

POLITECNICO DI MILANO  
Facoltà di Ingegneria dell'Informazione



POLO REGIONALE DI COMO  
Master of Science in Computer Engineering

ON THE USE OF CORRESPONDENCE ANALYSIS TO LEARN  
ONTOLOGIES AND OTHER SEMANTICS FROM TEXT

Supervisor: Matteo Matteucci  
Assistant Supervisor: Davide Eynard

Master Graduation Thesis of: Fabio Marfia  
Student Id. Number: 718843

Academic Year 2008/2009



POLITECNICO DI MILANO  
Facoltà di Ingegneria dell'Informazione



POLO REGIONALE DI COMO

Corso di Laurea Specialistica in Ingegneria Informatica

USO DELL'ANALISI DELLE CORRISPONDENZE PER  
L'APPRENDIMENTO AUTOMATICO DI ONTOLOGIE E ALTRE  
INFORMAZIONI SEMANTICHE DAL TESTO

Relatore: Matteo Matteucci  
Correlatore: Davide Eynard

Tesi di laurea di: Fabio Marfia  
Matricola: 718843

Anno Accademico 2008/2009



*a Simone Molteni*



# Sommario

Nell'ampia disciplina dell'ingegneria della conoscenza è possibile identificare una importante sfida che impegna specialisti e ricercatori di differenti settori quali data mining, intelligenza artificiale, basi di dati, information management: essa consiste nell'attività di combinare in maniera opportuna statistica e logica per realizzare modelli atti a generare delle soddisfacenti rappresentazioni della conoscenza, ricevendo in input un (grande, tipicamente) insieme di dati più o meno strutturati, dotato di una propria semantica interna.

Nel presente lavoro di tesi mi sono occupato, per larga parte, di estendere le funzionalità di un tool software che utilizza il modello statistico dell'analisi delle corrispondenze per studiare le occorrenze di determinati termini in documenti di testo libero (informazione, dunque, non strutturata), al fine di generare delle gerarchie di concetti sotto forma di ontologie. Una parte finale del lavoro è invece dedicata allo studio di differenti potenzialità dell'analisi delle corrispondenze per generare un modello di rappresentazione della conoscenza diverso dalle ontologie, più vicino alla logica fuzzy e alle attività di inferenza umana.

La precisione nella generazione di gerarchie ontologiche del tool si attesta intorno al 60% per il miglior metodo automatico ed intorno al 90% per il miglior metodo assistito dall'utente. Anche il differente modello di conoscenza appare affidabile e la sua precisione nell'individuare particolari attributi di entità è valutata intorno al 70%.

# Abstract

In the broad area of Knowledge Engineering it is possible to identify an important challenge faced by researchers and practitioners of different sectors as Data Mining, Artificial Intelligence, Databases, Information Management: it is the activity of wise combining statistics and logics in order to set up models able to generate satisfying representations of knowledge, receiving as input a (typically large) set of more or less structured data, characterized by own internal semantics.

In the present work of thesis I deal, for the great part, with the extension of the functionalities of a software tool that uses the statistical model of Correspondence Analysis in order to analyze the occurrences of specific terms in documents of free text (so, characterized by an unstructured form of information), to generate hierarchies of concepts in form of ontologies. The latest part of this work, instead, deals with the study of different potentialities of Correspondence Analysis to generate a knowledge representation model different than ontologies, nearer to Fuzzy Logic and to the abilities of human inference.

Precision in the generation of hierarchies of the tool is attested to be around 60% for the best automatic approach and around 90% for the best human-assisted approach. Also, the different model of knowledge seems to present a good reliability and its precision in identifying specific attributes referred to entities was evaluated to be around 70%.



# Contents

<b>1</b>	<b>Introduction</b>	<b>11</b>
1.1	Formal Languages . . . . .	13
1.2	Ontologies . . . . .	13
1.3	The Unstructured Nature of Human Language . . . . .	16
1.3.1	My Little Wubbie . . . . .	16
1.3.2	Thus Spoke Papa Smurf . . . . .	19
1.3.3	Five Red Apples . . . . .	21
1.3.4	Summary . . . . .	23
1.4	Ontology Learning from Text . . . . .	23
1.5	Our Solution . . . . .	25
1.6	Identification of Fuzzy Propositions . . . . .	29
<b>2</b>	<b>State of the Art</b>	<b>33</b>
2.1	Ontology Learning from Text . . . . .	33
2.1.1	Ontology Learning from Text, Step by Step . . . . .	34
2.1.2	Concept Hierarchies Extraction . . . . .	38
2.1.3	Concept Relations Extraction . . . . .	60
2.1.4	Ontology Learning Algorithms Evaluation . . . . .	63
2.2	Inferring fuzzy semantics from text . . . . .	64
<b>3</b>	<b>Correspondence Analysis</b>	<b>67</b>
3.1	History . . . . .	68
3.2	The Framework . . . . .	69
3.2.1	Context . . . . .	69
3.2.2	First Passages . . . . .	70

3.2.3	The Rows Space . . . . .	70
3.2.4	The Columns Space . . . . .	79
3.2.5	Superimposition of the spaces . . . . .	81
3.3	Interpretation of Results . . . . .	83
3.3.1	Proximity between modalities of the same variable . . . . .	84
3.3.2	Axes . . . . .	85
3.3.3	Proximity between modalities of different variables . . . . .	85
3.4	Multiple Correspondence Analysis . . . . .	86
3.5	Final Comments . . . . .	86
<b>4</b>	<b>The Extraction Tool</b>	<b>89</b>
4.1	Extraction 1.0 . . . . .	89
4.1.1	User Requirements . . . . .	89
4.1.2	Main Conceptual Entities . . . . .	90
4.1.3	Use Cases . . . . .	93
4.1.4	Main Modules of the Application . . . . .	95
4.1.5	Execution Environment and Deployment . . . . .	100
4.2	Extraction 2.0 . . . . .	100
4.2.1	User Requirements . . . . .	100
4.2.2	Specification of the Interface . . . . .	100
4.2.3	Main Conceptual Entities . . . . .	101
4.2.4	Use Cases . . . . .	101
4.2.5	Main Modules of the Application . . . . .	103
4.2.6	Execution Environment and Deployment . . . . .	108
4.3	Extraction 2.0 - Verbal Version . . . . .	109
4.3.1	User Requirements . . . . .	109
4.3.2	Specification of the Interface . . . . .	110
4.3.3	Main Conceptual Entities . . . . .	111
4.3.4	Use Cases . . . . .	113
4.3.5	Main Modules of the Application . . . . .	113
4.3.6	Execution Environment and Deployment . . . . .	117
4.4	Considerations and Conclusions . . . . .	117

---

<b>5</b>	<b>Tests and Results</b>	<b>119</b>
5.1	Distributional Similarity View Evaluation . . . . .	120
5.1.1	2D-Plot Concepts Distribution against Documents . . . . .	120
5.1.2	2D-Plot Concepts Distribution against Attributes . . . . .	128
5.1.3	Comments . . . . .	133
5.2	Fionn Murtagh Hierarchy Generator Evaluation . . . . .	134
5.2.1	Tests . . . . .	134
5.2.2	Comments . . . . .	135
5.3	Hearst Patterns on Web Hierarchy Generator Evaluation . . . . .	136
5.3.1	Tests . . . . .	137
5.3.2	Comments . . . . .	139
5.4	Hearst Patterns on Web Hierarchy Generator Evaluation (Semi-Automatic Acquisition) . . . . .	140
5.4.1	Tests . . . . .	140
5.4.2	Comments . . . . .	142
5.5	Bootstrapping Hierarchy Extension Algorithm . . . . .	142
5.6	Boostrapping plus Hearst Patterns Hierarchy Extension Algorithm . . . . .	142
5.7	Euclidean Space of Terms and Attributes - 20 Nearest Attributes . . . . .	142
5.7.1	Tests . . . . .	143
5.7.2	Comments . . . . .	152
5.8	Euclidean Space of Terms and Attributes - Best Concept for a Group of Attitudes . . . . .	153
5.9	Summary and Conclusions . . . . .	155
5.9.1	Ontology Learning from Text . . . . .	155
5.9.2	Fuzzy Knowledge Model . . . . .	155
5.10	Performance Tests . . . . .	155
5.10.1	Extraction 2.0 . . . . .	156
5.10.2	Extraction 2.0 Verbal Version . . . . .	156
<b>6</b>	<b>Conclusions and Future Work</b>	<b>157</b>
6.1	Conclusions . . . . .	157
6.2	Future Work . . . . .	158

**Bibliography**

**161**

# Chapter 1

## Introduction

In the last years there has been a considerable increase in research on knowledge-based systems, especially in the context of the Semantic Web. Systems of this kind, as long as they numerate between their objectives something more than suppling trivial functionalities, suffer in their development process of the so-called *knowledge acquisition bottleneck*: it is the problem that creating large, usable, expandable and valid representations of semantics about a specific domain of interest often represents the most time-consuming task of the whole project. The cause of this stands in the fact that, being these structures supposed to represent a collection of semantics previously unknown to machines, they seem to submit to the necessity to be notated “by hand” by experts of their domain of interest.

In fact, machines often have at their disposal a huge amount of information carrying a lot of semantics, as databases, large corpora of documents, Web pages, and other kinds of sources. These repositories, anyway, are not manageable by semantic systems, because they are not notated in any standard for knowledge representation. The most important between these standards, at the current state of the art, is represented by *ontologies*: the Semantic Web community has developed, actually building on decades of research in knowledge representation, different standard languages for ontologies notation. The question is then: can not we, instead of manually generating standard ontologies from scratch, at least obtain from the unmanageable sources that we have at our disposal a *seed* ontology, a first approximate representation? We can then modify (eventually correcting wrong information) and expand it, considerably reducing the time requested for the generation of our structured knowledge.

While the acquisition of large ontologies from databases and other structured sources comes down to the exact problem of defining the right algorithm, the main challenge to face, that seems still nowadays to be more similar to an art than a science or an engineering problem, is *Ontology Learning from Text*. Free text ends up to be, in the last analysis, a peculiar repository of unstructured knowledge, and this asks researchers to adopt original heuristics in order to extract structured semantics. This approaches often return inaccurate results, and have definitely to be modified and validated by experts of the domain.

This thesis work presents the study and development of different solutions for the Ontology Learning from Text task, plus a derivate research for the generation of a different kind of representation of knowledge that involves fuzzy logic, more near to the

memorizing and reasoning abilities of the human mind. This model, even not having so much to do with Semantic Web, could be a solution for different problems of systems more near to the sphere of Artificial Intelligence.

The rest of this chapter is organized as follows:

- In §1.1 I define formal (or structured) languages
- In §1.2 I present the ontology standard for knowledge representation. It is not my intention to furnish an exhaustive definition, but just to take a deeper dive into some aspects of its nature that can be interesting with respect to the objectives of this work of thesis. For more information about ontologies, you can have a look at the large literature about the argument, maybe starting from the historical articles by Tom Gruber, [Gruber, 1992] and [Gruber, 1993]
- In §1.3 I analyze the natural human language that composes free text. The understanding of the aspects presented in this paragraph is very important to be aware of the problems both linked to methods and limits of the algorithms of Ontology Learning from Text and, as we will see, to a different problem faced by Knowledge Engineering, that is the challenge of Natural Language Understanding
- In §1.4 I explain the general approach to Ontology Learning from Text adopted by research, constituted in its most essential part by a wise application of statistical analyses of occurrences of specific terms in the documents, and by the application of some other logical heuristics in order to extract ontological propositions
- In §1.5 is a general presentation of the solution adopted in this thesis work for the Ontology Learning from Text task. It is in fact a collection of different solutions, all based on the statistical approach of *Correspondence Analysis*, developed by the French statistician Jean-Paul Benzécri. This work is for the most part inspired to some writings of Fionn Murtagh, a Computer Science professor who made his doctoral researches in Benzécri's statistics school. Fionn Murtagh presents his researches about the possibility to generate ontologies starting from the Correspondence Analysis technique applied to terms appearing in corpora of documents
- §1.6 is dedicated to the previous mentioned research that collaterally started from the principal Ontology Learning work: the software tools I came in contact with, during the development of the Ontology Learning algorithms, seemed to offer different powerful possibilities for the generation of other kinds of interpretation of statistical analyses over terms appearing in corpora of documents. It seemed to me a pity not to take advantage of such so powerful tools, and I used them to what it seemed their maximum potential to generate a representation of knowledge different from ontologies, that is, in last analysis, a collection of fuzzy logic propositions. I present in what way this model could be a good basis in trying to solve some problems dealing with Natural Language Understanding and other different challenges in the Artificial Intelligence research field

## 1.1 Formal Languages

Let us take a finite set of signs  $X$ , and a finite set of rules  $V$  that can be applied recursively and iteratively to ordinate sequences of the signs in  $X$  in order to obtain different sequences always composed by signs from the the same set  $X$ . An ordered sequence of signs is defined *proposition*, while the set of rules  $V$  is defined *formal grammar*. Infinite propositions can be derived from the first seed of signs by the infinite application of the rules: both these signs and these rules together are considered a *formal language*.

A proposition is defined well-formed against a formal language if it can be obtained with a finite number of iterative or recursive applications of the different rules over the signs of the language. For example, the  $\mathbb{N}$  set of natural numbers can be defined as a formal language where the finite  $X_a$  set of signs is:

$$X_a = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

and a set of rules according to our decimal system of enumeration is applied to the number  $n$  in order to derive the number  $n + 1$ .

A different language for the expression of the  $\mathbb{N}$  set of natural numbers is the ancient Rome numeral system. Here the finite set of signs  $X_r$  is defined as:

$$X_r = \{I, V, X, L, C, D, M\}$$

and a different set of rules is applied to combine these signs in order to derive the number  $n + 1$  from  $n$ <sup>1</sup>. I referred and I will continue to refer to these types of language both with the adjective of “formal” and “structured” indifferently, meaning by “structured” that they have at their disposal a finite and definite number of logical structures, that are the the grammar rules.

Noam Chomsky, linguist, philosopher, cognitive scientist, studied formal languages in the context of his work about generative grammars ([Chomsky, 1956], [Chomsky, 1957], [Chomsky, 1966]). In his article “Three Models for the Description of Language” [Chomsky, 1956] he specified for the first time the division of formal grammars in different categories, stating, in that context, that a language can be considered formal as long as it can be understood by a Turing machine. So, it is important to notice that the structured nature of a language is a necessary (and, according to Chomsky, sufficient) condition for any repository of semantics in order to be handled by a mere executor of algorithms such as our computers.

## 1.2 Ontologies

We can now have a look at the most general and standard model of formal knowledge organization in modern systems: the ontology. An ontology, in its simplest form, is composed by a set  $C$  of concepts and a set  $R$  of relations that can or can not stand between any

<sup>1</sup>There is to say that for very huge numbers romans added further signs, such as a little bar over a letter to specify a multiplication by 1000, and other conventions, anyway, this could change what I am saying just in the fact that more signs are to be added to  $X_r$ .





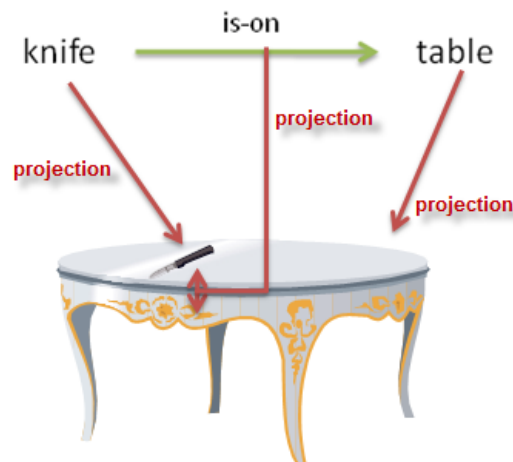


Figure 1.2: According to Ludwig Wittgenstein, the logical form of a proposition is an image of a relationship standing between real objects in the world

This assertion, at first, can leave us doubtful: a proposition, in fact, can not seem in any way an image of some facts in the world, just as we can easily conclude, for example, for a picture. There is no perceivable similarity between a proposition and the world, just as it is for a picture and the represented facts, a similarity, for example, of colours, shapes or lights. However, just as a picture is a representation “standing” for something in reality, in the sense that we can link every relationship between objects in the picture to a real existence of some facts in reality, in the same way a proposition can be linked to a real existence of facts thanks to its logical form (you can see the example in Figure 1.2).

Propositions, thanks to their logical form, can be considered “original” images, as for different reasons we can consider an original image a caricature or an impressionist picture. Are not we allowed to admit a semantic link between pictures like the one in Figure 1.3 and reality? And can not we do the same for images considered original in another sense, as propositions are? This process of link, of projection of the elements of the proposition towards the real world represents, according to Wittgenstein, the *sense* of the proposition. Propositions preserve their sense as long as this projection mechanism can be applied to them, no projection, no more sense.

