



SAPIENZA  
UNIVERSITÀ DI ROMA

## Ontology extraction and population from user-generated text on online marketplaces

Ingegneria dell'Informazione, Informatica e Statistica  
Corso di Laurea Magistrale in Ingegneria Informatica

Candidate

Sara Di Bartolomeo  
ID number 1494990

Thesis Advisor

Prof. Riccardo Rosati

Co-Advisor

Dr. Nome Cognome

Academic Year 2016/2017

Thesis defended on 16 April 2013  
in front of a Board of Examiners composed by:  
Prof. Nome Cognome (chairman)  
Prof. Nome Cognome  
Dr. Nome Cognome

---

**Ontology extraction and population from user-generated text on online marketplaces**

Master thesis. Sapienza – University of Rome

© 2017 Sara Di Bartolomeo. All rights reserved

This thesis has been typeset by L<sup>A</sup>T<sub>E</sub>X and the Sapthesis class.

Version: December 19, 2017

Author's email: [dibartolomeo.sara@gmail.com](mailto:dibartolomeo.sara@gmail.com)

*Dedicato a*



## Abstract



## Acknowledgments

*I would like to thank the whole Dipartimento di Ingegneria Informatica, Automatica e Gestionale at Sapienza. From what I experienced, the department aggregates a great deal of brilliant and passionate students and professors, from which I had the opportunity to learn the proudness in dedication and discipline, the love for the subjects, the long lasting satisfaction in pursuing achievements. The years I have spent in this building forged me into an Engineer, with a sharp attention for details and a deep rooted thirst for knowledge.*

*Specifically, I would like to thank professor Riccardo Rosati, who has been my thesis advisor, who trusted my ideas and wisely guided me through the process of shaping them into a Master's thesis.*

*I would also like to thank each one of my colleagues, as many of them have been role models and figures of guidance to me. Matteo, Alessandro, Lorenzo, Alessio, Gabriele [etc]. Along with them, I want to mention a couple of colleagues from Codemotion, Massimo and Bruno, who made me really understand that creativity and technology weren't mutually exclusive concepts, and with whom I had a productive friendship that made me grow up []*

*Lastly, I want to thank the single person who has been, at the same time, a loving partner, a perfect teammate, and a best friend. Giorgio, who has been at my side through all the difficulties I had to face.*





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Related works</b>	<b>3</b>
<b>3</b>	<b>Collecting Data</b>	<b>5</b>
3.1	Amazon . . . . .	5
3.2	The challenges . . . . .	6
3.2.1	Amazon API . . . . .	6
3.3	Scraping . . . . .	6
3.3.1	Scraping a single page . . . . .	7
3.3.2	Dynamic pages . . . . .	8
3.4	Scraping on a larger scale . . . . .	10
3.4.1	Identifying bottlenecks . . . . .	10
3.4.2	Multithreading . . . . .	10
3.5	Tampering requests . . . . .	10
3.5.1	Proxies . . . . .	10
3.5.2	Crafting user agents . . . . .	10
3.6	amazon-scraper . . . . .	10
3.7	Shape of the collected data . . . . .	10
<b>4</b>	<b>Processing the information</b>	<b>13</b>
4.1	Natural language processing . . . . .	13
4.1.1	Pattern recognition in strings . . . . .	14
4.1.2	Named entity recognition . . . . .	14
4.1.3	Language use differences in reviews and questions . . . . .	15
4.2	Questions and Answers . . . . .	16
4.2.1	Open-ended questions and closed-ended questions . . . . .	16
4.2.2	Detecting the object of a question . . . . .	17
4.2.3	Differences in Q/A language for different product categories . . . . .	18
4.3	Reviews . . . . .	18

---

4.3.1	Sentence structure . . . . .	22
4.3.2	Subjects, verbs and objects . . . . .	22
4.3.3	Relations . . . . .	22
<b>5</b>	<b>Ontology extraction and population</b>	<b>23</b>
5.1	Ontologies . . . . .	23
5.2	Translating the data to an OWL ontology . . . . .	24
5.3	Taxonomy of classes . . . . .	24
5.4	Populating the ontology via python . . . . .	25
5.5	dataProperties and objectProperties . . . . .	25
5.6	Querying the ontology . . . . .	25
<b>6</b>	<b>Results</b>	<b>27</b>
6.1	Use cases . . . . .	27
<b>7</b>	<b>Future work</b>	<b>29</b>
<b>8</b>	<b>Conclusioni</b>	<b>31</b>

# Chapter 1

## Introduction

In recent years, the shopping habits of the general public have been changing considerably: more and more people are choosing to use online marketplaces for their purchases. This form of shopping platform has introduced several advantages: users have access to a much bigger selection of products, barriers that would otherwise have prevented access to the market to smaller sellers have been broken down.

Nevertheless, these advantages are balanced by other downsides. As an example, judging the quality and features of an item is a more complex task, as an user will only have access to photos and textual descriptions of the item.

In this context, a trust-based system between multiple users has gained more and more relevance: reviews and reputation. The judgement of a buyer will be substantially affected by other users' reviews, as well as a level of trustworthiness assigned to sellers by other users.

We have to take into account that reviews represent a relevant strategic resource to sellers. Making an user able to gain knowledge about the quality, purpose and details of an item is a problem that concerns not only the users, but also the sellers. Indeed, a number of good reviews may be the deciding factor for a successful sale.

