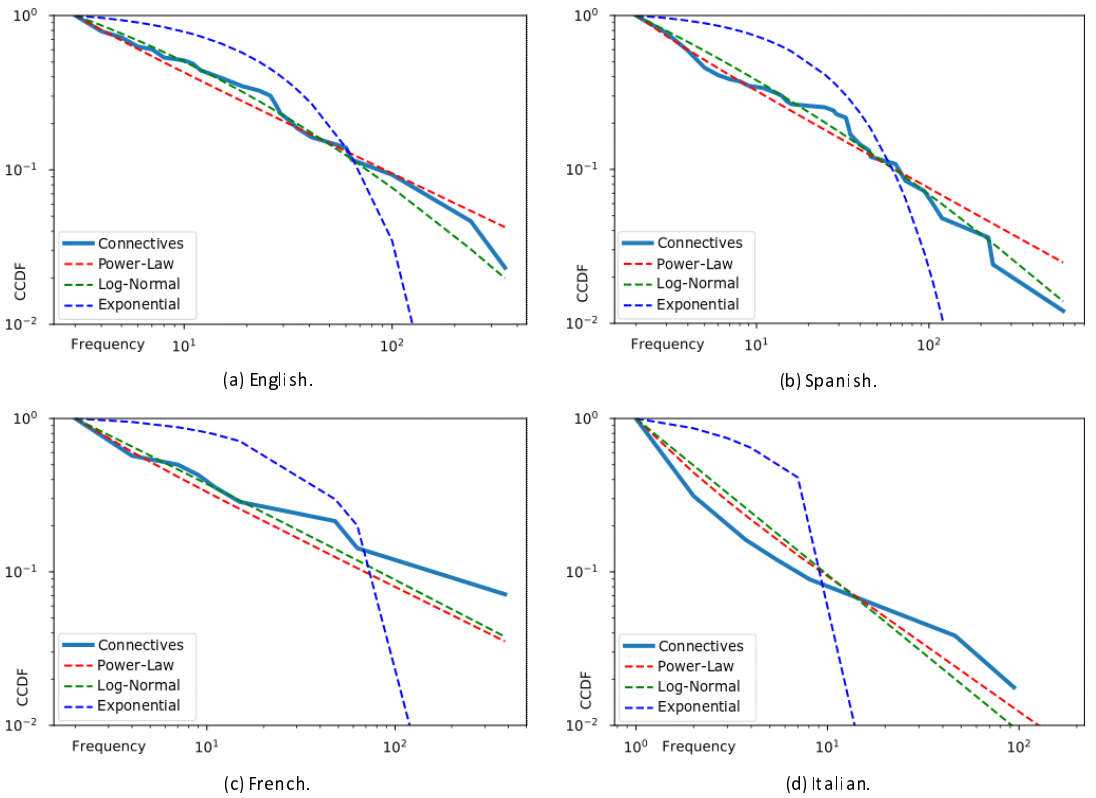


Figure 4.2: *Connective distribution: analysis of distribution.*

a Likelihood Ratio Test to check the previous idea statistically. The results of the test are shown in Table §4.3: for each language in our dataset, we compared the distribution of connectives to every two pairs of the previous standard distributions and computed R as the log likelihood ratio and the corresponding p-value.

Independently from the language, the comparison to the Power-Law distribution and the Log-Normal distribution returns p-values that are greater than the standard significance level $\alpha = 0.05$, which indicates that there is not enough empirical evidence to conclude that the connective distribution is significantly different from a Power-Law or a Log-Normal distribution; note that the comparisons to the Power-Law and the Exponential distribution or Log-Normal and the Exponential distribution return a positive log likelihood ratio with a p-value that is smaller than the standard significance level, which indicates that there is enough empirical evidence to conclude that the connective

Lang	$Dist_1$	$Dist_2$	R	p-value
en	Power-Law	Log-Normal	-1.14	0.33
	Power-Law	Exponential	19.32	0.04
	Log-Normal	Exponential	20.47	0.01
es	Power-Law	Log-Normal	-0.77	0.31
	Power-Law	Exponential	64.32	0.00
	Log-Normal	Exponential	65.09	0.00
fr	Power-Law	Log-Normal	-0.06	0.46
	Power-Law	Exponential	15.40	0.01
	Log-Normal	Exponential	15.46	0.01
it	Power-Law	Log-Normal	-7.00	0.11
	Power-Law	Exponential	70.02	0.00
	Log-Normal	Exponential	77.02	0.00

Table 4.3: *Connective distribution: fitting the connective distribution.*

distribution is similar to the Power-Law or the Log-Normal distributions, but very different from the Exponential distribution.