So, according to these considerations, an ontology could be just defined as a portrait of a domain of knowledge, just as Figure 1.3 can be defined a picture of a harbour at sunrise.

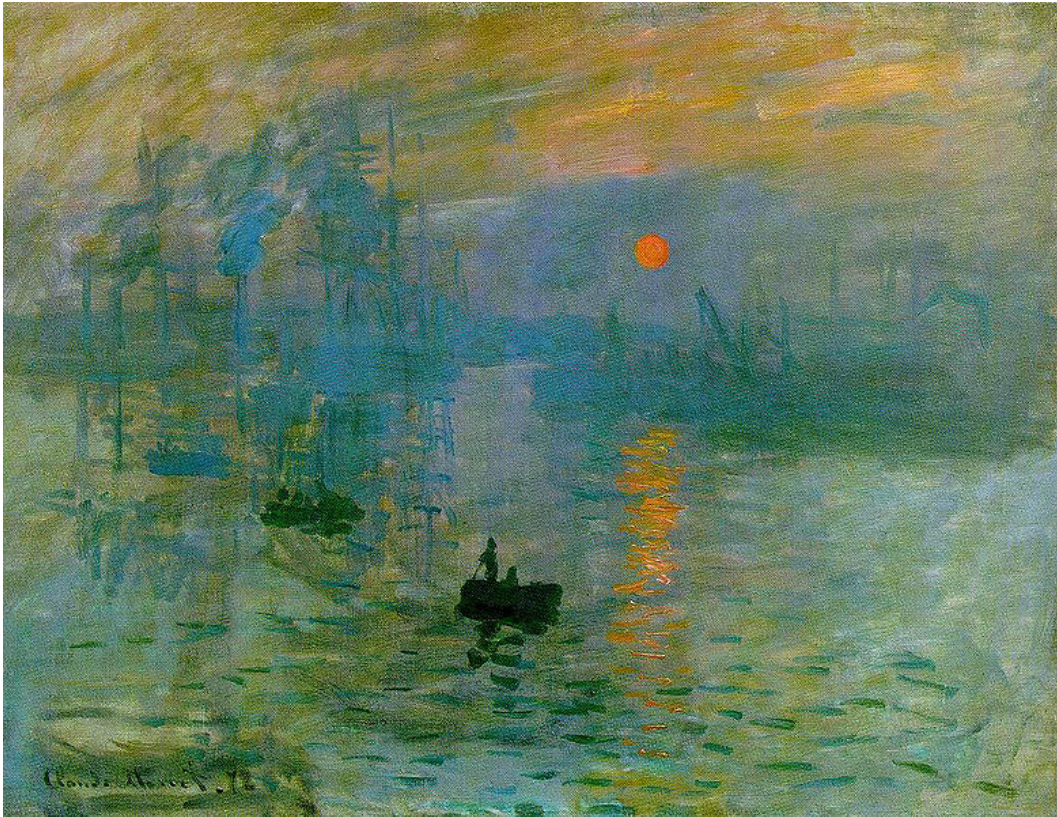


Figure 1.3: Claude Monet, Impression, Sunrise (1872), oil on canvas, Musée Marmottan - according to Ludwig Wittgenstein, propositions are images of reality not so different from the shapes we can see in this picture

## 1.3 The Unstructured Nature of Human Language

### 1.3.1 My Little Wubbie

The arguments of the last paragraph should have generated some reflections, at least between people more in touch with linguistics. I want to call the attention in particular to one of the assertions. I cite myself, from just some rows above:

“...This process of link, of projection of the elements of the proposition towards the real world represents, according to Wittgenstein, the *sense* of the proposition. Propositions preserve their sense as long as this projection mechanism can be applied to them, no projection, no more sense.”



Figure 1.4: Andy Milonakis is speaking to his dog in a scene from MTV's Andy Milonakis Show

This could work right when we talk about ontologies, but what about natural language? Is it still right, for every sentence of any human language, the assumption “No projection, no more sense”? There is to say that Wittgenstein, in fact, in [Wittgenstein, 1919], could not know anything about computer science ontologies and he was speaking about human language in its totality.

Probably everyone, at this point, should be able to express at least one sentence that has nothing to do with this mechanism of projection towards reality. Let us see an example to be kept as reference and that could be more interesting than others: a TV scene titled “My Little Wubbie”, extracted from the *Andy Milonakis Show*, produced by the American cable television MTV in 2005<sup>3</sup>. The star, Andy Milonakis, is speaking to his dog:

“You’re my little pupie woopie shnuppie pupie luppie  
zubbie zubbie pupie bubbie bubbie zuppie,  
kuky kuky kuky kuky pupie  
pupie pupie, you’re my little zubbie zubbie...”

*(the scene fades out)*

It is obvious that semantic projection towards reality of terms such as “woopie”, “bubbie” and “shnuppie” is quite inadvisable. It is as much obvious that no one (at least, no

<sup>3</sup>A video of the scene can be found at [Milonakis, 2005], but note that it could be removed, due to copyright reasons

one living in our occidental society), looking at the video, would say: "I can not understand what he is doing". So, Andy Milonakis words do have a sense and this sense is understandable.

So, what? Was Wittgenstein wrong? Well, in fact, yes. Many criticisms and examples of the type just showed were submitted to Wittgenstein after 1919, and he was able to revise his considerations about natural language in different lessons and notes merged into the posthumously published book "Philosophical Investigations" [Wittgenstein, 1953]. In this book is expressed, in my own opinion, the most guessed and in-depth explanation ever of what natural language is, showed in fact as something different from a formal and structured language. In these writings not just the mechanism of projection is abandoned, but also the formalism and the structuralism of his first studies.

The reflections about logical structures of the language seem anyway to go quite well for our structured ontologies, and we can state, with a reasonable certainty, that language as was thought in Wittgenstein's Tractatus [Wittgenstein, 1919] coincides exactly with our notion of ontology: a formal and logical expression of facts. This allow us to say that everything asserted in §1.2 is still valid, thanks to the fact that everything referred to language by Wittgenstein in the Tractatus can be referred now to our notion of ontology.

What about Noam Chomsky? We left him in his researches about formal grammars, but, already in [Chomsky, 1956], he asked himself whether English grammar could be or not considered formal. He reports in the abstract of the article:

"We investigate several conceptions of linguistic structure to determine whether or not they can provide simple and "revealing" grammars that generate all of the sentences of English and only these. We find that no finite-state Markov process that produces symbols with transition from state to state can serve as an English grammar."

That, simply, means: there is no finite set of rules  $V_e$  that can serve as an English Grammar, so, English language is not formal. And our  $X_e$  finite set of signs? Well, at least for written English text, the ASCII standard table should be considered the right finite set of signs we need to communicate, even if, probably, identifying a set  $X_e$  for a spoken language is not as much easy. Anyway, to at least close the discussion about this latter set, let us assume, as a necessary condition for a communication in the English language, that this communication has to be expressible by signs belonging to the US standard ASCII table (shown in Figure 1.5).

Let us go back to Chomsky: if English grammar is not a generative grammar, what is it? Chomsky spent a lot of research to answer to this question, never arriving, in my own opinion, to a general satisfying vision of the problem as Wittgenstein did just some decades before. This consideration, in fact, is strong and has a complex background, and it would request a lot of discussion to be confirmed or denied.

Anyway, I present now two other examples selected to explain why English (as every natural language) cannot be considered formal, being that its grammar set of rules  $V_e$  (as every natural language's set of rules) can not be considered finite. I present, together with my arguments, some between the central Wittgenstein Philosophical Investigations' considerations about natural language, preferring to ignore the work of Noam Chomsky about the same matter. More information about Chomsky's researches on these argu-

Regular ASCII Chart (character codes 0 - 127)

000d 00h	(nul)	016d 10h	► (dle)	032d 20h	sp	048d 30h	0	064d 40h	@	080d 50h	P	096d 60h	`	112d 70h	p
001d 01h	☉ (soh)	017d 11h	◄ (dc1)	033d 21h	!	049d 31h	1	065d 41h	A	081d 51h	Q	097d 61h	a	113d 71h	q
002d 02h	(stx)	018d 12h	↑ (dc2)	034d 22h	"	050d 32h	2	066d 42h	B	082d 52h	R	098d 62h	b	114d 72h	r
003d 03h	♥ (etx)	019d 13h	!! (dc3)	035d 23h	#	051d 33h	3	067d 43h	C	083d 53h	S	099d 63h	c	115d 73h	s
004d 04h	♦ (eot)	020d 14h	‡ (dc4)	036d 24h	\$	052d 34h	4	068d 44h	D	084d 54h	T	100d 64h	d	116d 74h	t
005d 05h	♣ (enq)	021d 15h	§ (nak)	037d 25h	%	053d 35h	5	069d 45h	E	085d 55h	U	101d 65h	e	117d 75h	u
006d 06h	♠ (ack)	022d 16h	■ (syn)	038d 26h	&	054d 36h	6	070d 46h	F	086d 56h	V	102d 66h	f	118d 76h	v
007d 07h	• (bel)	023d 17h	⚡ (etb)	039d 27h	'	055d 37h	7	071d 47h	G	087d 57h	W	103d 67h	g	119d 77h	w
008d 08h	▣ (bs)	024d 18h	↑ (can)	040d 28h	(	056d 38h	8	072d 48h	H	088d 58h	X	104d 68h	h	120d 78h	x
009d 09h	(tab)	025d 19h	↓ (em)	041d 29h	)	057d 39h	9	073d 49h	I	089d 59h	Y	105d 69h	i	121d 79h	y
010d 0Ah	(lf)	026d 1Ah	(eof)	042d 2Ah	*	058d 3Ah	:	074d 4Ah	J	090d 5Ah	Z	106d 6Ah	j	122d 7Ah	z
011d 0Bh	♂ (vt)	027d 1Bh	← (esc)	043d 2Bh	+	059d 3Bh	;	075d 4Bh	K	091d 5Bh	[	107d 6Bh	k	123d 7Bh	{
012d 0Ch	♀ (np)	028d 1Ch	~ (fs)	044d 2Ch	,	060d 3Ch	<	076d 4Ch	L	092d 5Ch	\	108d 6Ch	l	124d 7Ch	
013d 0Dh	(cr)	029d 1Dh	-- (gs)	045d 2Dh	-	061d 3Dh	=	077d 4Dh	M	093d 5Dh	]	109d 6Dh	m	125d 7Dh	}
014d 0Eh	↓ (so)	030d 1Eh	* (rs)	046d 2Eh	.	062d 3Eh	>	078d 4Eh	N	094d 5Eh	^	110d 6Eh	n	126d 7Eh	~
015d 0Fh	□ (si)	031d 1Fh	▼ (us)	047d 2Fh	/	063d 3Fh	?	079d 4Fh	O	095d 5Fh	_	111d 6Fh	o	127d 7Fh	~

Extended ASCII Chart (character codes 128 - 255; Codepage 850)

128d 80h	Ç	144d 90h	Ë	160d A0h	á	176d B0h	¸	192d C0h		208d D0h		224d E0h		240d F0h	-
129d 81h	ü	145d 91h	¸	161d A1h	í	177d B1h		193d C1h		209d D1h		225d E1h		241d F1h	±
130d 82h	é	146d 92h	¸	162d A2h	ó	178d B2h		194d C2h		210d D2h		226d E2h		242d F2h	-
131d 83h	á	147d 93h	ó	163d A3h	ú	179d B3h		195d C3h		211d D3h		227d E3h		243d F3h	
132d 84h	ä	148d 94h	ö	164d A4h	ñ	180d B4h		196d C4h		212d D4h		228d E4h		244d F4h	
133d 85h	à	149d 95h	ò	165d A5h	Ñ	181d B5h		197d C5h		213d D5h		229d E5h		245d F5h	
134d 86h	â	150d 96h	û	166d A6h		182d B6h		198d C6h		214d D6h		230d E6h		246d F6h	
135d 87h	ç	151d 97h	ù	167d A7h		183d B7h		199d C7h		215d D7h		231d E7h		247d F7h	
136d 88h	è	152d 98h	ý	168d A8h		184d B8h		200d C8h		216d D8h		232d E8h		248d F8h	
137d 89h	ë	153d 99h	ÿ	169d A9h		185d B9h		201d C9h		217d D9h		233d E9h		249d F9h	
138d 8Ah	è	154d 9Ah	ÿ	170d AAh		186d BAh		202d CAh		218d DAh		234d EAh		250d FAh	
139d 8Bh	í	155d 9Bh	ø	171d ABh	¼	187d BBh		203d CBh		219d DBh		235d EBh		251d FBh	
140d 8Ch	î	156d 9Ch	£	172d Ach	½	188d BCh		204d CCh		220d DCh		236d ECh		252d FCh	
141d 8Dh	ï	157d 9Dh	£	173d ADh		189d BDh		205d CDh		221d DDh		237d EDh		253d FDh	
142d 8Eh	Ë	158d 9Eh		174d AEh		190d BEh		206d CEh		222d DEh		238d EEh		254d FEh	
143d 8Fh	Ë	159d 9Fh	f	175d AFh		191d BFh		207d CFh		223d DFh		239d EFh		255d FFh	

Figure 1.5: US-ASCII table of roman characters can be considered an exhaustive  $X_e$  set of signs of English natural language

ments can be found in his works about transformational grammars, [Chomsky, 1965], and I-Language and E-Language, [Chomsky, 1986].

### 1.3.2 Thus Spoke Papa Smurf

Let us take another example from the TV world, an argument quite beloved by our Italian most famous living semiotician, Umberto Eco (see, e.g., [Eco, 1997]): the strange case of smurfs' language.

“The Smurfs” is a cartoon created for the first time in 1958 by the Belgian illustrator Pierre Culliford. Smurfs are little blue forms of life that speak the same language as the humans, with just the little bad habit to change at will, with no clear or defined rule, nouns, adjectives and verbs in their speech with the word “smurf”. Let us see an example: here the smurfs' village chief, papa smurf, is speaking with Clumsy Smurf, that was transformed in a little green dragon by a potion and wants to turn back to its normal aspect:





Figure 1.6: “The Smurfs” (1981) - Clumsy Smurf transformed into a little dragon, and Papa Smurf

*Papa Smurf: Try this...  
(the remedy given by Papa Smurf doesn't work)*

*Papa Smurf: We'll have to smurf something else...  
(Papa Smurf tries different remedies)*

*Papa Smurf: It's no use Clumsy, none of my potions seems to work!*

*Clumsy Smurf: I'll be this way for the rest of my... smurf?*

*Papa Smurf: No no no, I'll find another smurf, go home, I'll call you!*

Someone may comment: And so? Do humans talk like that? What is important to notice here is that humans are *perfectly able* to understand the meaning of every different sentence in the dialogue. This is a very particular example because here signs are changed, but the proposition keeps exactly the same sense. From what formal language we would expect something like this?

The fact here is that we use our past experiences and our deep knowledge of the rules to make the machine work even when fed with wrong data. And we do it in many occasions, not just following the Smurfs' cartoon. How far can this ability go? Very far, in fact, the propositions of the example dialogue are simple, but we can do a lot more<sup>4</sup>. Turing machines cannot do anything like that, and they will not, while these mechanism will not be better understood. This is in fact a great limit in their ability to understand natural language.

<sup>4</sup>I do not go into more depth, for more information about these arguments you can see [Eco, 1997], or any writing about human inference abilities, for example, Charles Sanders Peirce's articles like [Peirce, 1883] (Peirce is a mathematician and philosopher, founder of the American pragmatism).

From the first time that smurfs were dubbed in English language, a set of rules  $V_{smurf}$  has to be considered as part of  $V_e$ , in order to comprise the usage of this word in the English language (that *has* to be comprised, because we understand its sense). But what would these rules be? Are they finite or infinite? Let us try for example to summarize the smurf's habit with the subsequent rule:

"You can change every noun, verb and adjective in a sentence with the word "smurf" and its sense does not change"

Could this work well? Let us see a well-formed sentence, obtained applying this rule to the English grammar:

"Smurf, smurf me that smurf!"

It can not work. This is not allowed because the sentence is inunderstandable. The rule should be something like:

"You can change every noun, verb and adjective in a sentence with the word "smurf", preserved that sentence's sense is still understandable"

Well, right, but this is not a formal rule, because a formal system has not, by default, the elements to establish whether the sentence is still understandable or not. There is to say that somebody, with a huge amount of work on smurfs' linguistics, could be able to determine a large, but finite number of rules that state with certainty when a sentence preserves its sense and when it does not after a *smurfreplacement*, and these rules, with the "You can make your changes, preserved that sentence's sense is still understandable" rule, could be added to the set of rules  $V_e$ .

We can not assert with certainty that this could not be done. So,  $V_e$  has to be, up to now, with some difficulties in fact, still considered a possible finite set. However, it will not be anymore at the end of the next paragraph.