[sellers buying reviews]

quotes on trust system

Since reviews are so relevant, they have been the object of a series of studies. They are, infact, made of unstructured informations about a product: the data they contain, if succesfully extracted, may help in identifying precisely the features of a product.

The overwhelming amount of informations contained in reviews makes them difficult to consider in every detail for a human reader.

If we consider the scale on which the informations we obtain becomes significative, though, we notice that an automatized system for extracting information from text

is needed.

The project presented in this thesis is aimed at extracting and structuring the information contained in user generated content regarding products on online marketplaces.

Natural Language Processing is the field that studies the interpretation, from an algorithmic point of view, of phrases in commonly written/spoken languages. Thanks to advancements in this field, we are able

## Chapter 2

# Related works

In 2012, S.Z. Haider [4] published a case study in which he used reviews from Amazon to populate an ontology. His approach used a pre-made ontology about mobile phones, and applied sentiment analysis to the Amazon reviews of three specific items (three mobile phones) to understand the opinion of the public about the qualities of the items that were being analyzed.

Biemann [2] published in 2005 a survey of methods to study the different approaches at extracting data from unstructured text in order to build an ontology.

Julian McAuley and Alex Yang [6] used an approach based on machine learning to classify questions and answers in the Questions/Answers section of Amazon.

The purpose of this thesis is to build on the aforementioned works to:

- include information from Questions/Answers sections as well as reviews
- include a much bigger range of products and categories of products
- extract the structure of the ontology from the data



## Chapter 3

# Collecting Data

Modern online marketplaces often offer customers the ability to give feedback to the vendors, often in the form of reviews. Recently, some platforms also started to insert question/answer forms, in order to let the users be able to formulate questions about a product, and have other users or the seller respond to their inquiries about the product.

Feedback is given by the users in the form of:

- Numerical score (i.e. 1 to 5 stars)
- Reviews
- Questions and Answers

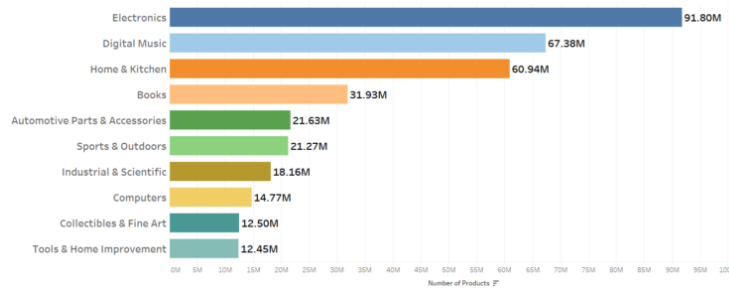
Although the first value is very easy to be semantically represented, the latter ones aren't so straightforward. They are, infact, expressed in natural language, and their analysis will be the focus of the following chapters.

The first step of the project is, thus, the collection of data.

### 3.1 Amazon

The main focus of the project is extracting data from the biggest marketplace as of today, namely, Amazon. As stated in an article by Business Insider [5], Amazon is accountable for 43% of online sales in the US in 2016.

TODO: insert more data about amazon, explain why amazon is important



**Figure 3.1.** a comparison between different product categories on Amazon

## 3.2 The challenges

### 3.2.1 Amazon API

Amazon offers an API that makes available some of the information I wanted to examine. Unfortunately, not the request limits imposed to API users nor the information made available were useful for the purpose of this project.

Indeed, on 8 November 2010 Amazon removed the possibility to retrieve reviews from their API, returning now an URL to an IFrame containing just the first three reviews. The reason behind the removal of this feature were never explained by Amazon, and it has been discussed that the huge amount of data represented by written feedback is now considered a valuable resource that Amazon wants to protect.

Offering the first three reviews in an IFrame is intended for sellers to showcase on their website some Amazon reviews about their products, but I needed much more than just three reviews per product. The nature of the IFrame DOM element makes it difficult to deal with for accessing its contents, and makes the process of retrieving clean review content similar to the process of scraping a web page.

## 3.3 Scraping

Scraping is a technique for extracting data from a website via a software program. Scraper programs simulate human behaviour in accessing the content in webpages, with the purpose of collecting and transforming unstructured data, often found inside the HTML code of webpages, in metadata useful for being memorized, manipulated and analyzed locally.

A scraper program will access the content on a page directly through the HTTP protocol. A page is first downloaded through an HTTP GET request, in the same way in which a browser would request a page for visualizing it. The web server at



the requested address will respond with the HTML code used by the browser to visually render the page. Instead of rendering it, a scraper will parse the code of the page to find patterns and references for purposeful data in the HTML. An example of this is contact scraping: a scraper is useful in finding all the email addresses in a page.

Commonly, scraping is done on a huge number of pages at the same time, and is therefore deeply linked with web crawling, that is the technique of navigating through a series of links to obtain a huge number of pages.

In addition to a number of challenges directly involved in the process, some websites will try to prevent this practice through various techniques, like IP blacklisting, CAPTCHAs, A/B testing and obfuscation of the content.

### 3.3.1 Scraping a single page

Scraping a single web page is performed in the following steps:

1. An HTTP GET request is used to download the HTML code of the page that is being analyzed. The response, if the request was successful, is HTML plain text.
2. The response is then parsed to identify HTML components.
3. Thanks to the tree-like structure of HTML elements in a page, each component is stored in a tree shaped data structure, called parsetree.
4. The parsetree is searched according to several criteria (an example may be regular expression matching) to extract relevant data.