The conclusion is that there is enough statistical evidence to consider the connective distribution a long-tail distribution. Simply put, relying on a collection of handcrafted patterns will typically fall short in terms of recall because there are too many ways to introduce conditions, which clearly argues for a machine-learning solution.

4.4 Condition similarity

We have also analysed the similarity of the conditions that are available in the sentences of our dataset. Our goal was to check if there are groups of conditions that are similar enough to be modelled using some common features, e.g., verbs, adverbs, or prepositions. To carry this analysis out, we changed every word into lowercase and then computed a vectorisation of each condition as follows: each component of the vectors corresponds to a different word and represents its tf-idf frequency in the condition being vectorised. The English vectorisation has 2 311 words, the Spanish vectorisation has 3 796 words, the French vectorisation has 1 386 words, and the Italian vectorisation has 658 words.

In order to visualise them, we performed a dimensionality reduction by means of two well-known techniques, namely: Isomap and truncated single

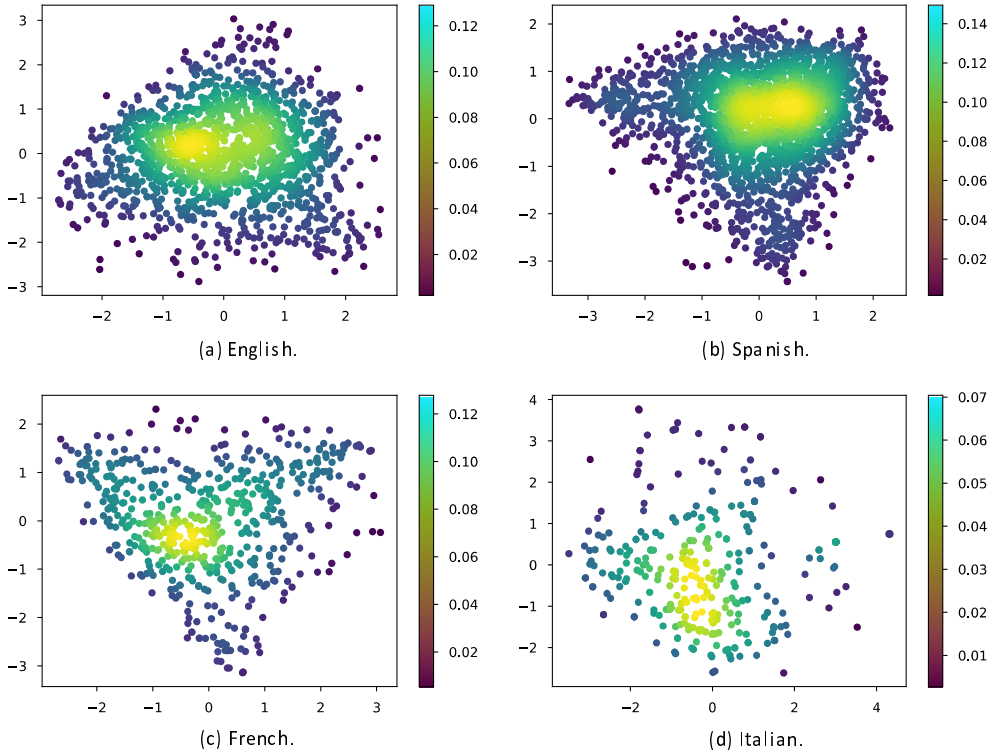


Figure 4.3: *Condition similarity: Isomap projections.*

value decomposition (TSVD). Furthermore, we computed the Gaussian Kernel Density Estimation to better visualise the density of samples with Scott's Rule to compute the estimator bandwidth. In Figures §4.3 and §4.4, we show a graphical representation of the Isomap and the TSVD projections of the conditions, respectively. The hues range from bright yellow, which represents the highest densities (conditions that are very similar to each other), to dark blue, which represents the lowest densities (conditions that are not similar to each other). It is not difficult to realise that the conditions are organised as follows: there is one small group with high density, a larger group with average density, and a very large group with low density.