### 1.3.3 Five Red Apples

In the previous examples we saw a man speaking to his dog, and some fantastic characters of a cartoon. Is not it possible that, at least, an ordinary communication between ordinary people can be considered formal? Well, the next example does not come from TV and can be considered a very common scene in our ordinary life. It is directly taken from Wittgenstein's *Philosophical Investigations* [Wittgenstein, 1953]:

"Think of the following use of language: I send someone shopping. I give him a slip marked 'five red apples'. He takes the slip to the shopkeeper, who opens the drawer marked 'apples', then he looks up the word 'red' in a table and finds a colour sample opposite it; then he says the series of cardinal numbers (I assume that he knows them by heart) up to the word 'five' and

for each number he takes an apple of the same colour as the sample out of the drawer. It is in this and similar ways that one operates with words, "But how does he know where and how he is to look up the word 'red' and what he is to do with the word 'five'?" Well, I assume that he 'acts' as I have described. Explanations come to an end somewhere. But what is the meaning of the word 'five'? No such thing was in question here, only how the word 'five' is used."

What Wittgenstein is showing us is something very important to our scopes: this is an example of usage of English language in a *different logical form at all*, with a seriously different set of rules  $V_s$  from the English set of rules  $V_e$ . We still have to admit that English language is used, because a slip with the words "five red apples" as signs is passed and a use of these signs is made, but there is no standard English grammar in it, we see another conventional grammar generated according to the usage of the message.

Someone may say: "Look! Five, red. . . adjectives posed before the noun apple!". Well, let us just change the slip in the example, with another reporting the message: 'apple, five, red', does not the example still work well? If the shopkeeper understands it in relation to a convention, and answers to the message with the same acts, 'apple, five, red' is an English communication as well, and its original grammar is a grammar as well.

How many situations like this are in the world? Potentially infinite. Every set of rules  $V_{s_i}$  of the situation  $i$  has to be added to the set of rules  $V_e$  in order for this latter to comprise the totality of the usages of the English language, and, so, English grammar set of rules  $V_e$  is infinite, as it is every natural language grammar set  $V_{i_i}$  for the same reasons.

A group of people, signs, rules of interpretation of signs and the context in which they operate is defined by Wittgenstein as *linguistic game*. The last example is, in fact, a simple example of a linguistic game: two people, some signs on a slip, an interpretation and usage of this signs, the context of the shop. We have arrived to the general interpretation of natural language by Wittgenstein: not just a single grammar, but infinite linguistic games that involve different people, contexts, rules and signs, that take form and sense in function of a practical usage. It is not a case that Wittgenstein uses the definition of "game": it refers both to the presence of pre-defined rules and to the fact that humans are always ready to change them, *playing* with words and grammatical rules, *enjoying* the fact that they are acting in this manner.

According to this, we can try to re-interpret the other examples of this chapter: Andy Milonakis, shown in §1.3.1, is playing a linguistic game in the presence of his dog, that gains sense in function of the usage he is making of the words, that is expressing in some way the internal state of mind in which his dog puts him. Also, Smurf's smurfreplacement act (§1.3.2) is another linguistic game, which our little blue friends enjoy themselves to play.

A thought that comes from these considerations is that probably we will not be able to create machines able to handle the natural language well while we will not be able to create them as able to *enjoy* the act of using it. How this could be done, anyway, goes far beyond the objectives of this thesis work.



### 1.3.4 Summary

We can finally summarize the main features of natural language propositions, compared with the ontological propositions' ones, in a table:

Ontological proposition	Natural Language proposition
Formal	Informal
Structured	Unstructured
Finite set $X$ of signs	Finite set $X$ of signs (with some reservations)
Gain sense as an image of a state of facts in the world	Gain sense in relation to people, contexts and a usages
Shared and explicit	In relation to people and contexts

## 1.4 Ontology Learning from Text

Ontology Learning from Text is a field of research considered part of the Information Retrieval discipline, in the largest Computer Science sector of Knowledge-Based Systems. In the light of the last paragraphs considerations, we can define Ontology Learning from Text as the art to subsume ontological proposition from collections of natural language propositions. The table in §1.3.4 could discourage us, because the differences are many and deep. But there is to notice that, even if, in a last analysis, the natural language that composes free text is unstructured, it usually follows (always ready to, so to speak, go off the lines) a set of descriptive rules that are the *grammar* and the *syntax* of the language, as we learnt them since our primary schools. There are algorithms that try to identify the grammar and syntax roles of a word in a sentence, they are part of the discipline in Computer Science called *Natural Language Processing* (NLP).

So, at least, from our modest Turing machine's point of view, it seems that we can try to understand what is understandable, being expressed according to these grammatical and syntactic rules, and simply discard the rest. This approach leads to some drawbacks that have to be challenged:

1. Informal structures misunderstood for formal structures
2. Formal structures not recognized, and then discarded
3. Proposition acquired not an image of a state of facts in the world, but stored as it were
4. Proposition valid only for specific people and contexts, acquired and stored as shared and explicit

A remedy to overcome point 4 is the so-called domain specificity of the corpus of documents: Ontology Learning from Text algorithms are used to extract information from corpora of documents dealing with very specific domains of knowledge (e.g., tourism,

biology, chemistry): this make us expect from the propositions found in the documents to express a knowledge that can be considered shared and explicit at least between all the practitioners of that specific domain.

In order to analyze point 3, let us take back our examples, Andy Milonakis' elegy of a dog beginning in §1.3.1 was:

“You're my little pupie...”

A formal algorithm of Ontology Learning from Text will probably extract from this situation the relation: *is-a*(dog, pupie)<sup>5</sup>, but pupie has no sense, that is our point 3. And imagine Ontology Learning from Text algorithms parsing a corpus of documents written by some smurfs (§1.3.2), such an ambiguous concept would be “smurf”! That is point 3 again.

To overcome this problem, Ontology Learning from Text algorithms always start identifying, by following some statistical rules, terms that should be considered relevant for the message carried by a  $D$  corpus of documents. Heuristics to determine whether a term is relevant or not are based on a statistical analysis of occurrences distributions of terms, providing a score of importance to a term in a corpus. This score is often evaluated in its difference from the score of the same term obtained from a reference corpus  $D_r$ , in order to obtain a final relative score which observation can overcome the problem of terms that result often important in every corpus of documents (this can happen with relative pronouns, common adverbs, verbs as “to be” or “to have”...).

The relative score evaluation can overcome even the “smurf” problem: being “smurf” probably very important in every document, its relative importance is low and the term is discarded. For terms like “woopie”, instead, we expect them, carrying no semantic, not to be repeated so much in our  $D$  corpus and so terms like this should be discarded too, receiving a low score of importance. You can understand that the  $D_r$  corpus must be chosen with care, in order to obtain the right relative importance measure for the terms, we can say, in last analysis, that this measure of relative importance is the measure on how much a term contributes to distinguish the  $D$  corpus from the  $D_r$  corpus. So, different  $D_r$  can generate very different scores.

A set  $T$  of  $n$  relevant terms is then obtained, selecting the  $n$  terms with the highest score. At this point, set  $T$  is sometimes pruned according to other heuristics, then the Ontology Learning work can begin, and different approaches are applied in order to retrieve from text (sometimes with a help from other more or less structured sources of knowledge) as much ontological information as possible about our relevant concepts.

Remedies to overcome points 1, 2 are specific to every approach, we can generally say that just some patches can be applied to the algorithms in order to avoid drawbacks like these, that irreparably lower the validity of the obtained ontologies. A deeper examination of these aspects can be found in the presentation of many different approaches to the Ontology Learning from Text task in Chapter 2, State of the Art, in §2.1. You will see that every solution is fundamentally based on a wise combination of statistics and logics, plus often algorithms of NLP.

---

<sup>5</sup>Just imagine that the information about the fact that “you” is referred to the dog is found somewhere, it is not relevant for our discussion

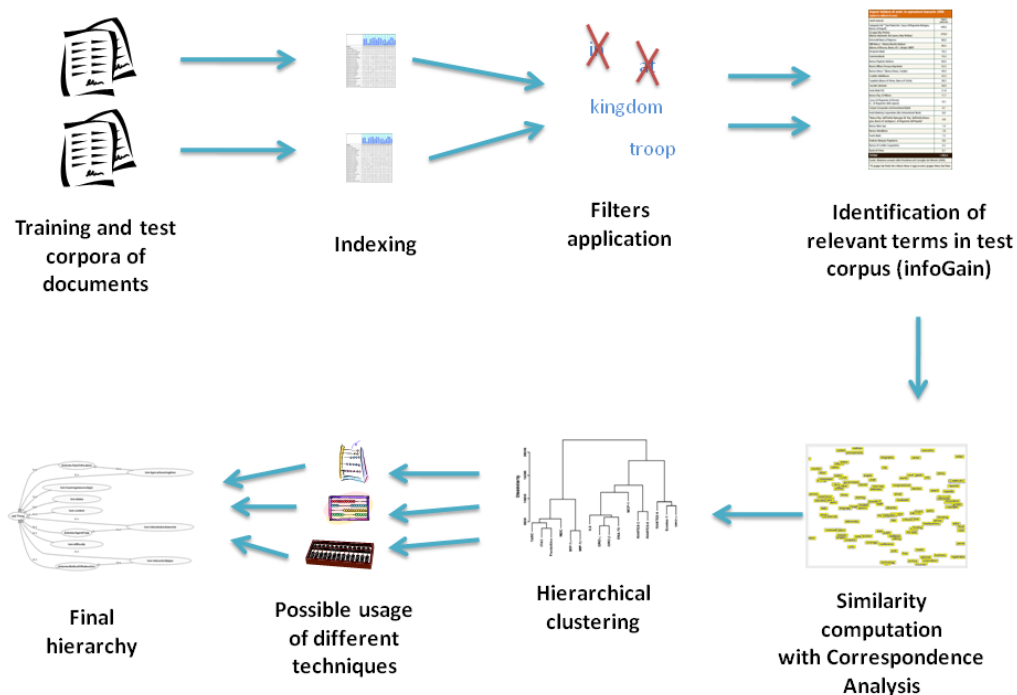


Figure 1.7: Main passages of our solution to the Ontology Learning from Text task

## 1.5 Our Solution

In 2009 Davide Eynard ended his PhD in Computer Science and Engineering at Politecnico di Milano, and he received in 2009 a research grant in order to develop a software tool that could instantiate the conceptual model presented by Fionn Murtagh in [Murtagh, 2005] and [Murtagh, 2007] for the generation of concept hierarchies from unstructured text.

A first version of the program was ready by September, 2009. I then extended the functionalities of the tool as thesis work. The program in its different versions is presented in Chapter 4, The Extraction Tool. I summarize here the main steps of the model (you can see them in Figure 1.7). This model belongs, in the large field of the different Ontology Learning from Text approaches, to the subgroup of *Distributional Similarity* approaches, presented in §2.1.2.3:

### Corpora of Documents

A corpus of documents is thought as a unitary set of free text files. The tool asks to be fed with two corpora of documents: a training corpus, that holds the role of the referential corpus, (the set  $D_r$  of §1.4), and a test corpus, that is the main corpus from which the relevant terms are to be extracted.

### Indexing

The two Corpora are parsed by an indexer, in order to be rapidly handled by the tool. The indexing process is usually very time-consuming, so, the user is allowed to execute the indexing task of a single corpus just one time and then feed the tool with the index obtained as many times as she like.

### Filters Application

During the indexing phase, filters can be applied in order to select terms in function of some needs, for example NLP algorithms can be used to discard everything but the nouns, or terms composed by less than  $n$  letters can be discarded (usually, very short terms rarely represent relevant concept, even if there are some exception to this consideration, for instance: acronyms).

### Identification of Relevant Terms in Test Corpus

For every term an InfoGain score is computed, that is a measure of the relative entropy of the term (for an exhaustive explanation you can see §4.1.4.3). It is a score measure computed according to the principles explained in §1.4, it preserves terms that more contribute to distinguish the test corpus from the training corpus. A set  $T$  of  $n$  (with  $n$  specified by the user) relevant terms is collected.

### Similarity Computation

The approach of Correspondence Analysis is used to project the relevant terms in a 2-dimensional euclidean space according to their distributional behavior over the documents. The nearer two terms are in this plot, the more their distributional behavior over the documents of the corpus is similar (an example of this plot is shown in Figure 1.8). The concept of distributional behavior, how Correspondence Analysis can create this plot, all the information and implications about the model and its results are explained in Chapter 3, Correspondence Analysis.

The more the distributional behavior over documents of two terms is similar, the more we are allowed to infer a semantic similarity between the terms. This is the basic concept of Distributional Similarity, and it is based on the so-called Harris *distributional hypothesis* [Harris, 1968]:

“Words are similar to the extent that they share similar context”

Harris is saying: in what sense humans consider an orange more similar to a pomegranate than to a pen? The answer is that pomegranates and oranges share similar contexts more frequently than oranges and pens. So, expressing the similarity of behavior of occurrence

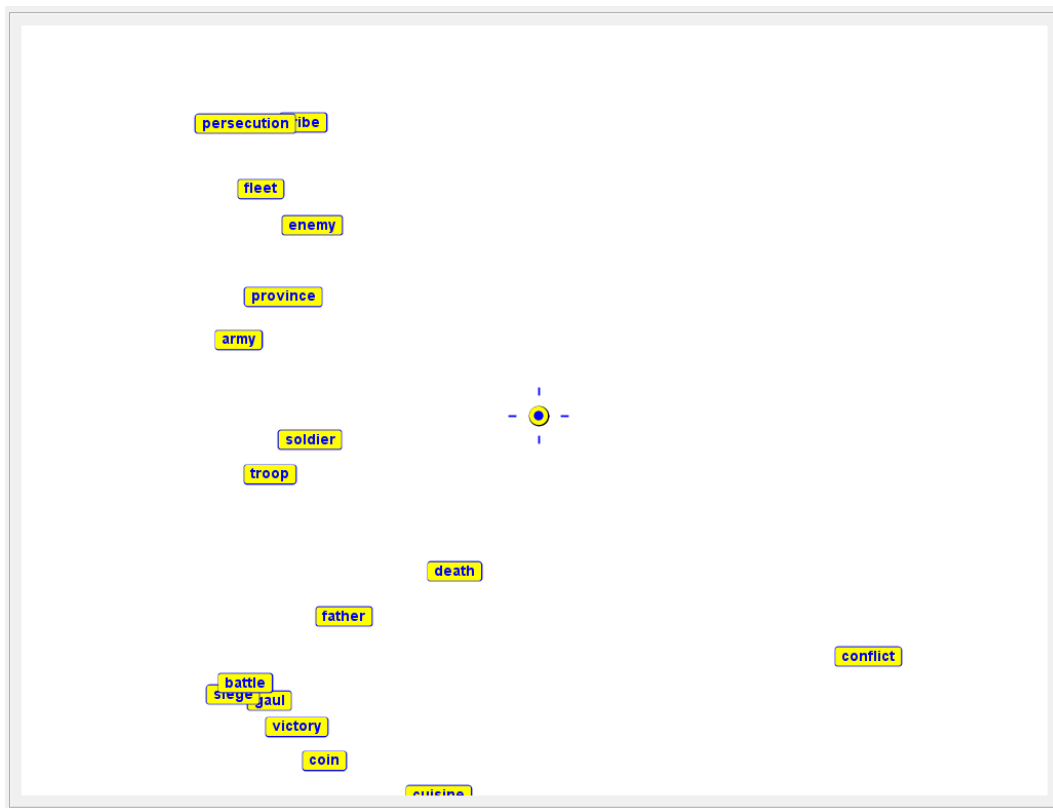


Figure 1.8: An example of Correspondence Analysis projection, from a corpus of documents about Roman Empire

of two terms with in respect to a specific type context (in this case, the document) means that we are expressing their semantic similarity. So, the more two terms are near in the plot obtained by the Correspondence Analysis framework, the highest should be the semantic similarity between them, according to the information carried by the  $D$  corpus.

### Hierarchical Clustering

A Hierarchical Clustering tree is created defining the terms' proximities in the tree according to their proximity in the 2-dimensional plot. The nature of this tree and the algorithms for its creation are presented in §2.1.2.3 and §4.1.4.5. An example of a Hierarchical Clustering tree is shown in Figure 1.9.

We can state that this tree carries a not so much different information from the 2-dimensional plot, it is just a more immediate representation of the semantic proximities between terms.