HTTP/1.1 404 Not Found

Date: Sun, 18 Oct 2012 10:36:20 GMT

Server: Apache/2.2.14 (Win32)

Content-Length: 230

Connection: Closed

Content-Type: text/html; charset=iso-8859-1

<html>

<head>

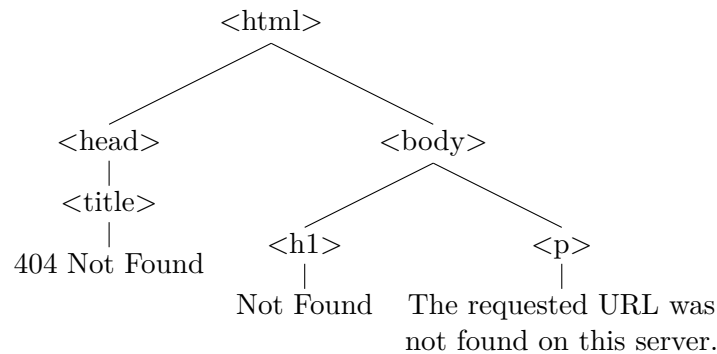
<title>404 Not Found</title>

</head>

<body>

<h1>Not Found</h1>

<p>The requested URL was not found on this server.</p>



**Figure 3.2.** The parsetree generated from 3.3.1

```

</body>
</html>

```

For the purpose of parsing, storing and accessing the result of the request, I used a python library designed to ease this task, BeautifulSoup [1]. Processing the page in this way allows me to navigate HTML easily and query the contents.

In the code example above, at 3.3.1, retrieving the title of the received page would be written like this:

```

BeautifulSoup bs = BeautifulSoup(res , 'html.parser')
title = bs.find('title').text

```

This code would perform a search on the document's parsetree, then extract what is contained in the tag `<title>`.

Real world html pages are much more complex than this, usually containing thousands of tags, and are often dynamically generated, thus require fine scraping criteria. The procedure used needs to be able to adapt to slight modifications in the page structure and contents.

As you can see in 3.3, the structure of a standard product page on Amazon is convoluted, and identifying the parts that contain relevant information via code may not be straightforward.

### 3.3.2 Dynamic pages

The modern web is comprised of a huge amount of content that is purposefully tailored for the user's needs and desires. In particular, when an HTTP request is made to a server, the server may respond with content specifically tailored for the particular user who made the request.

Think of facebook: each user sees the same website at the same address in a

amazon **US**

Departments **Home & Garden** **Electronics** **Books** **Video Games** **Music** **Gift Cards** **Registry** **Sell** **Help** **EN** **Account & Lists** **Orders** **Try Prime** **Cart**

Toys & Games **Holiday Toy List** **Shop now**

Toys & Games • Kids' Electronics • Remote- & App-Controlled Figures & Robots

**Wonder Workshop**  
**Wonder Workshop Cue Robot**  
 ★★★★★ **28 customer reviews** | **18 answered questions**

List Price: \$169.99  
 Deal of the Day: **\$139.99** + \$77.29 Shipping & Import Fees Deposit to Italy **Details**  
 Ends in 07h 59m 38s  
 You Save: \$60.00 (35%)

**In Stock.**  
 This item ships to **Roma, Italy**. Want it Tuesday, Dec. 27? Order within **18 hrs 14 mins** and choose **AmazonGlobal Priority Shipping** at checkout. **Learn more**  
 Ships from and sold by Amazon.com. Gift-wrap available.

- Cue is a witty robot with attitude that is powered by breakthrough Emotive AI
- Build your skills with games and challenges and makes programming your own interactive experiences fun for any level
- Choose your favorite avatar and explore an amazing depth of personality, expressions and actions
- Send and receive text messages to share witty comments, memes and funny jokes, keeping you coming back for more.
- Unlock cue's secret to coding for any skill level by easily switching between block and JavaScript programming
- Discover a headless environment to program robot adventures using cue's proximity sensors, encoders, gyro, accelerometer, microphones and more.
- Engage Cue's intelligent auto modes (seek, avoid, and explore) to navigate tight corners or obstacles while expressing personality at every turn.
- Cue is for ages 11+ and works with iOS, Android and Kindle Fire HD 8, 10, 2017 SILVER PARENTS' CHOICE AWARD WINNER

Compare with similar items  
**New (4)** from \$139.99 & FREE shipping. **Details**  
[Report incorrect product information.](#)

**HOLIDAY Toy List**  
 Explore more

Qty: **1**

**Add a Protection Plan:**  
☐ 5-Year Accident Protection for \$22.99  
☐ 2-Year Accident Protection for \$15.99

**Add to Cart**

Turn on 1-Click ordering for this browser

**Ship to:**  
 Sara Di Bartolomeo - Roma - 00162  
 +

**Add to List**

**Other Sellers on Amazon**  
**New (4)** from \$139.99 & FREE shipping. **Details**  
 Have one to sell? **Sell on Amazon**

**Frequently bought together**

**Total price: \$289.93**  
**Add both to Cart**  
**Add both to List**

☒ **This item:** Wonder Workshop Cue Robot **\$139.99**  
☒ **Wonder Workshop Dash Robot** **\$149.94**