As a conclusion, it must not be difficult for a person to learn a rule to mine instances of the first group since there are many examples available and they seem very similar to each other; but it must not be that easy to deal with the many other conditions since they are not similar to each other.

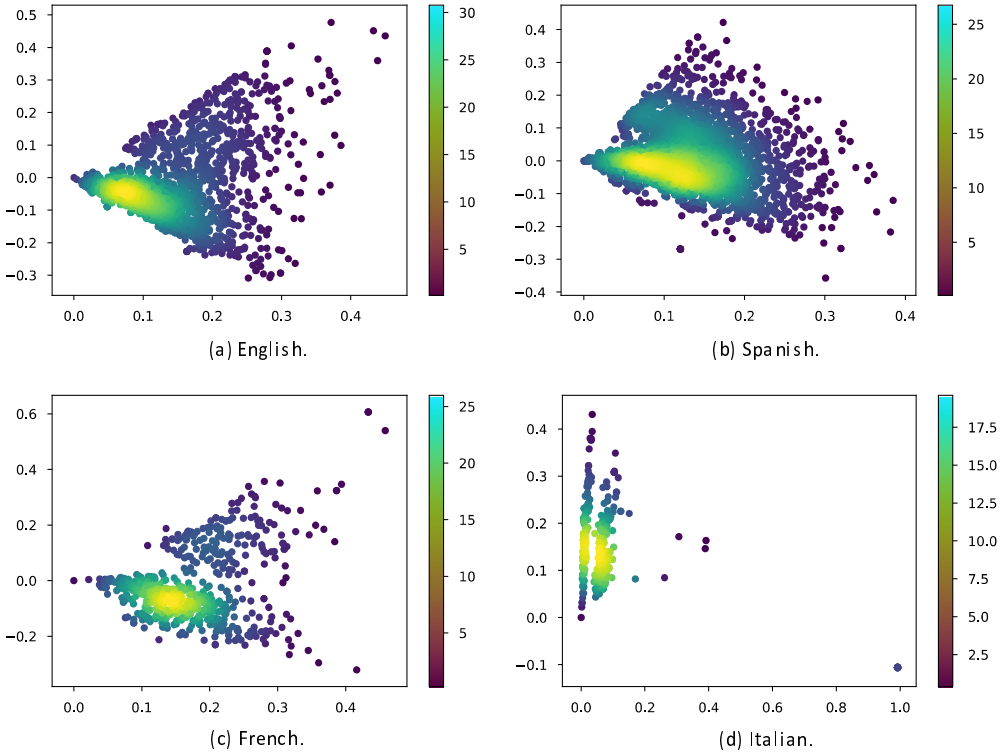


Figure 4.4: *Condition similarity: TSVD projections.*

This also argues for a machine-learning solution.

4.5 Summary

In this chapter, we have presented our publicly available dataset of conditions. We performed a detailed analysis of our dataset in which we confirm that the distribution of connectives follows a long-tail distribution in which there are many similar conditions, but too many dissimilar conditions for a person to spot them all and handcraft a set of rules that can mine them.

Chapter 5

Conclusions

The Web has become one of the most valuable sources of data for companies. Many research fields that are related to text mining have sprouted or have grown quickly around web data in recent years. The goal of text mining is to process the text in a collection of documents to produce structured information that can be used to feed business processes. There is a growing interest in taking advantage of the information that customer reviews convey. Specifically, sentiment analysis has grown up as one of the most promising text mining tasks in terms of market sales. Furthermore, we estimate that about 10% of the sentences in our dataset contain conditions, which makes it clear that they are important enough to be taken into account. Mining conditions is a text mining process that seeks to spot the pieces of a review that contain conditions in an attempt to improve the interpretation of the results of other text mining tasks.