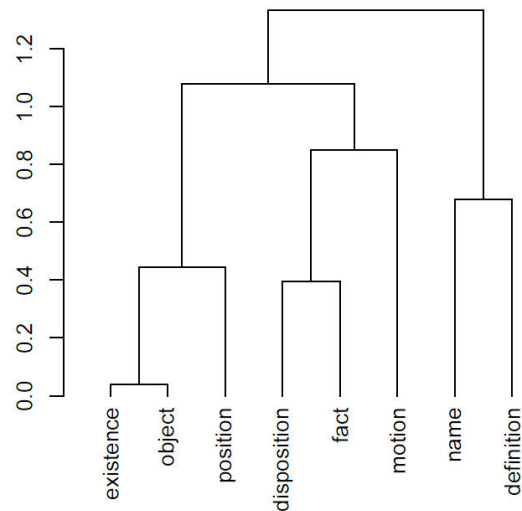


Figure 1.9: Example of Hierarchical Clustering of terms extracted from a Plato’s dialogue

### Different Techniques for the Creation of the Hierarchy

Here the user can choose between different algorithms to be applied for the creation of the hierarchy. They are presented in §4.2.5.6. Each of them is inspired to another work presented in Chapter 2: in fact, none between these algorithms is an original technique. Anyway, what is original is that these techniques were never applied to statistical analyses obtained by the Correspondence Analysis model, but were fed with the results of other statistical approaches.

Being Ontology Learning for Text an approximate activity, and not an exact task, the user can hardly foresee what is definitely the best solution that will extract the more usable and valid ontology from his corpus of documents. So, the main objective of this thesis work, with respect to the Ontology Learning from Text task, is to offer a valid and versatile alternative to other approaches, allowing the user to try different possibilities, maybe finding between them the best solution, according to his necessity.

### The Final Hierarchy

A hierarchy of concepts is finally obtained, as the one in Figure 1.10. Someone may wonder now where the ontology is: well, a concept hierarchy is defined (as better explained in §2.1.1) as a particular type of ontology, expressing relations of subsumption between concepts. And, for what may concern ontologies in this thesis work, there is nothing more. No algorithms to generate ontologies with different types of relation were attempted.

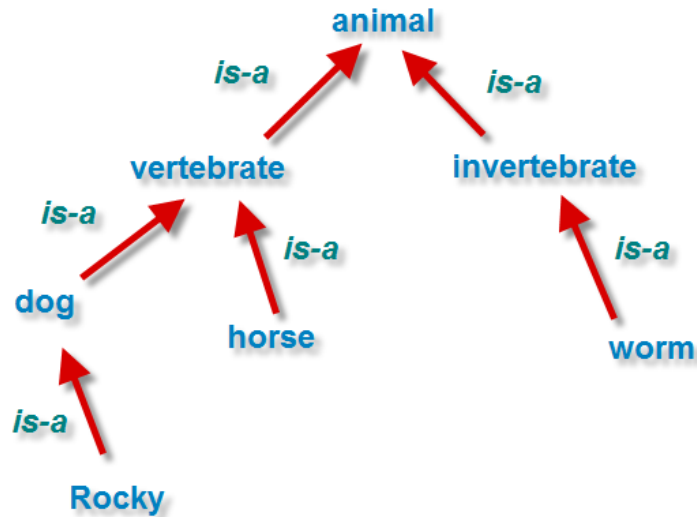


Figure 1.10: Example of a hierarchy of concepts

Anyway, in the next paragraph a different and (as far as we know) original attempt to extract a different type of semantics from corpora of documents is presented. This model uses the same instruments at our disposal for the extraction of concept hierarchies (e.g. CA framework, NLP algorithms), but with a different approach.

## 1.6 Identification of Fuzzy Propositions

In 340 BC Aristotle wrote his “Categories” [Aristotle, 340], a treatise with the objective to classify all the possible propositions that can be referred to a being. This arguments are still today referential for many works on logics, in particular in works like this dealing with ontologies, because Aristotle divided all the possible propositions that can have as subject a being in the world in 10 different categories, so, being ontologies a collection of propositions about facts in the world, every ontological relation should be comprised in one of the ten categories.

In this paragraph we are interested in particular in two of these categories of attributes:

- **Action:** what a being can do
- **Affection:** what can be done with a being

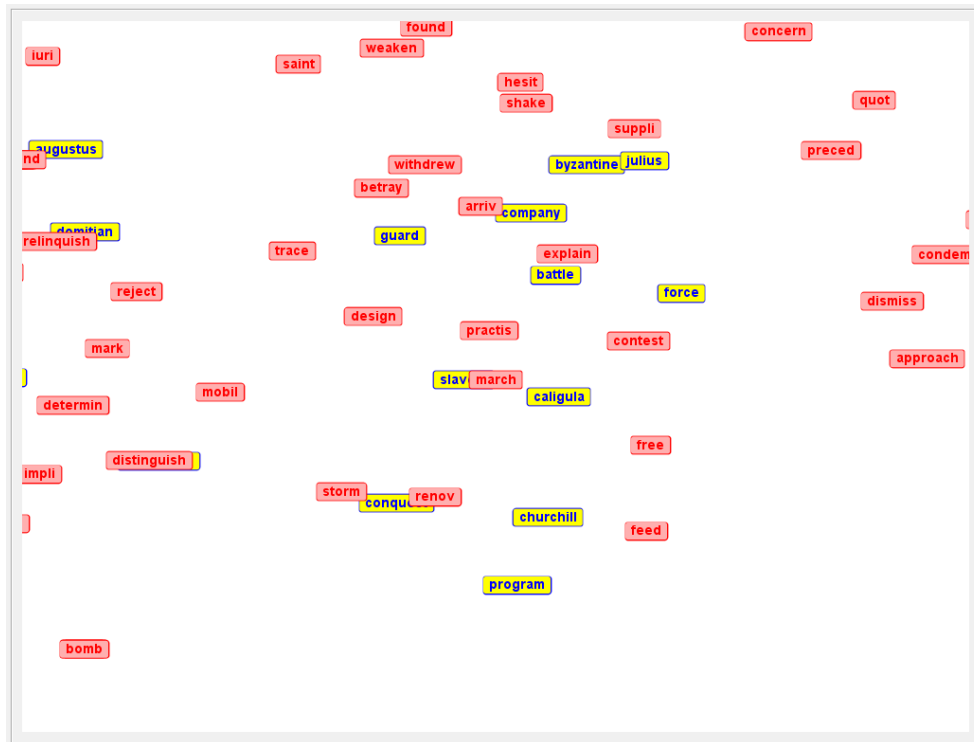


Figure 1.11: Example of a 2-dimensional plot of nouns (in blue over yellow) and action attributes (in red over pink), the attributes are verbs that underwent a stemming transformation, in order to reconduct verbs of different sentence, mood, voice and aspect to the same root. The corpus of documents parsed is about Roman Empire

The idea is to use NLP algorithms over a corpus of documents in order to compile a table of occurrences of nouns seen to do or undergo determinate actions, like this:

	march	conquer	devastable	...
caligula	34	120	0	...
city	0	10	120	...
troop	200	160	4	...
...	...	...	...	...

The table is saying, for instance, that “caligula” appears in the parsed corpus of documents 34 times as subject of the verb “march” and 0 times as object of the verb “devast”. Correspondence Analysis is able to project all the nouns and verbs in a table like this in an Euclidean space of 2, or, in fact, even more dimensions (the more the dimension, the more the precision of the information carried by the graph, see §3.2.3.2), an example of 2-dimensional representation can be seen in Figure 1.11.



This representation lends itself to three different kind of interpretations:

- **Semantic similarity between nouns:** the analysis of the distribution of nouns in respect to done or undergone actions can furnish information about their semantic similarity. According to Aristotle's Categories, the more two beings share the same attributes (in this case, action and affection are analyzed), the more they can be classified as similar. Also, attributes can be considered a context according to Harris distributional hypothesis presented in §1.5. So, the more two nouns are seen near in the plot, the more we expect a semantic similarity between them. This is not so much different from what was done in the previous paragraph, but the distribution analysis here is conducted in respect to verbs of sentences and not in respect to documents, furnishing a different and, I think, complementary view of similarity. Again, we offer different alternatives, the user can try the ones she prefers, in order to find the best solution.
- **Semantic similarity between verbs:** as said for the nouns, we expect that the more two verbs share a similar distribution in respect to the nouns, the more they can be considered semantically similar.
- **Cross-distance between a noun and a verb:** according to Correspondence Analysis (as explained in §3.2.5), the distance between a noun and a verb is significant in the sense that the more an attribute (action or affection) is seen to be referred to a noun in the corpus of documents, the more the noun is expected to be near to that attribute in the Euclidean space.

The last point represents the more interesting and innovative information that Correspondence Analysis provides in this context: from a huge corpus of documents we can obtain a  $n$ -dimensional synthetic maps<sup>6</sup> of beings expressed by the corpus, and of actions done and undergone by that beings.

This is the brand new idea for a new representation of knowledge: an  $n$ -dimensional Euclidean space of beings and attributes referred to these beings, automatically obtained by parsing a corpus of documents and from which a system can desume fuzzy information having a look at the distances between the elements.

Let us make an example, considering a simple proposition:

'A chicken is eatable'

Having at its disposal a corpus of documents speaking about chickens, our tool can desume the probability for a chicken to be eat as an inverse function of the distance between `chicken` noun and `eatable` attribute in the representation, and this distance is the less the more a chicken is seen as an object of the verb eat by the NLP algorithms. This reflects the human way of consideration that a fact is the more probable the more is seen happening. If a child see another child eating a coin, for instance, he is not absolutely sure that a coin is eatable, but he will be more and more sure of the possibility to eat a coin the more he observes people eating coins (quite unlikely, thanks Lord).

In this thesis work just action and affection types of attributes are extracted, but other techniques can be attempted to extract other types of attributes, even in a representation

<sup>6</sup>we can choose the  $n$  numbers of dimensions of the representation at will, according to some limits and considerations, see §3.2.3.2

of more than two variables (nouns and verbs), with the possibility to build huge repositories of fuzzy knowledge. I elaborate on these and other considerations in Chapter 6, Conclusions and Future Work, in §6.2.

For other information and considerations about this approach see Chapter 4, The Extraction Tool, in §4.4. In particular the discussion focuses on scalability, usability, extendability, validity of this type of representation.

Also, as far as we know, approaches like this were never attempted, anyway, information on approaches that appear to be in some way similar to this were collected in §2.2.

## Chapter 2

# State of the Art

In §2.1 I present the Ontology Learning from Text actual state of the art. A brief look to the paragraph could give you more the impression of a *history* of the art, because the first solutions presented in §2.1.2.1 are in fact the first attempts to extract structured data from dictionaries in the first '80s. The fact is that, up to now, these first approaches are still the best solutions in their specific field of application, and there is no reason to ignore them, being strong and valid as much as the most modern solutions, even if the field of application of these latter is much more wide, going, in fact, from large corpora of documents to the entire Web. Also, as presented in §1.6, a part of my work goes beyond the objectives of Ontology Learning from Text trying to extract a different type of representation of knowledge from corpora of documents. The research on the more similar attempted approaches is presented in §2.2.

### 2.1 Ontology Learning from Text

As described in §1.4, Ontology Learning from Text, in its most general meaning for computer engineering, is that branch of Information Retrieval that tries to generate formal and structured data of semantic information using as source a corpus of unstructured text dealing with a specific domain of knowledge.

The extraction can be assisted by a human agent, in this case we are speaking of *assisted* or *semi-automatic learning*, or it can use, as further source, any sort of structured data, in this latter case we are talking about *oracle guided learning*. If the clear objective of the algorithm is the expansion of a pre-constructed ontology, it is preferred to talk, instead of oracle-guided learning, of *bootstrapping*. If the algorithm makes no use of structured sources or human help, it is considered an *automatic learner*.

We are going to see that we are very far from building valid algorithms of automatic learning, and the most number of solutions strongly relies on the support of an oracle or a human that is supposed to be an expert of the domain of knowledge which the document corpus is related to.

In §2.1.1 I present the different sub-problems which the Ontology Learning task is divided into, then I discuss in detail about the two most important of them for the actual

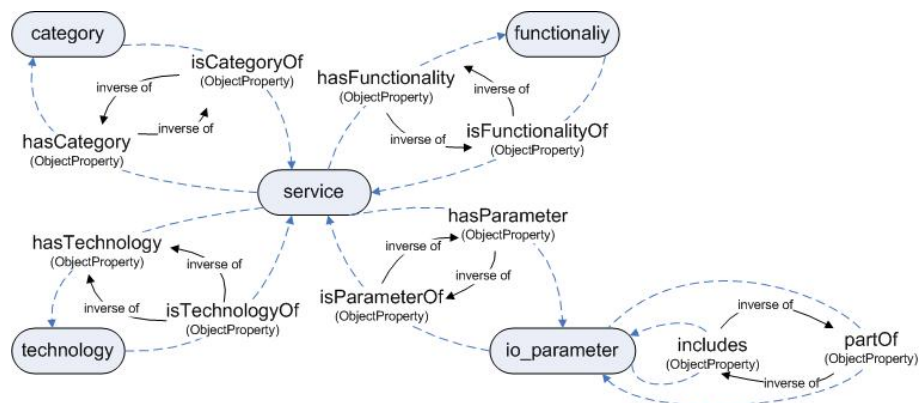


Figure 2.1: Ontology example from a business domain

state of the art: Concept Hierarchy Extraction (§2.1.2) and Concept Relations Extraction (§2.1.3). The most applied techniques in this field for the evaluation of the results are explained in §2.1.4. I suggest to refer to this latter paragraph in order to understand the value and meaning of the precision measures for the different algorithms presented in §2.1.2 and §2.1.3.

### 2.1.1 Ontology Learning from Text, Step by Step

As already said, ontologies are formal and structured representations of concepts and relations between these concepts. What we would attend from a satisfactory Ontology Learning algorithm is a sort of mechanical parsing of free text and a generation of structured data representable, for instance, like the one in Figure 2.1.

This, in fact, is correct, but there is a different type of information that can be seen more and more frequently going arm in arm with this representation, that is the definition of axioms. An axiom is in general a proposition that can be considered true in order of a evidence, and even if we cannot epistemologically admit its absolute and definitive truth, the conventional assumption of truth of axioms is needed for the creation of a basis of knowledge in any specific domain.

Let us just make an example: in a certain domain we see that a dog is white, a mouse is white and a cat is white. Would it be a right choice to assume as generally true the axiom: “all the animals are white”? Well, whether this choice would be right or not depends on the domain we are dealing with, you (just as me), having seen so many animals of so different colors, would probably think that is just the case to wait a little more and get some further observations before getting into this strong conclusion, but let us just imagine, for instance, that our domain of knowledge is a zoo of albino animals (quite improbable, in fact). Our axiom seems to go quite right in the case.

What is important in this example is that the axiom “all the animals are white” is a strong assumption that can be inducted (rightly or wrongly) from what we see in the microworld represented by our ontology. So, if we really expect from an ontology to represent a strong structured basis of knowledge for a specific domain, it seems at least convenient to make it carry also information about inferred axioms, and in fact the most modern language of definition of ontologies, like OWL [W3C, 2004], support the definition and storage of them. So, even if axioms are in fact something epistemologically<sup>1</sup> different from ontologies, their acquisition has historically been included as an objective of Ontology Learning.

The stairway going from the rugged ground of the unstructured text to the heaven of axioms is composed by seven steps, that are considered the seven main objectives of the Ontology Learning research field. They are:

1. Acquisition of **terms**
2. Identification of **synonyms**
3. Identification of **concepts**
4. Identification of **hierarchies of concepts**
5. Identification of **relations** between concepts
6. Identification of **hierarchies of relations**
7. Inference of **axioms**

### Terms

The terms are, simply, the words, the ordinate sequences of phonemes that, as we saw in §1.2, *stand for the objects*, as many little placeholders, in our logical propositions.

The problem of acquiring terms from free text that have relevant sense for the text’s domain of knowledge is very important, and it is a matter faced by the totality of the Ontology Learning algorithms. I present it as a intermediate passage to further objectives and, because of this, this problem has no standalone section in this chapter. There is to say, however, that presenting, for instance, an algorithm that tries to identify hierarchies of concepts, I will surely have to start the presentation speaking about how these concepts (terms, in fact, as it is explained in the “Concepts” subparagraph) are identified and extracted.

### Synonyms

Synonyms are terms that have interchangeable role in the construction of the logical proposition. So, replacing a term with a synonym, the sense of the proposition does not change, or at least not in our domain of knowledge. Some algorithms have between

---

<sup>1</sup>This adjective is quite compulsory, because ontologies are formally considered, axiomatic theories of the first order. So, ontological propositions are axioms too. The only difference, in fact, between ontological propositions and inferred axioms is that the ontological relations are *proper axioms* of the theory, while what we call axioms in this chapter are inferred one.

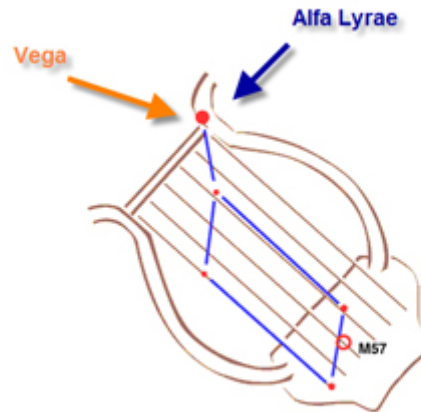


Figure 2.2: An example of synonyms in a specific domain of knowledge: the same star in the constellation of Lyra is indifferently called Vega or Alpha Lyrae in astronomy, by virtue of two different nomenclatures: an earlier Arabian denomination and a more modern European classification

their functionalities the identification of potential synonyms, but researches soon understood that it is quite rare to find synonyms inside a quite specific domain of knowledge, because a single domain, in its evolution, growing in its specificity, tends to weed synonyms out, to be better understood by the people involved. So the problem of synonyms is, in most of the different approaches, simply ignored. In Figure 2.2 an exception to what I have just asserted can be seen.