**Customers who bought this item also bought**

Page 1 of 11

Product	Price	Prime
For Cue Robot Decorative Vinyl Decal Wrap Skin Set Includes Wheels & Fender Decoration Do It Yourself by JPS (Pearl Red FLAME)	\$9.95	prime
Wonder Workshop Dot Creativity Kit Robot	\$79.99	prime
Wonder Workshop Dash Robot	\$149.94	prime
Hard EVA Travel Case for Wonder Workshop Dash Robot by HermitHill	\$21.99	prime
For Cue Face Screen, Fenders, Wheels & Back Invisible Guard protector 9 piece set best protection against scratches while...	\$11.90	prime
Dash & Dot Learn to Code Challenge Card Box Set	\$39.99	prime
LEGO Boost Creative Toolbox 17101 Building and Coding Kit (847 Pieces)	\$159.95	prime

**Special offers and product promotions**  
 • Your cost could be \$89.99 instead of \$139.99. Get a **\$50 Amazon.com Gift Card** instantly upon approval for the Amazon Rewards Visa Card. **Apply now**

**Have a question?**  
 Find answers in product info, Q&As, reviews

Figure 3.3. a product page on amazon

unique way, depending on the user's account, the cookies he has stored in its website, the times and types of content that the user's friends posted.

This may happen in the case of:

- visual elements:
- content:

## 3.4 Scraping on a larger scale

Scraping a large quantity of content introduces several challenges that I had to overcome.

### 3.4.1 Identifying bottlenecks

### 3.4.2 Multithreading

## 3.5 Tampering requests

### 3.5.1 Proxies

### 3.5.2 Crafting user agents

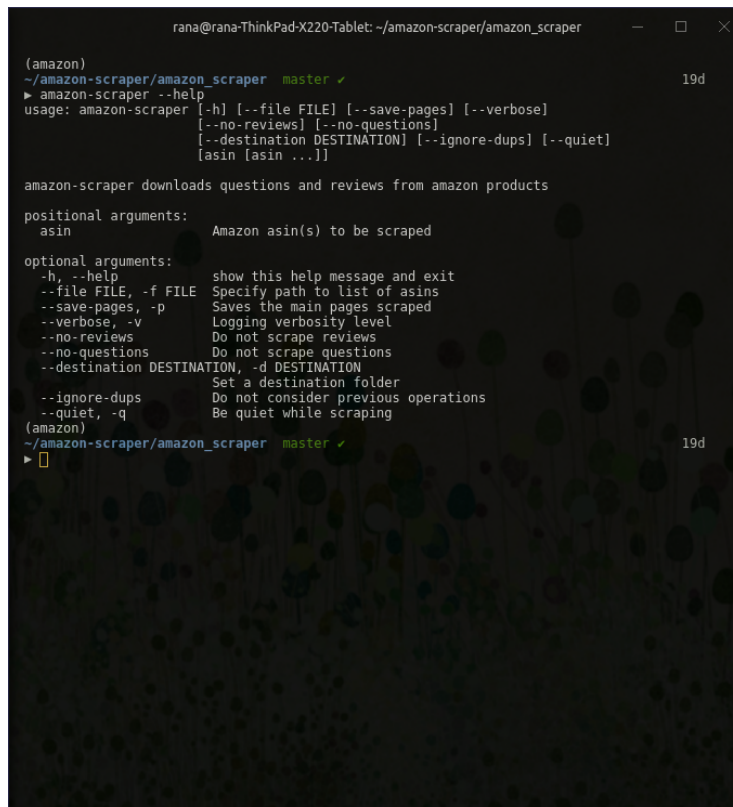
## 3.6 amazon-scraper

`amazon-scraper` is the name I gave to the software I wrote. In 3.4

`amazon-scraper` features:

- 
- 

## 3.7 Shape of the collected data



```
(amazon)
~/amazon-scraper/amazon_scraper master ✓ 19d
▶ amazon-scraper --help
usage: amazon-scraper [-h] [--file FILE] [--save-pages] [--verbose]
                    [--no-reviews] [--no-questions]
                    [--destination DESTINATION] [--ignore-dups] [--quiet]
                    [asin [asin ...]]

amazon-scraper downloads questions and reviews from amazon products

positional arguments:
  asin                Amazon asin(s) to be scraped

optional arguments:
  -h, --help            show this help message and exit
  --file FILE, -f FILE  Specify path to list of asins
  --save-pages, -p      Saves the main pages scraped
  --verbose, -v         Logging verbosity level
  --no-reviews          Do not scrape reviews
  --no-questions        Do not scrape questions
  --destination DESTINATION, -d DESTINATION
                        Set a destination folder
  --ignore-dups         Do not consider previous operations
  --quiet, -q          Be quiet while scraping

(amazon)
~/amazon-scraper/amazon_scraper master ✓ 19d
▶ □
```

Figure 3.4. amazon-scraper usage



## Chapter 4

# Processing the information

Once enough data has been gathered, we can proceed with the language analysis part. The intent of this analysis is to extract meaningful information from text. User generated text on product pages can infact contain relevant details about a product, that may or may not be specified in the seller's description of the same product.

The reasons for a missing detail in the seller-provided [?] description may be several:

- The seller didn't consider the detail relevant, but discussion about the detail present in the product's comments proves that clients consider the detail important
- The seller forgot to include the detail
- The seller didn't realize that his product had a particular characteristic
- The seller lied about a characteristic of his product

The characteristics listed above define a knowledge about the product that may be considered useful for a client to judge a product, but may not be easily understood. Moreover, these characteristics are relevant in our intent of building an ontology of products.

[add content here]

### 4.1 Natural language processing

Natural Language Processing is the process of programmatically treating information written in natural language, that is in this case American English.