In this dissertation, we have studied the problem of devising an aspect-based sentiment analysis system that integrates a condition miner, which has resulted in a system called Torii. It relies on a micro-service architecture to fulfil the requirements of scalability, reliability, and flexibility. It integrates a deep neural network to mine conditions, a frequentist approach to discover aspects, and a lexicon-based approach to summarise sentiment. It also provides a user interface to interact with the system. Torii has significatively improved the services that Opileak provides to their customers. In future, we would like to explore how recommenders or entity-relation extractors may benefit from mining conditions.

We have explored several approaches to mine conditions and we have devised Kami, which consists of two proposals to mine conditions. The first one relies on a deep-learning regression approach to rank a set of candidates that are computed from the dependency tree of the input sentence. The

second one relies on a deep neural encoder-decoder approach that performs sequence labelling on the input sentence. Our proposals do not require to provide any user-defined patterns, do not require any specific-purpose dictionaries, taxonomies, or heuristics, can mine conditions in both factual and opinion sentences, and rely on readily-available components (a stemmer, a word embedder, and a deep learner). Our results confirm that they can beat the others in terms of F_1 score. In future, we would like to explore transfer learning. An approach might be to use machine translation engines to translate the sentences in our dataset to another language and then use the results as the training set. This idea might help avoid over-fitting due to the inherent noise that it introduces.

Finally, we have realised that it does not exist any publicly available datasets to evaluate condition miners. This motivated us to assemble Norito, which provides 4.7M sentences in English, Spanish, French, and Italian that were classified into 16 categories according to their sources. We have also analysed it regarding how connectives are distributed and how similar the conditions are. In future, we would like to extend our dataset to Latin languages like Portuguese or Catalan, Germanic languages like German or Dutch, and Asian languages like Japanese or Chinese. Furthermore, we found that current labelling agreement measures do not work well in our context basically because the set of labels does not pre-exist, but must be spotted by the labellers; straightforward applications of the existing measures at the token level do not work well because their IOB tags are heavily unbalanced and overlapping is very common.

Summing up, assuming that our research hypothesis is accepted, we think that we have sufficiently proven our thesis. We hope that our results can effectively help companies make better decisions from the reviews that are available on the Web. We also think that we have opened up an interesting research path that may soon lead to new research results.

Bibliography

[1] A. M. Abirami and A. Askarunisa. *Sentiment analysis model to emphasize the impact of online reviews in healthcare industry*. *Online Information Review*, 41(4):471–486, 2017.

[2] C. C. Aggarwal. *Instance-based learning: A survey*. In *Data Classification: Algorithms and Applications*, pages 157–186. Springer, 2014.

[3] C. C. Aggarwal and C. Zhai. *An introduction to text mining*. Springer, 2012.

[4] G. Antoniou, S. Batsakis, R. Mutharaju, J. Z. Pan, G. Qi, I. Tachmazidis, J. Urbani, and Z. Zhou. *A survey of large-scale reasoning on the Web of Data*. *Knowledge Eng. Review*, 33:e21, 2018.

[5] B. Athiwaratkun, A. G. Wilson, and A. Anandkumar. *Probabilistic FastText for multi-sense word embeddings*. In *ACL*, pages 1–11, 2018.

[6] M. Beccue and A. Kaul. *Emotion recognition and sentiment analysis*, 2018. Available at https://www.tractica.com/research/emotion-recognition.

[7] Y. Bengio, P. Y. Simard, and P. Frasconi. *Learning long-term dependencies with gradient descent is difficult*. *IEEE Trans. Neural Networks*, 5 (2):157–166, 1994.

[8] S. Bird. *NLTK: the natural language toolkit*. In *ACL*, pages 69–72, 2006.

[9] L. Breiman. *Random forests*. *Machine Learning*, 45(1):5–32, 2001.

[10] P. Chikersal, S. Poria, E. Cambria, A. F. Gelbukh, and C. E. Siong. *Modelling public sentiment in Twitter*. In *CICLing (2)*, pages 49–65, 2015.

[11] C. Cortes and V. Vapnik. *Support-vector networks*. *Machine Learning*, 20(3):273–297, 1995.

[12] P. Cortez, S. Moro, P. Rita, D. King, and J. Hall. *Insights from a text mining survey on expert systems research from 2000 to 2016*. *Expert Systems*, 35(3):1–10, 2018.