### Concepts

A concept represents the *sense* of a term, as a term acquire it in the context of a proposition (see §1.2). In relation to our work, concepts must be distinguished from terms because of the presence of synonyms (as already presented), and the presence of *polysemic* terms, that are terms with more than one sense. As already said, the more specific a domain of knowledge is, the more improbable it is to find synonyms in it, and polysemic terms are still more rare, always for the same natural process in a domain of ostracism of ambiguities.

So, at the actual state of the art, concepts and terms are considered the same, and synonyms and polysemic terms are a little rumor in the results that researches can tolerate. This will be more understandable later, having a look at the precision of the algorithms of Ontology Learning: rumors and errors are frequent and synonymity and polysemy in this scenario are the last of researchers problems. I will also use from here on indiffer-

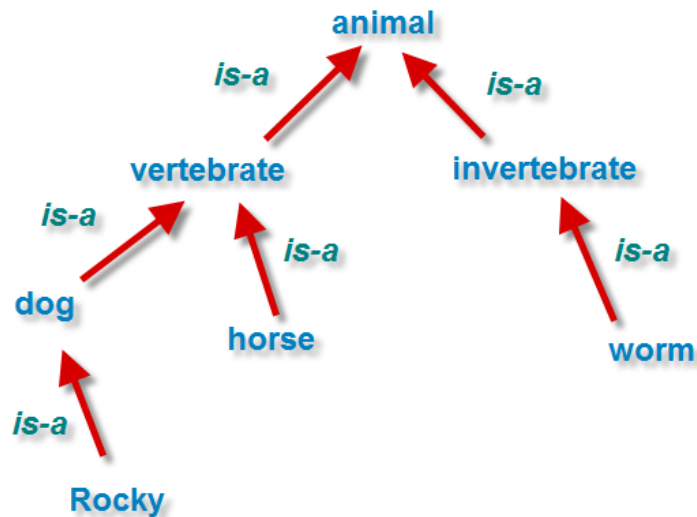


Figure 2.3: Example of a hierarchy of concepts with *is-a* relationships, already seen in Chapter 1

ently “term” and “concept” in my discussion, giving in fact few or no importance to their distinction.

### Hierarchies of Concepts

A hierarchy is a particular specification of an ontology where all the relations between concepts can be just of two type: the *is-a* relationship and the *is-a-part-of* one. What comes out generating an ontology with these bindings is a particular type of taxonomy, characterized by a tree where the concepts are the nodes of the data structure, and every node can have as many descendants as it likes.

Hierarchies are an important type of ontology, because they represent in fact an ordered categorization of concepts and give us the possibility to extend the properties of a single concept to all its descendants, considerably amplifying our knowledge of the domain. A structure like this is considered the backbone of any type of human knowledge, since Aristotle’s term logic (see [Aristotle, 340], or even [Kant, 1781]).

A child of a node in a tree composed by *is-a* relationships is defined a *hyponym* of that node. Viceversa, the father of a node in the tree is a *hypernym* of that node. An example of hierarchy is shown in Figure 2.3.

### Relations

Relations are the link between the concepts, they are what creates the unitary and significant proposition in our logic. The acquisition of relations between terms is the other important objective of Ontology learning and different important approaches are exposed in §2.1.3.

### **Hierarchies of Relations**

The creation of hierarchies of relations is the possibility to specify that a relation between terms strongly implies also the presence of all its ancestors, for instance the relation *is-child-of* can be defined as a sub-relation of *is-descendant-of*. The possibility to specify hierarchies of relations is part of any specification of ontologies, but the problem of their acquisition is mainly considered, at the actual state of the art, as a future objective: when algorithms that define relations will be more precise, then the time will come to understand how to generate relations hierarchies. For this reasons, there is no state of the art about this matter.

### **Axioms**

As for hierarchies of relations, the inference of axioms from ontologies extracted from free text is a future objective of Ontology Learning.

### **Summary and Conclusions**

As we saw, the most important objectives of Ontology Learning at the actual state of the art are the acquisition of terms, the acquisition of hierarchies of concepts and the acquisition of relations between concepts. Synonyms and distinction between terms and concepts are for the most ignored. Hierarchies of relations and axioms are future objectives. Hierarchies and relations have their singular dedicated paragraph respectively in §2.1.2 and §2.1.3. The matter of acquisition of terms will be faced in the same paragraphs as an intermediate step on the way to the other two objectives.

## **2.1.2 Concept Hierarchies Extraction**

I review now different approaches to learning concept hierarchies. Approaches presented are, in the order:

- Machine Readable Dictionaries (early '80s)
- Hearst Patterns (1992)
- Distributional Similarity (1998-today)
- Formal Concept Analysis (2000-today)
- Learning By Googling (2004-today)
- Bootstrapping (1990-today)



### 2.1.2.1 Machine Readable Dictionaries

Early work on extracting taxonomies from machine readable dictionaries (MRDs) goes back to the early '80s ([Amsler, 1981], [Calzolari, 1984]). The idea is to exploit the regularity of dictionary entries to first of all find a suitable hypernym for the defined word. In many cases, the head of the first noun appearing in the dictionary definition is in fact a hypernym. We can consider, for example, the following definitions taken from Dolan and others [Dolan W., 1993]:

- spring:** "the season between winter and summer and in which leaves and flowers appear"
- nectar:** "the sweet liquid collected by bees from flowers"
- aster:** "a garden flower with a bright yellow center"

However, there are also some exceptions to the above rule. On the one hand, the hypernym can be preceded by an expression such as 'a kind of', 'a sort of' or 'a type of'. Here follow some examples taken from Alshawi [Alshawi, 1987]:

- hornbeam:** "a type of tree with a hard wood, sometimes used in hedges"
- roller coaster:** "a kind of small railway with sharp slopes and curves"

The above problem is easily solved by keeping an exception list with words such as 'kind', 'sort', 'type' and taking the head of the noun following the proposition "of" as the searched term.

The above examples suggest that one can extract structures from dictionaries containing a wealth of semantic relations linking the different words together. Of particular interest is the work described in [Amsler, 1981], [Calzolari, 1984], and [Dolan W., 1993]. These researchers have attempted to build a large network containing taxonomic links between words denoting hypernymy or hyponymy relations. As argued by Dolan [Dolan W., 1993], such network has important applications in natural language understanding. An important advantage of using dictionary definitions for building a taxonomy is that dictionaries separate different senses of words. So, taxonomic relations are learned between concepts, rather than between terms.

In general, approaches deriving taxonomic relations from MRDs are quite accurate. Dolan and others, for example, mention that 87% of the hypernym relations they extract are correct. Calzolari cites a precision of more than 90%, while Alshawi mentions a precision of 77%. These methods are quite accurate due to the fact that dictionary entries show a regular structure. Dictionary definitions in fact contain quite explicit knowledge compared to unstructured text and so provide an interesting basis for Ontology Learning. I personally see two main drawbacks in using a dictionary-based approach in Ontology Learning: the first is related to the fact that the acquired knowledge heavily depends on the "writing style", the stylistic (however regular) behavior of the authors in writing the entries. Second, in Ontology Learning we are mostly interested in acquiring domain-specific knowledge. Dictionaries are generally domain independent resources.

It is thus clear that we cannot go too far limiting our sources of knowledge to dictionaries and similia.

### 2.1.2.2 The Importance of Being Hearst... Pattern

In 1992 Marti A. Hearst wrote a seminal work titled "Automatic Acquisition of Hyponyms from a Large Text Corpora" [Hearst, 1992]. The simple ideas contained in this writing have heavily influenced most of the following approaches in Ontology Learning which applied different techniques in the next years to look in a corpus for, from there on, the so-called Hearst Patterns.

Hearst Patterns are a pre-defined collection of patterns indicating hyponymy relations in a text for a specific language. An example of such a pattern used by Hearst is the following:

such  $NP_0$  as  $NP_1, \dots, NP_{i-1}$  (or | and) other  $NP_i$

where  $NP$  stands for an English noun phrase. If such a pattern is matched in a text, according to Hearst we could derive that for all  $n$  from 1 to  $i$   $NP_n$  is an hyponym of  $NP_0$ .

The patterns suggested by Hearst are the following:

- $NP$  such as  $NP,^*$  (and | or)  $NP$
- such  $NP$  as  $NP,^*$  (and  $j$  or)  $NP$
- $NP, NP^*$ , or other  $NP$
- $NP, NP^*$ , and other  $NP$
- $NP$  including  $NP,^* NP$  (and | or)  $NP$
- $NP$  especially  $NP,^*$  (and | or)  $NP$

Here  $*$  stands for a 0 to  $n$  presence of other  $NPs$ .

According to Hearst, the patterns should satisfy the following requirements:

1. They should occur frequently and in many text genres.
2. They should accurately indicate the relation of interest.
3. They should be recognizable with little or no pre-encoded knowledge.

Hearst mentions that an important issue is how to treat nominal modification, in particular adjectives pre-nominally modifying a noun. She does not give a definite answer to this problem, but mentions that the choice here certainly depends on the application. Furthermore, Hearst also suggests a procedure in order to acquire such patterns:

1. Decide on a lexical relation of interest, such as hyponymy or hypernymy.
2. Gather a list of terms for which this relation is known to hold, for example: *is-a(car, vehicle)*. This list can be found automatically using the patterns already learned or from an existing lexicon or knowledge base.

3. Find expressions in the corpus where these terms occur syntactically near one another.
4. Find the commonalities and generalize the expressions in 3. to yield patterns that indicate the relation of interest.
5. Once a new pattern has been identified, gather more instances of the target relation.

As mentioned by Hearst, the value of such lexico-syntactic patterns is that they can be identified easily and are quite accurate. Hearst, for example, showed that, out of 106 relations extracted with her method from New York Times texts where the hyponym and hypernym were in WordNet, 61 were correct with respect to WordNet<sup>2</sup>.

The accuracy of Hearst patterns is, in this case, 61/106, i.e. 57.55%. The drawback of the patterns is however that they appear rarely and most of the words related through an is-a relation do not appear in Hearst-style patterns. Thus, one needs to process large corpora to find enough of these patterns. For this reason, recently, several researchers have attempted to match these patterns on the Web as a big corpus using some query engine ([Markert K., 2003], [Pasca, 2004], [Etzioni, 2004]). A further drawback of such approach is that the patterns are typically specified in the form of regular expressions and this imposes limits on their accuracy. Given a sentence as:

“The main historic area of Soleminis is a charming village”

most of the approaches based on matching lexico-syntactic patterns would derive: Soleminis *is-a* village, which is definitely not correct. In fact, language’s variety and its transformational power is difficult to be captured merely relying on regular and static patterns.

### 2.1.2.3 Distributional Similarity

Distributional Similarity is the Ontology Learning research branch which the ontological approach presented in this work entirely belongs to.

As already mentioned in §1.5, the *distributional hypothesis* says that words are similar to the extent that they share similar context [Harris, 1968]. In fact, empirical investigations corroborate the validity of the above hypothesis. Miller and Charles [Miller G., 1991], for example, found in several experiments that humans determine the semantic similarity of words on the basis of the similarity of the contexts they are used in. Grefenstette [Grefenstette, 1994] further showed that similarity in vector space correlates well with semantic relatedness of words.

Starting from this early researches, it has been hypothesized that the occurrence of some word implies the occurrence of some other word in the same sentence, paragraph or document hints at a potential directed relation between both words. Directed means

---

<sup>2</sup>WordNet is a huge English hierarchy of concepts built and constantly update in Princeton University since 1985, it is used as a reference from many algorithm of Ontology Learning because researches can be sure of the correctness of its relations, being created manually. For more information see, for example, [Fellbaum, 1998].

for example a sub-topic, *is-a* or *is-a-part-of* relation. This notion is related to the one of *collocation*. We say that two words form a collocation if they occur together in a paragraph, sentence, document or next to each other more often than predicted by chance.

Though directed co-occurrences seem in fact to indicate some directed relation between the involved words, it is unclear which specific relation actually holds between these. Some research has suggested that depending on the context we consider: considering sentences, paragraphs or even whole documents, we tend to get different types of relations.

Starting from this background of considerations, different approaches tried to statistically analyze the co-occurrences of the terms over a specific type of context, to then generate a sort of classification of the terms based on the similarity of their distribution over the contexts of the chosen type. After this they build a hierarchy of concepts trying somehow to infer the possible *is-a* relations starting from the classification obtained with the support of some other information (e.g., Hearst Patterns).

Every algorithm can be distinguished from others for:

1. What is chosen as **context**
2. The **statistical approach**
3. The **distance metric** used to compute the similarity between the terms
4. The so-called method of **Hierarchical Clustering**, used to generate a first sort of taxonomy of the terms, classifying them by their reciprocal proximity
5. The algorithm (if present) to generate the final hierarchy starting from the representation in 4.

I am going now to describe in detail these 5 different aspects that characterize the distributional similarity's algorithms, then I get on to present some different distributional similarity's approaches tried in the last two decades. The main steps common to every distributional similarity's approach are summarized in Figure 2.4.

### The 5 Main Aspects of a Distributional Similarity Algorithm

#### Context

The Context is, as said, the place where collocations of terms are evaluated. Different context can take to very different results. These are the contexts seen to be used in the different algorithms:

- Window of  $n$  words before and/or after the term
- Sentence
- Paragraph
- Page

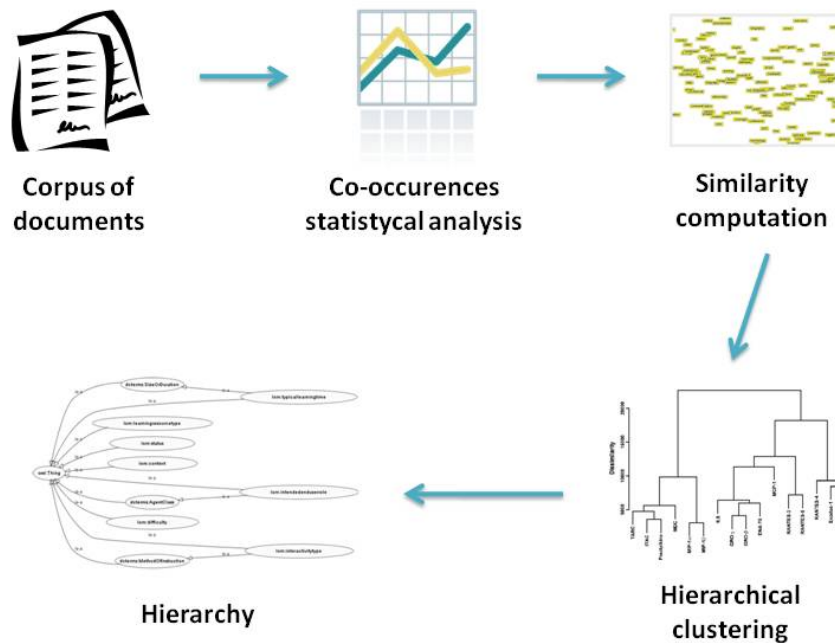


Figure 2.4: Main steps of a Distributional Similarity algorithm for the generation of a hierarchy of terms from a corpus of unstructured documents

- Document
- Verb which the term is subject or object of (or, rarely, other complements)

### Statistical Approach

The co-occurrences of the terms must be statistically analyzed in order to compute the similarity of distribution of the different terms over the different contexts (for example, a distribution of terms in different documents). The different techniques used are:

- **Simple co-occurrences analysis:** a vector is generated for every term containing its probability to occur in every context; these terms can be so considered as projected in a  $j$ -dimensional Euclidean space, with, as coordinates, their probability to occur in the  $j$ th context. The similarity between two terms is often computed as the

inverse of the distance between the two terms in the  $j$ -dimensional space, but, in other cases, other measures can be used and weights can be applied (see **Distance Metric**).

- **Reduction of Complexity:** As can be easily understood, the main matrix of terms occurrence probabilities, for large corpora of documents, can be very huge (having a dimension of terms  $\times$  contexts). Actual computational limits suggest to take the terms in a lower dimensional space, and this can be done applying mathematical techniques that try to identify the most important axes of distribution of the terms and translate them in a new space where axes that carry lower or no information can be removed. A mathematical approach usable in order to do so is the Singular Value Decomposition (SVD): for a detailed description of its operation, you can have a look at [Golub and Van Loan, 1996]. Our own statistical model of Correspondence Analysis uses instead the different approach of eigendecomposition, you can see §3.2.3.2.