Natural language contains data, but that data is not easily usable by a computer program. A phrase like

This tablet costs 279\$

clearly contains a relevant information: the price of an item we may be interested in, clearly underlined by the fact that the string contains a number. Nevertheless, our human brain is wired to understand that the number represents a price, but the same is not a straightforward process for a computer program. The process is made more difficult by the underlying ambiguity of the human language.

The process of analyzing a string is performed in four different stages:

- Splitting the string in sentences and words, namely *tokens*
- Assigning each chunk (or word) a part-of-speech tag
- Arranging the tokens in a syntactical structure (a *parsetree*)
- Assigning to the structure a semantical meaning

[decision trees for tagging vs machine learning techniques]

To perform these tasks (for example, part-of-speech tagging), we rely on machine learning techniques to learn from a large *corpora*<sup>1</sup> how certain words in certain positions are intended.

The machine learning approach is much more versatile because it is able to manage unfamiliar structures - that is, structures of the phrase that were not present in the original dataset, but can be derived from similar entries.

#### 4.1.1 Pattern recognition in strings

#### 4.1.2 Named entity recognition

[explain better how named entity recognition works]

An important task in this project was the extraction of relevant entities from the sentences written by the users.

For this purpose, a python library, **spacy**, has been used. **spacy** features a default model that maps entities to []

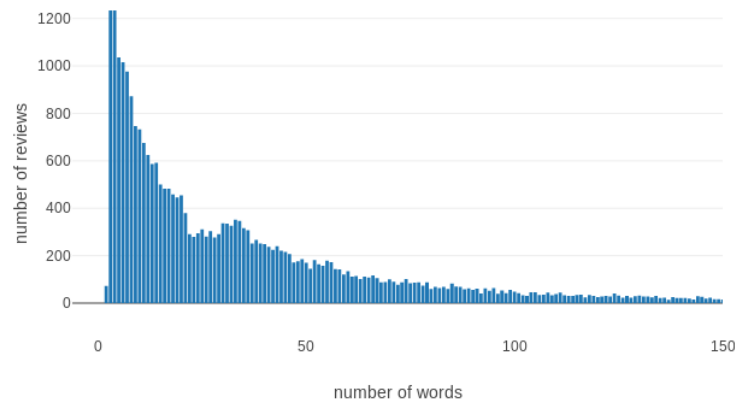
An example of Named Entity Recognition performed on a simple phrase:

Linus Benedict Torvalds<sup>Person</sup> (born December 28, 1969<sup>Date</sup>) is a Finnish  
NORP software engineer who is the creator, and for a long time, principal  
developer of the Linux<sup>ORG</sup> kernel.

It can easily be noticed that this sentence contains a lot of information: named entity recognition can recognize where the relevant data is, and what kind of data type it is.

<sup>1</sup>A corpora is a large collection of documents with human-made annotations





**Figure 4.1.** Frequency distribution of the number of words used in reviews

#### 4.1.3 Language use differences in reviews and questions

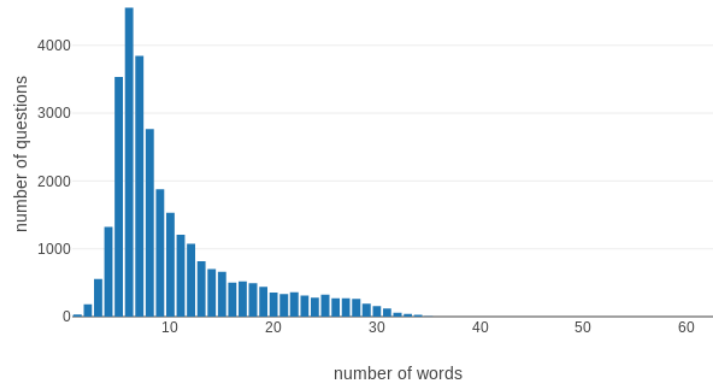
After having collected enough data, it became apparent that there were relevant differences in language usage in Q/A and reviews. Questions, indeed, tend to be much shorter, use a simpler language and are much more straightforward than reviews. Although very short reviews can be found when dealing with a very diverse set of user generated content, they usually talk about multiple features of a product, while questions and answers usually address just one information. The structure of a sentence is also evidently different.

For this reason, I deemed appropriate to separate the methods in which Q/A and reviews are treated in the analysis process.

In 4.2 and 4.1, the frequency distribution of the number of words used in both reviews and questions/answers is shown. It can clearly be seen that it is much more common for questions to employ much less words, while reviews are more verbose.

Another indicator of a more complex use of the language in reviews is [variance in frequency distribution of words]

[insert average question length for type of question] [insert word frequency distribution for each category]



**Figure 4.2.** Frequency distribution of the number of words used in questions/answers

## 4.2 Questions and Answers

### 4.2.1 Open-ended questions and closed-ended questions

Questions about a product on a online marketplace can be open-ended or closed-ended. The affiliation of each question to one of these categories entails major differences in how information is expressed.

A **closed-ended** question is a question that accepts 'yes' or 'no' as answer. An example of a closed-ended question is:

**Question:** Does this phone have a camera?

**Answer:** Yes.

An **open-ended** question is, instead, a question that expects a more complex answer, such as a list, an explanation, a description. An example of an open-ended question is:

**Question:** What features does this phone have?

**Answer:** A camera, two sim slots and a headphone jack.

As you can see, information is expressed in a very straightforward way in closed-ended questions. If we, through natural language processing, manage to understand the object of the question, the answer is just a boolean value representing if the analyzed product corresponds to the requested quality or not.