[13] E. Costa-Montenegro, A. Tsybanev, H. Cerezo-Costas, F. J. González-Castaño, F. J. Gil-Castiñeira, A. B. Barragáns-Martínez, and D. Almuiña-Troncoso. *In-memory distributed software solution to improve the performance of recommender systems*. *Softw., Pract. Exper.*, 47(6):867–889, 2017.

[14] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. *Getting more out of biomedical documents with GATE's full lifecycle open source text analytics*. *PLoS Computational Biology*, 9(2), 2013.

[15] C. N. dos Santos, B. Xiang, and B. Zhou. *Classifying relations by ranking with convolutional neural networks*. In *ACL (1)*, pages 626–634, 2015.

[16] E. I. Elmurngi and A. Gherbi. *Unfair reviews detection on amazon reviews using sentiment analysis with supervised learning techniques*. *JCS*, 14(5):714–726, 2018.

[17] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. *Open Information Extraction: the second generation*. In *IJCAI*, pages 3–10, 2011.

[18] B. Evelson and S. Sridharan. *The forrester wave: AI-based text analytics platforms*, 2018. Available at https://www.forrester.com/report/The+Forrester+Wave+AIBased+Text+Analytics+Platforms+Q2+2018/-/E-RES141340.

[19] D. Ferrucci, A. Lally, K. Verspoor, and E. Nyberg. *Unstructured information management architecture (UIMA)*, 2009. Available at https://docs.oasis-open.org/uima/v1.0/uima-v1.0.html.

[20] F. O. Gallego. *Torii: A novel attribute-based polarity analysis*. In *PAAMS (Doctoral consortium)*, pages 395–397, 2016.

[21] F. O. Gallego. *Torii: Attribute-based polarity analysis with big datasets*. In *SIGIR (Doctoral consortium)*, page 1187, 2016.

[22] F. O. Gallego and R. Corchuelo. *A dataset to evaluate condition miners*. In *MLSS*, pages 1–10, 2018.

[23] F. O. Gallego and R. Corchuelo. *A hybrid approach to mining conditions*. In *HAIS*, pages 264–276, 2018.

[24] F. O. Gallego and R. Corchuelo. *A dataset to evaluate condition miners*. In *ESWC*, 2019. Under review.

[25] F. O. Gallego and R. Corchuelo. *A deep-learning approach to mining conditions*. *Knowl.-Based Syst.*, 2019. Under review.

[26] F. O. Gallego and R. Corchuelo. *On mining conditions using encoder-decoder networks*. In *ICAART*, 2019. Under review.

[27] F. O. Gallego and R. Corchuelo. *Torii: An aspect-based sentiment analysis system that can mine conditions*. *Softw., Pract. Exper.*, 2019. Under review.

[28] S. Grimes. *Sentiment, subjectivity, and social analysis go to work: An industry view*. In *WASSA@NAACL-HLT*, page 2, 2016.

[29] H.-G. Han, S. Zhang, and J.-F. Qiao. *An adaptive growing and pruning algorithm for designing recurrent neural network*. *Neurocomputing*, 242:51–62, 2017.

[30] M. Honnibal and I. Montani. *spaCy: Industrial-strength natural language processing*, 2018. Available at https://spacy.io/.

[31] M. Hu and B. Liu. *Mining opinion features in customer reviews*. In *AAAI*, pages 755–760, 2004.

[32] Y. Kim. *Convolutional neural networks for sentence classification*. In *EMNLP*, pages 1746–1751, 2014.

[33] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*. In *ICML*, pages 282–289, 2001.

[34] Y. LeCun, Y. Bengio, and G. E. Hinton. *Deep learning*. *Nature*, 521: 436–444, 2015.

[35] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. *The stanford CoreNLP natural language processing toolkit*. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[36] F. Marozzo, D. Talia, and P. Trunfio. *A workflow management system for scalable data mining on clouds*. *IEEE Trans. Services Computing*, 11 (3):480–492, 2018.

[37] Mausam. *Open information extraction systems and downstream applications*. In *IJCAI*, pages 4074–4077, 2016.

[38] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. *Open language learning for information extraction*. In *EMNLP-CoNLL*, pages 523–534, 2012.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. *Distributed representations of words and phrases and their compositionality*. In *NIPS*, pages 3111–3119, 2013.