### Distance Metric

As said, what we obtain from co-occurrences analysis is a vector  $1 \times j$  for every term containing the probability of the term to occur in the different contexts. The distance between terms can be computed using different metrics, and similarity is obviously computed as the inverse of this distance:

- **Cosine:** the cosine between the two vectors is computed as the distance between the terms:

$$\cos(x, y) = \frac{\sum_{x \in X, y \in Y} xy}{\sqrt{\sum_{x \in X} x^2} \sqrt{\sum_{y \in Y} y^2}}$$

- **Euclidean Distance:** this is the distance between the terms if considered as projected in a  $j$ -dimensional Euclidean space, it can be computed as:

$$D(x, y) = |x - y| = \sqrt{\sum_{i=0}^j (x_i - y_i)^2}$$

- **Relative Entropy:** the entropy of a random variable  $X$ , distributed as the probability function  $p(X)$ , can be defined as its *average uncertainty*, it can be computed as follows:

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$

Entropy can be thought as the average length of the message needed to transmit the outcome of that variable. *Relative Entropy* or *Kullback-Leibler divergence* is a measure of how different two probability distributions (in our case, the probability distribution of a term in respect to the totality of the contexts) are:

$$D(p||q) = \sum_{x \in X} p(x) \log_2 \frac{p(x)}{q(x)}$$

- **$\chi^2$  distance:** the  $\chi^2$  distance is similar to the Euclidean distance, but in fact it applies a weight to the projection of the distance along a specific axis (representing a context) that is in inverse proportion to the population of that context, giving larger importance to contexts (axis) with few terms in respect to context with many terms. This is the distance we use in our algorithms and its advantages and disadvantages in respect to other metrics, and particularly in respect to the Euclidean distance, are discussed in §3.5.

$$\chi^2(x, y) = \sqrt{\sum_{i=0}^j \frac{(x_i - y_i)^2}{P_j}}$$

where  $P_j$  is the sum of the probabilities of all the terms to be seen in the  $j$ th context.

### Hierarchical Clustering

Starting from terms' similarity values, Distributional Similarity's approaches organize the terms in a first taxonomy, in a way that the more two terms are similar, the shorter is the path to be covered from the first to the second (Figure 2.5). The structure can also be seen as a set of clusters growing bigger, collecting all the terms (Figure 2.6). The algorithm applied to create this structure is *Hierarchical Clustering*. As described by Maedche and Staab [Maedche A., 2003], the tree of hierarchical clusters can be produced either bottom-up, by starting with individual objects and grouping the most similar ones, or top-down, whereby one starts with all the objects and divides them into groups.

The **bottom-up** algorithm starts with a separate cluster for each object. In each step, the two most similar clusters are found, and merged into a new cluster. The algorithm terminates when one large cluster containing all objects has been formed.

The **top-down** algorithm starts out with one cluster that contains all objects. The algorithm then selects the least coherent cluster in each iteration and splits it. Clusters with similar objects are more coherent than clusters with dissimilar objects.

An important aspect that influences the effectiveness of the results of both these two algorithms is the selection of a good similarity measure and a good context of distribution. Probably, in order to make a choice, some inferences can be made having a look at

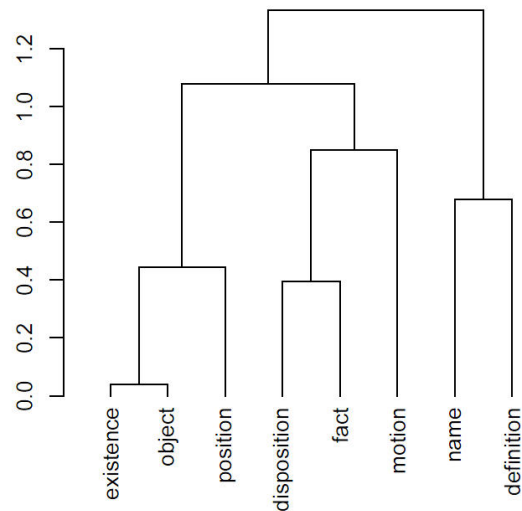


Figure 2.5: Example of Hierarchical Clustering of terms extracted from a Plato’s dialogue, already seen in Chapter 1

the structure of the corpus, but up to now there is no empirical rule to challenge the problem, and probably some experiment must be made with different contexts and measures before getting the best results.

### Generation of the Hierarchy

Different algorithms present very different approaches to the problem of getting the final hierarchy from the hierarchical clustering representation. Most of them ask for help to oracles, to algorithms looking for heurst patterns in the same corpus of documents or over the Web, or to a human that interacts with the program through any sort of interface. I describe now different distributional similarity approaches carried on in the last two decades, remanding to every paragraph for the detailed description of how every single researcher faced the problem.

### Different Distributional Similarity Approaches in the Last two Decades

#### ASIUM, 1998

ASIUM (see [Faure D., 1998]) implements a bottom-up clustering strategy for the purpose of learning a concept hierarchy. The first step of the algorithm is to build the so-called basic clusters, consisting of nouns collected in relation to the verb of the tense in



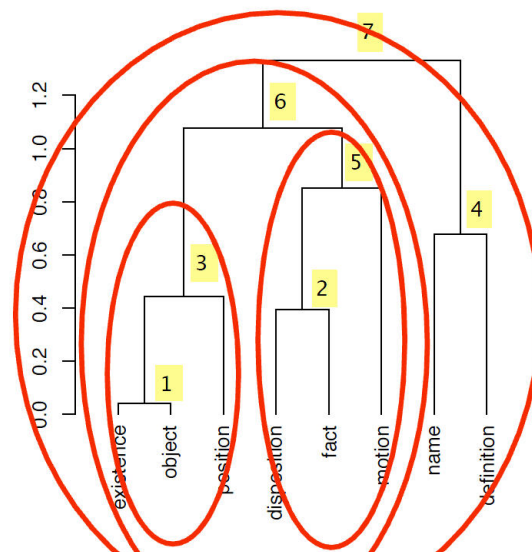


Figure 2.6: The taxonomy seen as a set of larger and larger clusters

which they appear and their syntactic function in the same tense. In particular, the authors consider the verb's object, subject as well as propositional complements or adjuncts.

For example, given the following verb structures found in a corpus:

```
<to travel> <subject: father> <by: car>
<to travel> <subject: neighbor> <by: train>
<to drive> <subject: friend> <object: car>
<to drive> <subject: colleague> <object: motorbike>
<to drive> <subject: friend> <object: motorbike>
```

ASIUM would generate the following two verb frames:

```
<to travel> <subject: {father(1), neighbour(1)}>
<by: {car(1), train(1)}>
<to drive> <subject: {friend(2), colleague(1)}>
<object: {car(1), motorbike(2)}>
```

and would create four basic classes:

```
C1={father(1), neighbour(1)}
C2={car(1), train(1)}
```

```
C3={friend(2), colleague(1)}
C4={car(1), motorbike(2)}
```

After this first step, the distance between the different clusters is computed. Clusters with a distance under a certain threshold are merged. The distance measure between clusters  $C_1$  and  $C_2$  is defined as follows:

$$dist(C_1, C_2) = 1 - \frac{|C_1 \cap C_2| \cdot \left( \frac{\sum_{c \in C_1 \cap C_2} w_{C_1}(c)}{|C_1|} + \frac{\sum_{c \in C_1 \cap C_2} w_{C_2}(c)}{|C_2|} \right)}{\sum_{c \in C_1} w_{C_1}(c) + \sum_{c \in C_2} w_{C_2}(c)}$$

where  $w_C(c)$  is the weight of each element  $c$  in the cluster  $C$ .

For example, the distance between  $C_2$  and  $C_4$  is computed as follows:

$$dist(C_2, C_4) = 1 - \frac{1 * \left( \frac{1}{2} + \frac{1}{2} \right)}{2 + 3} = 1 - \frac{1}{5} = \frac{4}{5}$$

According to ASIUM's algorithm, new clusters are only merged with clusters from lower levels.

The generation of the hierarchy is done with the help of a human agent. After each level, a user is asked to validate and label the clusters. Thus, clusters are only processed further if they have been validated by the user, decreasing the amount of noise produced by the algorithm. Furthermore, the user can not only accept the cluster as a whole, but also split it into subclusters or remove some elements. The generalized subcategorization frames can also be adjusted by the user in a similar way.

Faure and Nedellec present an evaluation of ASIUM by showing the number of correctly generated clusters in dependence of the size of the corpus used: the best result is a cluster accuracy of 99.53% using 90% of the corpus with an a posteriori evaluation of the clusters by a human judge. Obviously, the request for a human interaction is a large disadvantage in terms of time requested for any result, making the management of large corpora, in fact, almost impossible.

### Caraballo, 1999

Caraballo [Caraballo, 1999] presents a bottom-up clustering approach to build a hierarchy of nouns. For this purpose, she looks for terms with the paragraph as context, in a corpus of articles from the Wall Street Journal, using the parser described in [Caraballo and Charniak, 1998]. As similarity measure she uses the cosine measure. Caraballo first clusters the nouns in a bottom-up manner, thus yielding to the first taxonomy of nouns.

To generate the hierarchy, she tries to label the inner nodes of the tree, extracting appropriate hypernyms from the corpus using the Hearst pattern "NP, NP,... and other

NP". The hypernyms are added to every leaf node in the tree in form of a vector. For each internal node of the tree, the vectors of its children are aggregated. Finally, for each node the hypernyms are ranked according to frequency, and the best three hypernyms are chosen as cluster label if the hypernym subsumes at least two of the elements in the cluster. After the clusters have been labeled, the tree is then compressed removing every inner node which remains unlabeled, as well as every node with the same hypernyms as its parent. The children are correspondingly raised along the hierarchy to the parent concept.

Caraballo evaluates her approach by randomly selecting 10 internal nodes dominating at least 20 nouns and, for each internal node, randomly selecting 20 nouns under that node which were presented together with the three hypernyms to three human judges for evaluation. Additionally, five "noise" nouns selected from elsewhere in the hierarchy were also presented to the judges to check that they were not just confirming the results by default. The pairs consisting of the best hypernym of a cluster and some hyponym were accepted by the majority of judges in 33% of the cases, 39% of the cases at least by one judge. Considering any of the three hypernyms for each cluster, 47.5% relations were accepted by the majority, while 60.5% were judged as valid by at least one judge.

### **Terascale Knowledge Acquisition, 2005**

In 2005 Ravichandran, Pantel and Hovy [Ravichandran D., 2005] have attempted to speed up traditional clustering techniques to group similar nouns on the basis of very large corpora. They notice that every similarity-based method needs at least to build a similarity matrix which takes  $O(n^2j)$  time, where  $n$  is the number of nouns and  $j$  is the number of different contexts considered. Thus, if one wishes to have an algorithm which is linear in the number of nouns, one has to avoid calculating the whole similarity matrix using traditional techniques. Ravichandran, Pantel and Hovy propose to use the so-called Locality Sensitive Hash (LSH) functions, which are randomized and probabilistic, thus optimizing the similarity computation by creating short signatures for each vector and comparing their fingerprints. The computation of the similarity matrix is thus reduced to  $O(nk)$ . In particular, they use a local sensitive hash function approximating the cosine similarity measure.

### **Oracle-Guided Agglomerative Clustering, 2006**

Phillipp Cimiano in [Cimiano, 2006], describes an algorithm of generation of hierarchies parsing a corpus of documents that uses, as a sort of prompter, a pre-defined ontology. What they obtain is not in fact an extension of the pre-existent ontology (as in §2.1.2.7) but a new and independent one, with the effort in its generation of the previous referential ontology.

He uses, as distance metric, the cosine measure. What is interesting is the algorithm of generation of the hierarchy, using as a reference a pre-constructed structured ontology. let us take two terms  $t_1$  and  $t_2$  with a certain measure of similarity  $sim(t_1, t_2)$ . For each pair  $(t_1, t_2)$ , the algorithm thus first consults the hypernym oracle to find out if  $t_1$  is a hypernym of  $t_2$  or the other way round, creating the appropriate subconcept relation. If this is not the case, it consults the oracle for common hypernyms of both terms, selecting the most frequent hypernym  $h$  and distinguishes three cases. In case none of the terms has already been classified, it creates a new concept labeled with  $h$  together with two

subconcepts labeled as  $t_1$  and  $t_2$ . In case one of the two terms, say  $t_1$ , has already been classified as  $is-a(t_1, t')$ , there are three more cases to distinguish. In the first case, if  $h$  and  $t'$  are identical, the algorithm simply puts a concept  $t_2$  under  $t'$  (figure 2.7, left). In the second case, if, according to the oracle,  $h$  is a hypernym of  $t'$ , it creates the structure in Figure 2.7, middle. In case it is not a hypernym, it creates the structure in Figure 2.7, right. The algorithm proceeds analogously in case  $t_2$  has already been classified. In case there are no common hypernyms,  $t_1$  and  $t_2$  are marked as clustered for further processing. This is done for all the similarity pairs, provided that one of the two terms has not been classified yet.

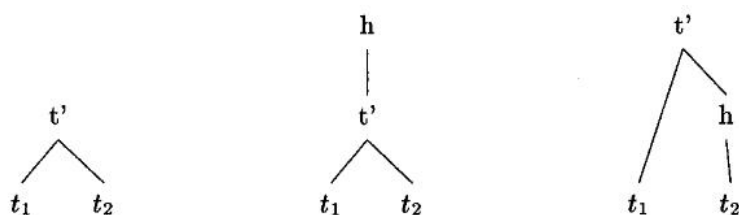


Figure 2.7: Different structures constructed by the oracle-guided agglomerative clustering algorithm

At the end, the algorithm tries to apply other techniques to infer any subsumption relationship from terms marked as unprocessed, as the research for hearst patterns, and other heuristics. If no subsumption is still found, the term is simply put under the root concept.

### Fionn Murtagh, 2007

This is the approach from which this thesis work starts, documented by the writings of Murtagh [Murtagh, 2005] and [Murtagh, 2007].

As already seen in §1.5, Fionn Murtagh proposes a particular statistical approach for the study of the distribution of terms that is the Correspondence Analysis (see Chapter 3), developed by a statistician of his same school, Jean-Paul Benzécri. He proposes to use as context the document, and to evaluate the distance between terms using the  $\chi^2$  distance.

He then uses a bottom-up approach to create the Hierarchical Clustering representation, and then he creates the hierarchy with an algorithm described in §4.1.4.5. In general, details about his algorithms, considerations and evaluation of performances can be found in Chapter 4 and 5.

### XTREEM, 2008

XTREEM (Xhtml TREE Mining, [Brunzel, 2008]) is a method to learn ontologies from Web documents. Using as a source the entire Web, Marko Brunzel pose between his objectives the learning of terms, the identification of synonyms and the identification of what he calls *sub-ordination relations* (our *is-a, is-a-part-of* relations), and *co-ordination relations*, that are ontological relations between hyponyms of the same term.

He uses as context what he calls the “text path”. What XTREEM does, parsing an xhtml Web page, is to divide it in many different areas according to some rules that try to identify sectors of the page with different communication role. These sectors are called *text path*. For example, the footage of a page is probably considered a different text path in respect to a text paragraph.

According to [Brunzel, 2008], what XTREEM does, up to 2008, is to extract from web Pages terms, synonyms and siblings (candidate hyponyms of the same term). Every other functionality is still under development. We can imagine that, having as a source of knowledge the entire Web, XTREEM points to something more than many single-domain knowledges, this, in fact, justifies the necessity to identify synonyms between the terms.

#### 2.1.2.4 Formal Concept Analysis (FCA)

Formal Concept Analysis, as described by Philip Cimiano [Cimiano, 2006], is an approach to extract concept hierarchies from text that heavily distinguish itself from distributional similarity for a different basis of assumptions and a large usage of algorithms of Natural Language Processing (NLP).

One of the main advantages of distributional similarity is that, in the most of the cases, its algorithms can be largely considered to be language-independent. The generation of the hierarchical clustering representation is in fact language-independent, heurst patterns are language-dependent but changing a pattern with its correspondent in another language is a trivial task. Algorithms using NLP approaches in distributional similarity are quite rare (The work presented in this thesis, however, belongs to this category).

FCA is an approach that requests deeply language-aware algorithms of analysis. Let us just examine now the sequence of passages of the algorithm, from the unstructured text to the hierarchy (see Figure 2.8), postponing other theoretical considerations in the single sections.

##### **Lemmatizer**

Nouns and verbs are extracted from the documents and lemmatized, that means, reported to their base form. This is important to put in the same set the occurrences, for instance, of “bicycle” and “bicycles”, or “ride” and “have been riding”. Both nouns and verbs extraction and lemmatizing are research fields of NLP still under work, their results are subject to a noise caused by the variety of the language and the way we play with it (as explained in §1.3). So, a noise will be always present, even if deaden year by year, as the algorithms are improved in their performances.