### Classifying the questions

Based on the work of [insert link], I used a simple Naive Bayes classifier to discern closed-ended from open-ended questions. The classifier has been trained on a pre-labeled dataset of [N] questions and answers. The labels were "open" or "Y/N".

### Classifying the answers

In the case of closed-ended questions, I analyzed the answers. The answers may be as simple as a "Yes" or "No", and in this situation the outcome is straightforward, but it is not always the case. Positive answers may appear in forms similar to:

**Question:** Does it work in Argentina?

**Answer:** It does work very well in Argentina.

**Question:** Does it work in Argentina?

**Answer:** Absolutely.

**Question:** Does it work in Argentina?

**Answer:** I think it does.

These are positive forms, but do not include the term "Yes".

Another detail to take into account is the possibility for a user to answer with an unclear or undefined answer. For this purpose, when classifying the answers, we consider a third "Unknown" label. Answers along the lines of:

**Answer:** I don't know. **Answer:** I'm not sure.

or other unclear answers are considered Unknown answers.

#### 4.2.2 Detecting the object of a question

[better] Named entities are definite noun phrases that refer to specific types of individuals, such as organizations, persons, dates, and so on. 5.1 lists some of the more commonly used types of NEs. These should be self-explanatory, except for "Facility": human-made artifacts in the domains of architecture and civil engineering; and "GPE": geo-political entities such as city, state/province, and country.

ORGANIZATION	Georgia-Pacific Corp., WHO
PERSON	Eddy Bonte, President Obama
LOCATION	Murray River, Mount Everest
DATE	June, 2008-06-29
TIME	two fifty a m, 1:30 p.m.
MONEY	175 million Canadian Dollars, GBP 10.40
PERCENT	twenty pct, 18.75%
FACILITY	Washington Monument, Stonehenge
GPE	South East Asia, Midlothian

The goal of a named entity recognition (NER) system is to identify all textual mentions of the named entities. This can be broken down into two sub-tasks: identifying the boundaries of the NE, and identifying its type. While named entity recognition is frequently a prelude to identifying relations in Information Extraction, it can also contribute to other tasks. For example, in Question Answering (QA), we try to improve the precision of Information Retrieval by recovering not whole pages, but just those parts which contain an answer to the user's question. Most QA systems take the documents returned by standard Information Retrieval, and then attempt to isolate the minimal text snippet in the document containing the answer. Now suppose the question was Who was the first President of the US?, and one of the documents that was retrieved contained the following passage:

The Washington Monument is the most prominent structure in Washington, D.C. and one of the city's early attractions. It was built in honor of George Washington, who led the country to independence and then became its first President.

Analysis of the question leads us to expect that an answer should be of the form X was the first President of the US, where X is not only a noun phrase, but also refers to a named entity of type PERSON. This should allow us to ignore the first sentence in the passage. While it contains two occurrences of Washington, named entity recognition should tell us that neither of them has the correct type.

#### 4.2.3 Differences in Q/A language for different product categories

### 4.3 Reviews

[better]

- Information extraction systems search large bodies of unrestricted text for specific types of entities and relations, and use them to populate well-organized