[40] T. M. Mitchell, W. W. Cohen, E. R. Hruschka, P. P. Talukdar, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. T. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. *Never-ending learning*. In *AAAI*, pages 2302–2310, 2015.

[41] Y. Nakayama and A. Fujii. *Extracting condition-opinion relations toward fine-grained opinion mining*. In *EMNLP*, pages 622–631, 2015.

[42] R. Narayanan, B. Liu, and A. N. Choudhary. *Sentiment analysis of conditional sentences*. In *EMNLP*, pages 180–189, 2009.

[43] M. Nilashi, O. Ibrahim, and K. Bagherifard. *A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Syst. Appl.*, 92:507–520, 2018.

[44] M. Nußbaum-Thom, J. Cui, B. Ramabhadran, and V. Goel. *Acoustic modeling using bi-directional gated recurrent convolutional units*. In *Interspeech*, pages 390–394, 2016.

[45] A. G. Pablos, M. Cuadros, and G. Rigau. *W2VLDA: almost unsupervised system for aspect-based sentiment analysis. Expert Syst. Appl.*, 91: 127–137, 2018.

[46] L. Padró and E. Stanilovsky. *Freeling 3.0: towards wider multilinguality*. In *LREC*, pages 2473–2479, 2012.

[47] R. Pascanu, T. Mikolov, and Y. Bengio. *On the difficulty of training recurrent neural networks*. In *ICML*, pages 1310–1318, 2013.

[48] J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, 1989.

[49] J. Pennington, R. Socher, and C. D. Manning. *Glove: Global vectors for word representation*. In *EMNLP*, pages 1532–1543, 2014.

[50] K. Ravi and V. Ravi. *A survey on opinion mining and sentiment analysis: tasks, approaches and applications*. *Knowl.-Based Syst.*, 89:14–46, 2015.

[51] J. Schmidhuber. *Deep learning in neural networks: An overview*. *Neural Networks*, 61:85–117, 2015.

[52] K. Schouten and F. Frasincar. *Survey on aspect-level sentiment analysis*. *IEEE Trans. Knowl. Data Eng.*, 28(3):813–830, 2016.

[53] M. Schuster and K. K. Paliwal. *Bidirectional recurrent neural networks*. *IEEE Trans. Signal Processing*, 45(11):2673–2681, 1997.

[54] M. Skeppstedt, T. Schamp-Bjerede, M. Sahlgren, C. Paradis, and A. Kerren. *Detecting speculations, contrasts and conditionals in consumer reviews*. In *WASSA@EMNLP*, pages 162–168, 2015.

[55] I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to sequence learning with neural networks*. In *NIPS*, pages 3104–3112, 2014.

[56] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. *Inception-V4, Inception-ResNet, and the impact of residual connections on learning*. In *AAAI*, pages 4278–4284, 2017.

[57] M. Taboada, J. Brooke, M. Tofiloski, K. D. Voll, and M. Stede. *Lexicon-based methods for sentiment analysis*. *Computational Linguistics*, 37(2): 267–307, 2011.

[58] P. Tang, H. Wang, and S. Kwong. *GoogleNet based multi-stage feature fusion of deep CNN for scene recognition*. *Neurocomputing*, 225: 188–197, 2017.

[59] The Apache Software Foundation. *Apache OpenNLP*, 2018. Available at https://opennlp.apache.org.

[60] H.-L. Yang and Q.-F. Lin. *Opinion mining for multiple types of emotion-embedded products/services through evolutionary strategy*. *Expert Syst. Appl.*, 99:44–55, 2018.

[61] S. Yoo, J. Song, and O. Jeong. *Social media contents based sentiment analysis and prediction system*. *Expert Syst. Appl.*, 105:102–111, 2018.

[62] F. Zhai, S. Potdar, B. Xiang, and B. Zhou. *Neural models for sequence chunking*. In *AAAI*, pages 3365–3371, 2017.

This document was typeset on January 14, 2019 at 12:03 using class RG-BOOK α2.14 for LaTeX$_{2\varepsilon}$. As of the time of writing this document, this class is not publicly available since it is in alpha version. Only members of The Distributed Group are using it to typeset their documents. Should you be interested in giving forthcoming public versions a try, please, do contact us at contact@tdg-seville.info. Thanks!