The influence of the NLP noise, however, loses a great part of its influence in the next passages, where the results are going to be statistically analyzed, making the rarest

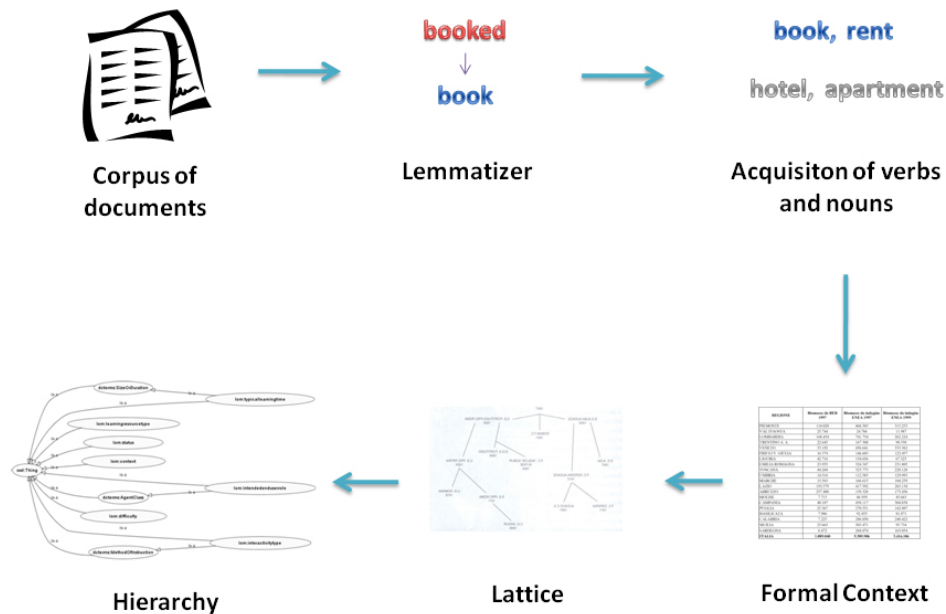


Figure 2.8: The FCA algorithm main passages

errors going to be ignored in the great number of the statistically more influential right occurrences.

### Acquisition of verbs and nouns

Propositions are acquired in the form of the triplet (*subject, verb, object*). The probability for a noun to appear as subject or object of a verb is reported in a table (figure 2.9).

Up to now, there are not so many differences with the distributional similarity techniques. Terms are denoted by a distribution of probabilities against a context, that is the combination of the the syntactical function of the term and the verb of the tense.

We are going to see that from now on FCA builds its hierarchies following different assumptions in respect to distributional similarity.

### Generation of the Formal Context

	$P(n book_{obj})$	$P(n rent_{obj})$	$P(n drive_{obj})$	$P(n ride_{obj})$	$P(n join_{obj})$
hotel	$\frac{10}{24}=0.42$				
apartment	$\frac{6}{24}=0.25$	$\frac{5}{11}=0.45$			
car	$\frac{3}{24}=0.13$	$\frac{4}{11}=0.36$	$\frac{5}{7}=0.71$		
bike	$\frac{2}{24}=0.08$	$\frac{3}{11}=0.27$	$\frac{2}{7}=0.29$	$\frac{2}{2}=1$	
excursion	$\frac{1}{24}=0.04$				$\frac{3}{5}=0.6$
trip	$\frac{2}{24}=0.08$				$\frac{1}{2}=0.4$

Figure 2.9: Table of term's probabilities of occurrences, taken from an example of extraction from a corpus about tourism, in this case terms are considered just when they appear as an object of a verb

	bookable	rentable	driveable	rideable	joinable
hotel	x				
apartment	x	x			
car	x	x	x		
bike	x	x	x	x	
excursion	x				x
trip	x				x

Figure 2.10: Example of a Formal Context

Other heuristics are applied to the table in order to smooth unmanageable results, but, in fact, what is obtained at the end is a table of 1s and 0s, *tertium non datur*, representing whether there is or not a relationship between the term and the verb (figure 2.10).

### The Lattice

The dependencies found are used to generate a tree of terms basing on what the concepts are able to do (considering their occurrences as subjects) or what can be done with them (considering their occurrences as objects). In Figure 2.11 you can see an example of the second case, the more can be done with an object, the more specific it is, being put in a deeper position in the hierarchical tree.

The theoretical basis of this approach remands again to Aristotle's Categories [Aristotle, 340]: a category is defined as a conceptual container of attributes common to

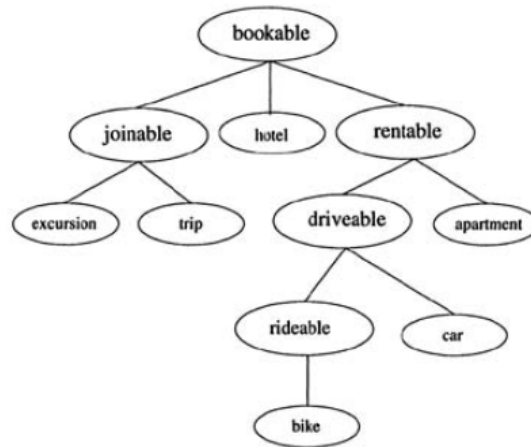


Figure 2.11: Example of a Lattice

all the beings belonging to that category. The more attributes a category has, the higher is its specificity, and the less beings belong to it. Between the attributes of a category are included what a being can do (the *action*) and what can be done with it (the *affection*), exactly the two types of information acquired about a concept in a formal context (we have seen that the last part our work refers to the same two categories in §1.6). So, FCA is, at the end, a categorization of concepts according to logical attributes inferred from the text.

### The Hierarchy

A concept hierarchy can be generated from the lattice substituting the verbs with categories, this can be done using already seen techniques:

- Hearst patterns
- User intervention
- Help of an Oracle

Nodes, also, can collapse to their father instead of being labeled, if no valid category is identified.

### Evaluation tests



Precision of these algorithms in putting the terms in the right sibling-relationships (so, as children of the same node) is attested by Cimiano to be between 30% and 40%.

### 2.1.2.5 Learning from Heterogeneous Sources of Evidence

An interesting idea comes from Cimiano, Pivk and Staab [Cimiano P., 2003]. They apply very different techniques on the same corpus of documents in order to combine the different results. Total precision is attested to improve of 10-15% the best precision between the single algorithms. In brief, they are:

1. Hearst pattern search
2. Search for the same hearst patterns on the Web (the number of hits returned is in fact an important clue to confirm or deny the reliability of a subsumption relationship, see §2.1.2.6)
3. Oracle: WordNet
4. Expansion of past-built hierarchies (or *bootstrapping*, see §2.1.2.7)
5. Distributional Similarity analysis on the corpus, using the document as context
6. Distributional Similarity analysis on a corpus of downloaded web pages that reasonably belong to the same domain of the corpus

### 2.1.2.6 Learning by Googling

The Learning by Googling approach is based on the idea that collective knowledge (that is: the Web) can be gathered as a first step and then as a second step filtered, either automatically or manually by a human expert of a certain domain. In fact, if the collective knowledge is presented to a human expert, he can then effectively customize this collective knowledge with regard to the specific context of interest. In this model, the purpose of general knowledge is to compensate the potential lack of knowledge of an individual with respect to a certain topic, while the role of the individual is to filter the collective knowledge with regard to the specific domain.

With respect to the task of generating hierarchies, the Learning by Googling paradigm tries to collect evidence from the Web for the different concepts a given instance could belong to. The evidence collected can then be used either to find the concept with the maximal evidence automatically, or be presented to a human expert who selects the most appropriate concept.

This abstract model is instantiated by PANKOW (Pattern-based Annotation through Knowledge on the Web, [Cimiano P., 2004b]) as well as by its successor C-PANKOW ([Cimiano P., 2005]). The core of PANKOW is a pattern generation mechanism which creates pattern strings out of a certain pattern schema conveying a specific semantic relation.

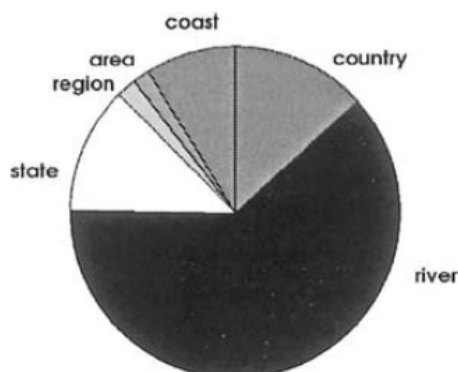


Figure 2.12: Results gathered looking for a hypernym of 'Niger'

## PANKOW

PANKOW counts the occurrences of hearst pattern strings on the Web using the Google API (see [Google, 2009]). In the automatic approach the term used to generate the patterns is then annotated to be an hyponym of certain concept, according to the the largest number of hits of the patterns.

let us see an example: the string 'Niger' appears and we have no idea about how to annotate it. Figure 2.12 shows the results of different Google searches of relative hearst patterns.

Given these Figures, the automatic algorithm would annotate 'Niger' as a river as it is its most prominent meaning on the Web.

If we wanted to help the algorithm in its decisions, instead, PANKOW would provide a user interface that would present itself with the top 5 suggestions about how to annotate the instance Niger, i.e. as a river, as a country, etc. The advantage of such an approach combining collective and individual knowledge seems thus obvious: even if the individual has never heard about the term in question, together with the collective knowledge and the local context in which the instance appears, he might get a fairly accurate idea of the concept it belongs to.

Cimiano and Staab, in [Cimiano P., 2004b], illustrates probably for the first time the fact that formal ontological annotations can be inferred from statistical distribution of certain syntactic structures over the Web. In this line, they present their vision of a 'Self-Annotating Web' in which globally available syntactic data are considered a powerful source able to suggest metadata creation.

## C-PANKOW

C-PANKOW ([Cimiano P., 2005]) has been developed in 2005 to address some of the shortcomings of PANKOW. First, due to the restrictions of the pattern generation process, a lot of instances of the patterns were not found. In particular there were problems generating the correct plural forms of concepts as well as matching more complex linguistic structures such as noun phrases including determiners, noun modifiers, etc.

C-PANKOW overcomes these problems downloading the pages, analyzing them linguistically and matching the patterns instead of merely generating plain strings and counting their Google hits. The results of the pattern-matching are also linguistically lemmatized, thus solving the problem with the generation of plural forms: it is in fact more easy to singularize a noun than to generate its plural form. Also, downloading the page, it is solved the problem of the high network traffic generated by a great number of concurrent Google queries and relative HTTP connections, often needed by the algorithms of PANKOW.

### 2.1.2.7 Bootstrapping

There are other approaches which do not attempt to generate ontologies from scratch, but pose as their unique objective to extend the existing ones.

The name *bootstrapping* comes from the saying “to pull yourself up by your bootstraps”, that means an impossible task. In order to better understand the concept you can have a look also at the painting “Waterfall”, of the dutch graphic artist M. C. Escher (figure 2.13).

Both ideas refer to the physical impossibility of a perpetual cycle where something’s reactions fed in his energy by the resultant energy of the reaction itself.

In our case this denotes the fact that bootstrapping algorithms need a pre-built ontology to operate, that has to be generated by another algorithm, etc... but the absurd is easily solved manually building at least the first ontology.

Many methods were developed and they are all based on the methods of distributional similarity (as described in §2.1.2.3), you can see for example [Hearst M., 1993], [Schutze, 1993] and [Alfonseca E., 2002]. Their common approach is to evaluate from a corpus of documents the similarity between a missing word in the ontology to expand and other words already present. Then the algorithms put the new concept as a son of the the concept in the ontology that has the more similar children, according to the similarity information gathered from the corpus of documents.

A different approach is presented by Maedche and Staab [Maedche A., 2003]. In the described procedure, the best father is computed according to the similarity of its children and nothing more. Maedche and Staab note that every descendant of the candidate father should be an important factor to evaluate.

let us see for example the subsequent scenario: I have the new concept *trailer* to evaluate, and the subsequent extracted similarity scores between the concept and other children in the ontology: *box* (similarity score to trailer: 0.8), *house* (0.7), *barn* (0.6), *villa* (0.5) See Figure 2.14.

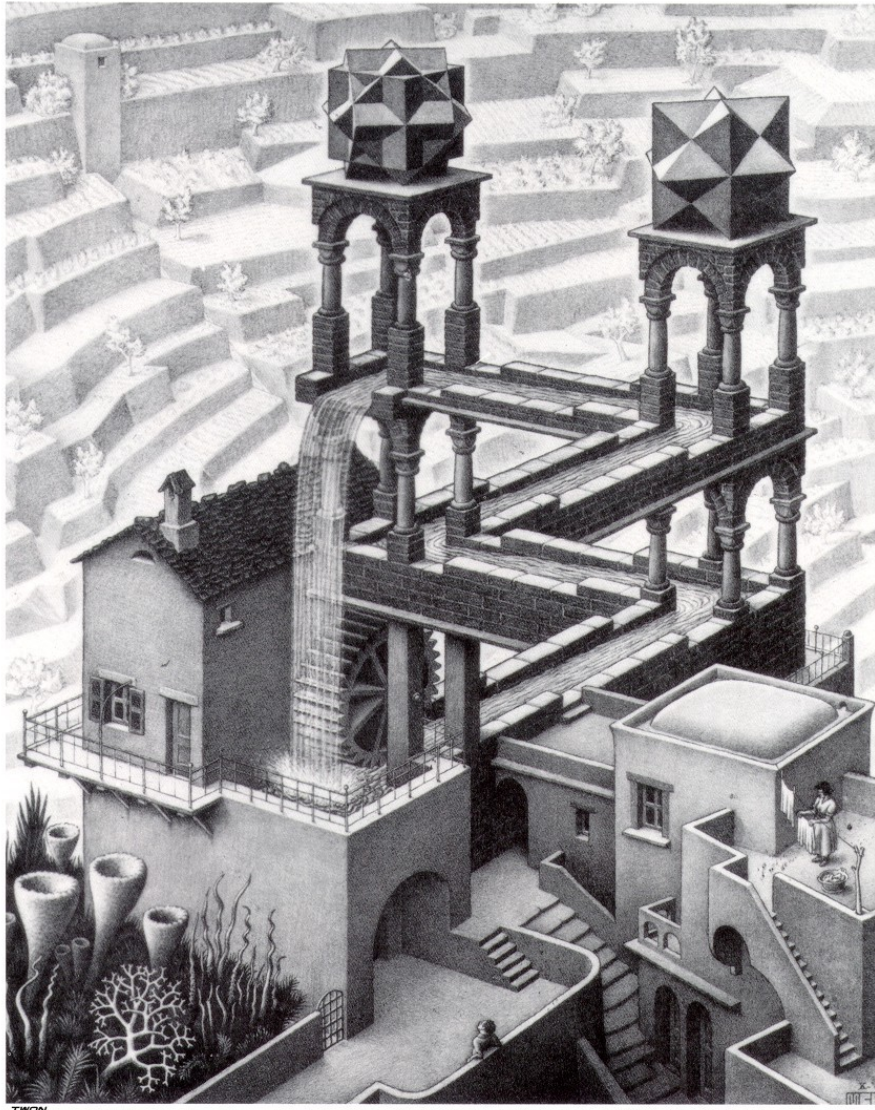


Figure 2.13: M. C. Escher, Waterfall, 1961 - This situation is impossible in the real world of physics: the potential energy of the water falling is, in part, lost for the friction of the rotating wheel, and it is not sufficient again to move back the water to the top of the fall. Also, the water's path from the bottom to the top is geometrically impossible

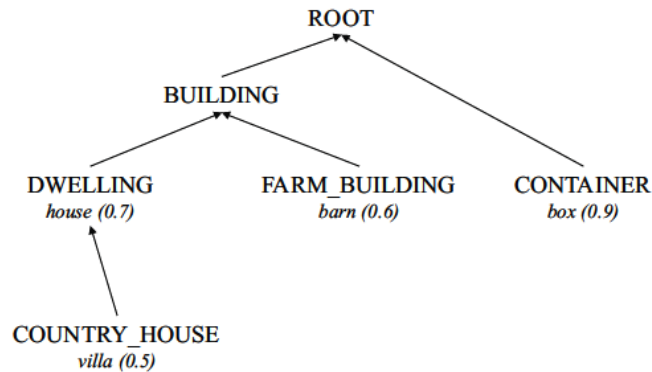


Figure 2.14: A semantic classification scenario, as presented by Maedche and Staab. Candidate fathers are presented as categories (in capital letters), while leaf terms are presented as concepts belonging to one category

In this case, the ordinary methods will classify the concept *trailer* under CONTAINER, since it appears to have biggest similarity to box. However, it is obvious that the most likely class of trailer is in a different part of the ontology: in the set there are three words which, though not belonging to one class, are semantically close to each other. It would thus be safer to assign the new word to a concept that subsumes one or all of the three semantically similar neighbors. For example, the concepts DWELLING or BUILDING could be good candidates in this situation.