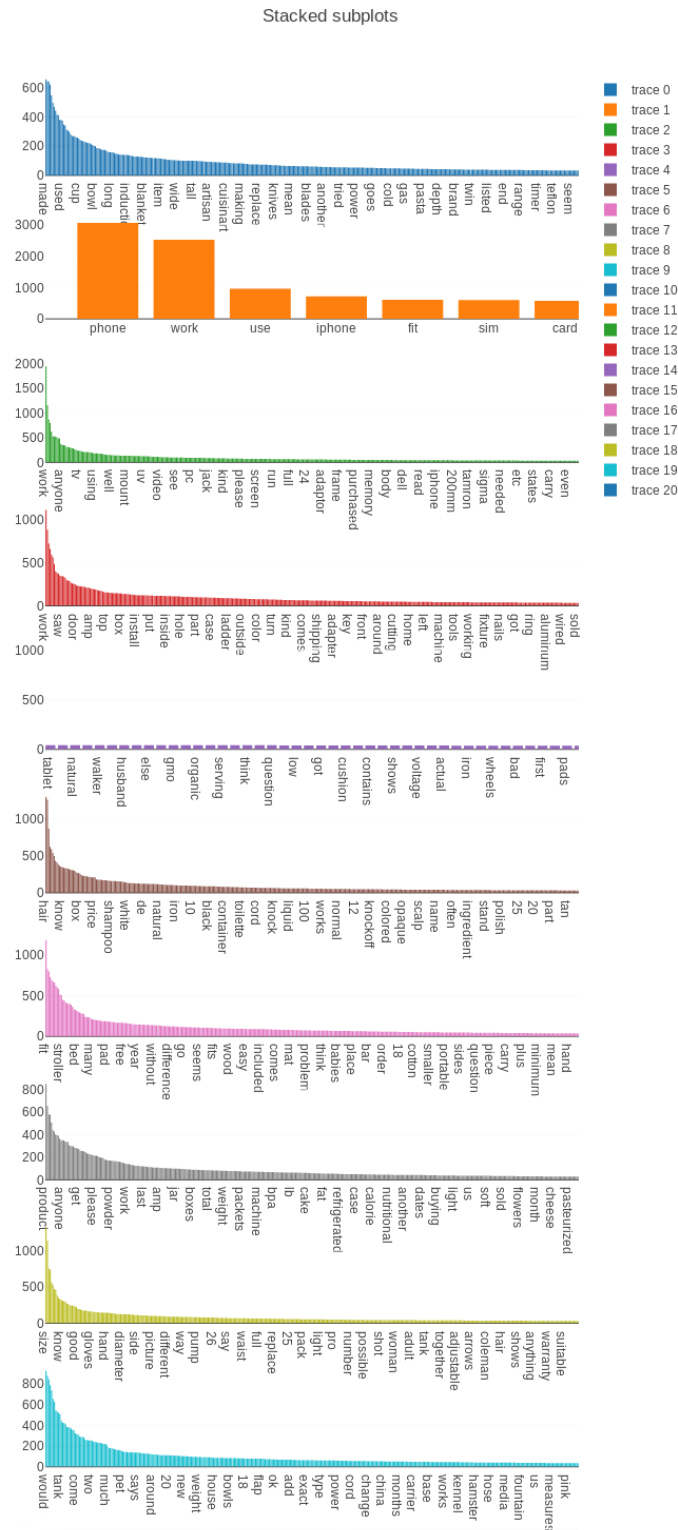


Figure 4.3. A tree of chunks of a sentence

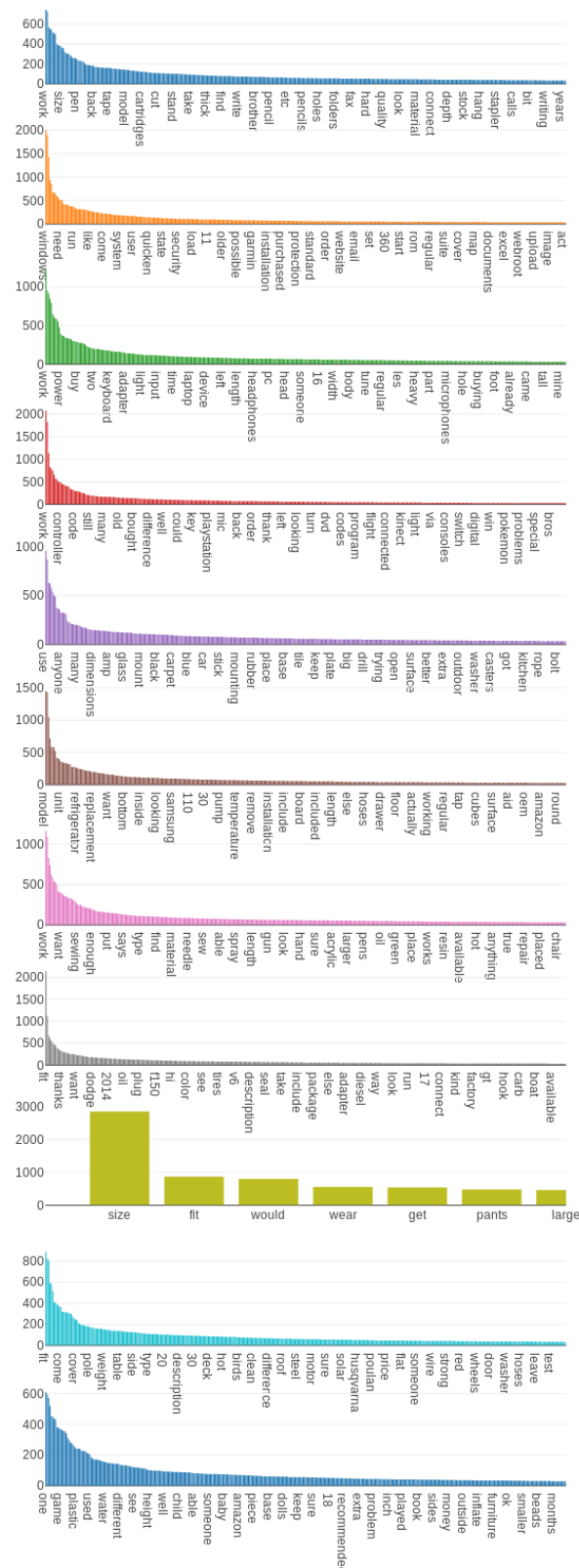
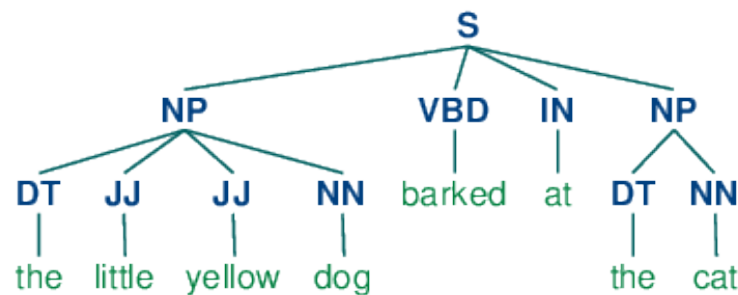


Figure 4.4. A tree of chunks of a sentence



**Figure 4.5.** A tree of chunks of a sentence

databases. These databases can then be used to find answers for specific questions.

- The typical architecture for an information extraction system begins by segmenting, tokenizing, and part-of-speech tagging the text.
- The resulting data is then searched for specific types of entity. Finally, the information extraction system looks at entities that are mentioned near one another in the text, and tries to determine whether specific relationships hold between those entities.
- Entity recognition is often performed using chunkers, which segment multi-token sequences, and label them with the appropriate entity type. Common entity types include ORGANIZATION, PERSON, LOCATION, DATE, TIME, MONEY, and GPE (geo-political entity).
- Chunkers can be constructed using rule-based systems, such as the Regexp-Parser class provided by NLTK; or using machine learning techniques. In either case, part-of-speech tags are often a very important feature when searching for chunks.
- Although chunkers are specialized to create relatively flat data structures, where no two chunks are allowed to overlap, they can be cascaded together to build nested structures.
- Relation extraction can be performed using either rule-based systems which typically look for specific patterns in the text that connect entities and the intervening words; or using machine-learning systems which typically attempt to learn such patterns automatically from a training corpus.

#### 4.3.1 Sentence structure

#### 4.3.2 Subjects, verbs and objects

#### 4.3.3 Relations



## Chapter 5

# Ontology extraction and population

The idea of the whole project was to make data collected from product pages on the marketplace - any kind of data present on the page, from the product details that are directly provided by the vendor, like size and color of an item, to data present in user generated text, namely reviews and questions/answers.

The data collected from the analysis is composed by different data types, classes of items and relationships between them.

As defined in [Gruber, 1993] [3],

A specification of a representational vocabulary for a shared domain of discourse — definitions of classes, relations, functions, and other objects — is called an ontology.

An ontology is perfectly apt to describe, in a human-readable and useful for [being interpreted by a computer] data and relationships between data.

### 5.1 Ontologies

[better] The main aim of ontology is to provide knowledge about specific domains that are understandable by both the computers and developers. It also helps to interpret a text review at a finer granularity with shared meanings and provides a sound semantic ground of machine- understandable description of digital content. Ontology improves the process of information retrieval and reasoning thus results in making data interoperable between different applications (Zhou, Chaovalit, 2007) According to Meersman(2005), most of the ontologies in the community of information systems are known as data models that are mainly used for structuring a fairly narrow application domain. He claimed that ontologies that includes lexicons and thesauri

may be a useful first step in providing and formalizing the semantics of information representation. According to Meersman (2005), in near future these ontologies will act as a semantic domain for the information systems and will be very useful. He also predicted with authenticity that:

“ It is unmistakable that with the advent of e-commerce, and the resulting natural language context of its related activities, that ontologies, lexicons and the thesauri and research in their use for system design and interpretation will receive a major market driven push”

(Meersman,2005,p.02) According to Meersman, (2005), a lexicon is defined as a language-specific ontology, for e.g English, Polish. Whereas thesaurus is defined as either a domain-specific ontology or an application specific ontology. Manufacturing, Laptop-manufacturing, Naïve physics, corporate law, Ontology theory are examples of domain specific ontologies, whereas Inventory Control, Airline Reservations, Conference Organization are examples of application specific ontology. Domain specific ontology and application specific ontology can be distinguished intuitively as the differences between the two types are not distinct. There is difference between ontologies and conceptual schemas but both are intimately related and are used to represent commonly perceived reality. Mathematically, ontology is the domain while the relational schema is the range of the semantic interpretation mapping. Both can be seen as a representation of a commonly perceived reality and intimately related.

## 5.2 Translating the data to an OWL ontology

[TODO]

## 5.3 Taxonomy of classes

[TODO] Amazon categorizes each item in classes.

For example, the item **Trivial Pursuit Game: Classic Edition** is classified as belonging to the class **Board Games**. The class **Board Games** is a subclass of the class **Games**, which is in turn subclass of **Toys & Games**.

**Super Jumbo Playing Cards**, instead, are categorized as **Standard Playing Card Decks**, subclass, again, of **Games**

These classes form a tree-like hierarchical structure.

[Amazon doesn't directly show to us how this tree is made, but we can extract it with python]

Classes		
Depth1	Depth2	Depth3
Defenders	LB	Lucas Radebe
	DC	Michael Duburrry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

## 5.4 Populating the ontology via python

[Populating the ontology via python has been done via the owlready library.]

## 5.5 dataProperties and objectProperties

In OWL, we have two possible kinds of properties that a member of the ontology can have:

- DataProperty
- ObjectProperty

## 5.6 Querying the ontology

[Having an ontology like this one allows us to query the contents] Protegé supports DL queries on the knowledge base.

color value black and brand value Samsung



## Chapter 6

# Results

### 6.1 Use cases



## Chapter 7

### Future work

Selection bias: users who post reviews may be a different subset than users who do not





## Chapter 8

## Conclusioni



# Bibliography

- [1] Beautiful Soup: We called him Tortoise because he taught us. Available from: <https://www.crummy.com/software/BeautifulSoup/>.
- [2] BIEMANN, C. Ontology learning from text: A survey of methods. In *LDV forum*, vol. 20, pp. 75–93 (2005).
- [3] GRUBER, T. R. A translation approach to portable ontology specifications. *Knowledge acquisition*, **5** (1993), 199.
- [4] HAIDER, S. Z. *AN ONTOLOGY BASED SENTIMENT ANALYSIS: A Case Study* (2012). Available from: <http://www.diva-portal.org/smash/record.jsf?pid=diva2:552748>.
- [5] INTELLIGENCE, B. I. Amazon accounts for 43% of US online retail sales. Available from: <http://www.businessinsider.com/amazon-accounts-for-43-of-us-online-retail-sales-2017-2>.
- [6] MCAULEY, J. AND YANG, A. Addressing Complex and Subjective Product-Related Queries with Customer Reviews. pp. 625–635. ACM Press (2016). ISBN 978-1-4503-4143-1. Available from: <http://dl.acm.org/citation.cfm?doid=2872427.2883044>, doi:10.1145/2872427.2883044.