Maedche and Staab start defining the notion of Least Common Superconcept between two concepts  $a$  and  $b$ :

$$lcs(a, b) = c \text{ such that } \delta(a, c) + \delta(b, c) + \delta(ROOT, c) \text{ is minimal}$$

where  $\delta(a, b)$  is the distance between  $a$  and  $b$  in terms of the number of edges which need to be traversed. Then they define the *taxonomic similarity*  $\sigma$  between two concepts:

$$\sigma(a, b) = \frac{\delta(ROOT, c) + 1}{\delta(ROOT, c) + \delta(a, c) + \delta(b, c) + 1}$$

where  $c = lcs(a, b)$ .

The voting weight for a certain node is finally computed as:

$$W(n) = \sum_{h \in H(n)} sim(t, h) \cdot \sigma(n, h)$$

where  $t$  is the target word to be classified and  $H$  is the set of hyponyms of node  $n$ . In this way not only the children of a candidate father are evaluated to give it a score, but also all its descendant, with a weight that decreases with the taxonomic distance of the descendant from the candidate father. A different approach, totally similar in its objectives, was attempted by Widdows, for details see [Widdows, 2003].

### 2.1.3 Concept Relations Extraction

The approaches to extract semantic formal relations from text corpora can be divided into two main types: the extraction conducted by looking for pre-constructed patterns (a generalization to relations different than *is-a* of the hearst patterns approach, described in §2.1.2.2), and the extraction based on the identification of (subject, verb, object) triplets, already seen in the distributional similarity approach, in §2.1.2.3 and FCA, §2.1.2.4.

In paragraph §2.1.3.1 and §2.1.3.2 I present two different approaches of the first type. The former looks in the corpus of documents for specific attributes that can be referred to specific entities (e.g. color, height, width...), the latter tries to extract from text the so-called *qualia structures*, that are specific genres of information considered very important to define the knowledge that humans can have of an entity, according to the work of Aristotle about categories [Aristotle, 340]. In paragraph §2.1.3.3 I present the general algorithms applied by different researches for the second type of approach.

#### 2.1.3.1 Learning Attributes by Looking for Pre-Constructed Patterns

An attribute can be defined as any phenomenological quality of a being. Typical attributes are, for example, name or color. In this section I speak about the issue of acquiring attributes automatically from text, according to the writings of Poesio and Almuhareb [Poesio M., 2005] and Philipp Cimiano [Cimiano, 2006].

Attributes are typically expressed in texts using the proposition of, the verb have or genitive constructs, e.g.:

- The color **of** the car
- Every car **has** a color
- The car's color
- Peter bought a new car. **Its** color...

However, we are not only interested in learning the domain (e.g. car) and name for an attribute (e.g. color), but also its range. In fact, values of attributes are expressed in texts in quite different ways, for instance using copula constructs, adjectives or expressions specific to the attribute in question:

1. the car **is** red / his name **is** Peter (copula + value)

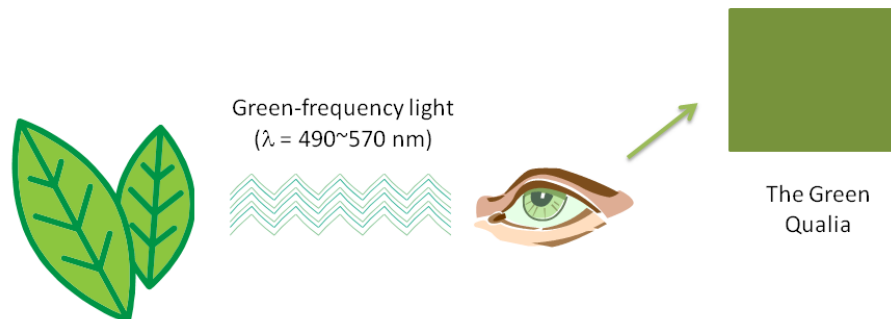


Figure 2.15: A specific frequency of a light reflected from two leaves generates in our perception the experience of green: we can say that the frequency of the light reflected is an attribute of the leaves, while the green experience in our consciousness is the quale

2. Peter **is** 176 cm **tall** (copula + value + adjective)
3. The car **is** 3 meters **long** (copula + value + adjective)
4. The **red** car (adjective)
5. The car **is painted** red / he **is called** Peter / the baby **weighs** 3kg (specific expressions)

All that algorithms do is to look for this specific types of syntactic structures in the text, and extract what they understand to be a specific relation *has-attribute*(domain, range).

### 2.1.3.2 Qualia Structures

The concept of *qualia structures* have been originally introduced by Pustejovsky [Pustejovsky, 1991] and was applied to different research fields of Information Retrieval. Qualia are defined in philosophy as the *subjective qualities of conscious experience*. The concept of *quale* (singular form of qualia) must be distinguished from the one of attribute in the fact that the quale is the way in which the perception of the attribute modifies our conscious experience (see Figure 2.15).

Pustejovsky, referring to Categories work of Aristotle [Aristotle, 340], with the support of more modern theories about qualia argument (e.g., Daniel Dennett [Dennett, 1991]), defines 4 different groups of qualia that should characterize every entity, as perceived by the human consciousness:

- *constitutive type*: describing physical properties of the object as its weight, material as well as parts and components
- *agentive type*: describing factors involved in the bringing about of an object, i.e. its creator or the causal chain leading to its creation
- *formal type*: describing those properties which distinguish an object in a larger domain, i.e. orientation, magnitude, shape and dimensionality
- *telic type*: describing the purpose or function of an object

As we said, the work of Pustejovsky is not directly correlated to the field of Ontology Learning, even if we can easily understand that defining qualia structures about entities we gather, in fact, a lot of important ontological information.

So, even if Pustejovsky imagined his qualia structures as a huge set of tables containing all the information about beings, some researches of the Ontology Learning field tried in the last decade to extract this type of information from the Web, looking for regular patterns, and to store it in the form of an ontology. You can see for example the works of Markert, Modjeska and Nissim [Markert K., 2003], or Cimiano and Staab [Cimiano P., 2004a].

### 2.1.3.3 Learning Relations from SVO triplets

Different researches tried in the more recent years to extract relations from verbal expressions. The work of Gamallo, Gonzalez, Augustini, Lopez and Del Lima [Gamallo P. et al., 2002] is referential, but you can see also Buitelaar, Olejnik, Sintek [Buitelaar P., 2004] and Ciaramita, Gangemi, Ratsch, Saric, Rojas [Ciaramita M. et al., 2005].

Their common method is to extract a set of (subject, verb, object) triplets and then use different heuristics trying to generalize every proposition extracted to an appropriate level of abstraction, that can be put as an information in the ontology.

let us see an example: if we take in consideration the relation *work-for*, and we extract from a corpus the subsequent triplets:

```
(man, work_for, department) (employee, work_for, institute)
(woman, work_for, store)
```

we can certainly say that these triplets are valid pieces of information, but from an ontological point of view we are interested in finding the most general relation describing all the instances that we see in the corpus, that, in this case, could be, for example: *work-for*(person, organization).

Researches, challenging this task, rely on past-constructed hierarchical ontologies, built both manually or semi-automatically with the algorithms seen in §2.1.2, using them as oracles. Algorithms simply try to generalize as much as they can the different instances seen in the corpus looking for them in the oracles: in our case, a good hierarchy would



probably be able to tell us that man, employee and woman are all descendants of the concept person, and, again, department, institute and store are all organizations.

#### 2.1.4 Ontology Learning Algorithms Evaluation

Ontology evaluation, in the software developing process, is part of the *quality assurance* sub-process. Dellschaft and Staab in [Dellschaft K., 2004] summarize the main functional requirements for an ontology valid for the most number of domains of work. An ontology is generally required to be:

1. Consistent
2. Complete
3. Concise
4. Expandable

Dellschaft and Staab suggest to evaluate ontologies in two different dimension: a *functional* and a *structural* one. The functional dimension try to evaluate the quality and usability of an ontology in respect to the software environment which the ontology serves the purpose of. If problems of usability are noticed, probably one or more of the four above listed requirements is not fulfilled.

The problem of the functional dimension is that it does not give us a general and unambiguous method to evaluate the result ontologies of a given algorithm, thus general performances and effectiveness can hardly be evaluated in a common way to every algorithm.

The structural dimension fulfills this requirement, evaluating the ontology in its logical structure. In this document we have evaluated the algorithms according just to this dimension, obviously because we evaluated a lot of algorithms generating ontologies with no idea of what these ontologies are needed for.

there is to say that even if a high evaluation in the structural dimension of an ontology is a good clue suggesting the consistency, completeness, conciseness and extendibility of it, structural evaluation itself is not sufficient to guarantee the fulfillment of these requirements at a level requested by the standard of a software, functional evaluations should always be done to ensure the quality of the software.

Structural evaluation of an ontology usually foresees two different measures of evaluation, **precision** and **recall**. Precision is computed as the number of right relationships found divided by the total number of the relationships in the ontology.

$$precision = \frac{right\ relationships}{total\ relationships\ in\ ontology}$$

Recall is the number of right relationship found divided by the total number of the right relationships.

$$\text{recall} = \frac{\text{right relationships}}{\text{total of the right relationships}}$$

Who decides whether a relationship is right or not? Well, there are two main methods. The **Manual Evaluation Method** foresees the presence of a human expert of the domain of knowledge which the ontology belongs to, that can personally judge whether a relationship is correct or not (precision), or if a relationship in the ontology is missing (recall).

The second method is the **Gold Standard Based Approach**: a corpus of documents and a manual-built ontology that is considered to be the correct representation of the concepts and the logical relations of the corpus are given. The corpus is given as input of the algorithm, and then the result ontology is compared to the referential one, thus precision and recall can be easily computed.

It is understandable that the gold standard test furnish us a general and unique measure which every algorithm can be evaluated against, but the disadvantage is that, existing a limited number of gold standard ontologies, they give us a too little window over the infinite results that a ontology learning algorithm can give in output, thus recommending manual evaluations too.

Given this information, it's now the case to say that the precision evaluations of the algorithms, as reported in the last paragraphs, have an indicative value and cannot be taken as an absolute "score of ability" of the algorithms, because they were acquired in the most of the cases from a little number of ontologies (being the gold-standard ontologies, as said, limited, and the human judgment time consuming), and with different methods (human judgment, gold-standard, other own approaches).

You can use them to have an idea of the results obtained by every researcher, but do not assume that an algorithm with average precision of 40% can be defined "better" than another with 30% of attested precision. There is also to explain that in order to keep as general as possible the evaluation of the algorithms presented in the last paragraphs, recall is not reported, because in fact was evaluated by a very little number of authors.

## 2.2 Inferring fuzzy semantics from text

As anticipated in §1.6, as far as we know, no approach to knowledge acquisition from corpora of unstructured text similar to the one presented in §1.6, and described more deeply in §4.3, was ever done.

Statistical study of distribution of nouns in corpora of documents, considering the verb of the tense in which they appear and their syntactic function as a context, are quite frequent in Ontology Learning from Text. We have seen two examples in the last paragraph, they are ASIUM (§2.1.2.3), and Formal Concept Analysis (§2.1.2.4). These approaches are different in the fact that they make their statistical analysis with the objective of extracting concept hierarchies from text and not fuzzy propositions.

Attempts to extract a sort of fuzzy information from corpora of documents using Correspondence Analysis were found in Fionn Murtagh's book [Murtagh, 2005], his works

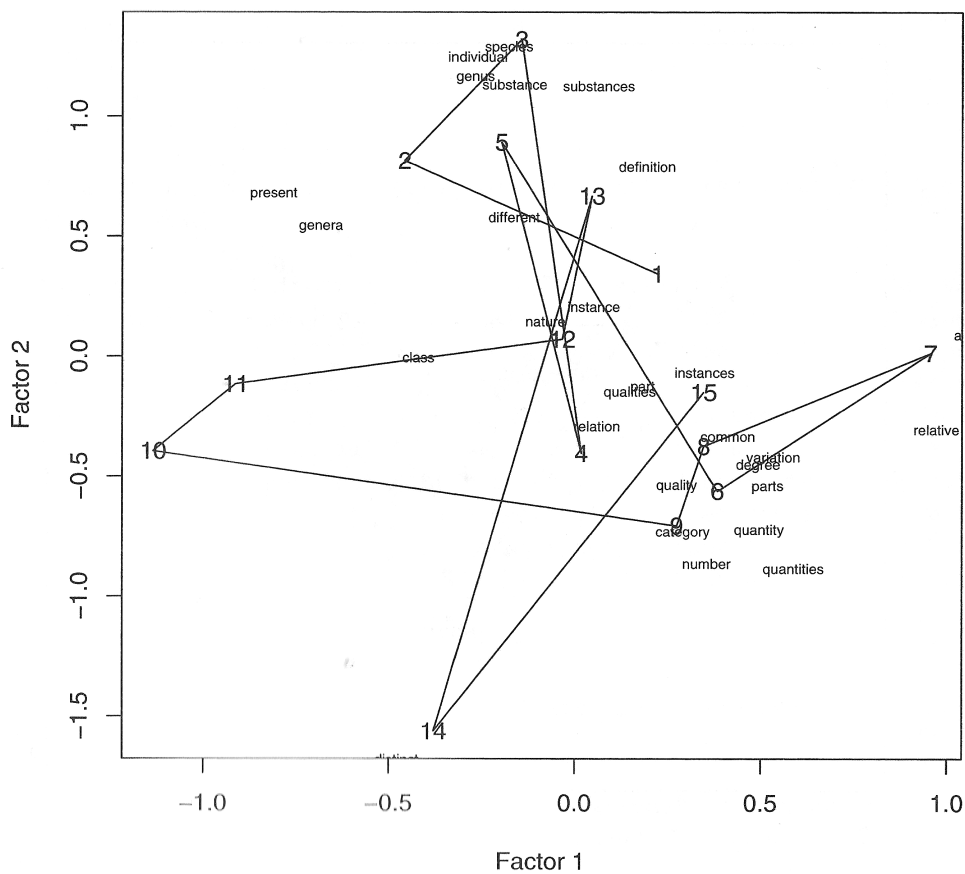


Figure 2.16: Example of a Correspondence Analysis of terms distributed in 15 different parts of Aristotle's "Categories"

refer to older works by Benzécri on linguistics, presented in [Benzécri, 1990].

We can see an example of Fionn Murtagh's study in Figure 2.16. What is analyzed is the distribution of relevant terms appearing in 15 different parts of Aristotle's "Categories" [Aristotle, 340]. As can be seen, the 15 different parts are also represented, allowing Murtagh to trace a path representing the semantic evolution of the treatise in its 15 parts. As can be seen, for instance, parts 6, 8, 9 deal with the same arguments, and these arguments have to do with the concepts, between the others, of "category", "quality" and "degree". Parts 7, 10, 11 and 14 are very far from the others and, probably, deal with arguments different from the ones of the most central parts.

All this information is about semantic similarity, but the difference with our approach here is that at the end no semantic can be extracted, because no proposition can be expressed looking at graphs like the one in Figure 2.16. Ok, maybe something like:

“Part 7 of Aristotle’s dialogue deals with categories”

But this is metainformation about text and it is not so useful. The fact is that semantics represented are about documents (or part of them) constituting the corpus, and not about beings expressed by the corpus itself. This can also be said about every study on linguistics presented by Murtagh in [Murtagh, 2005], because the distribution of terms is always studied using as context the document, our approach instead projects in the Euclidean spaces both concepts and their attributes, allowing a cross-observation of both that can infer semantic information.

## Chapter 3

# Correspondence Analysis

Let us take two different qualities in a population of 1000 individuals, say eyes colour and hair colour. Let us call them respectively  $X_1$  and  $X_2$ . We consider for the variable eyes colour the subsequent modalities:

- blue
- brown
- green
- chestnut

And for the variable hair colour the subsequent modalities:

- blond
- red
- dark
- auburn

We report in a  $X_1 \times X_2$  table  $C$ , in the  $n_{i,j}$  cell, the number of individuals presenting that specific modalities of the  $X_1$  and  $X_2$  qualities.

$C$	blond	red	dark	auburn
blue	159	29	34	142
brown	12	44	115	200
green	27	24	8	49
chestnut	17	26	25	91

We could want to understand, having a look at this table, if it is possible to induce some information about the reciprocal influence of these two variables: whether the appearance of the specific  $i$ th modality give us or not any sort of probabilistic information about the appearance in the same individual of the specific  $j$ th modality, and vice versa.

Do in fact brown-haired people tend to be brown-eyed? Do in fact red-haired people tend to present a green colour of their iris? Matters of this type are typical problems faced by that part of statistics called *multivariate statistics*. Multivariate statistics is a form of statistics encompassing the simultaneous observation and analysis of more than one statistical variable.

Different models were created in order to understand in what way two or more different variables influence each other with their appearance in a specific context. Correspondence Analysis (CA) is one of these models, very peculiar, in fact, for some of its aspects: we are going to see that these peculiarities tend to make it a very powerful tool but also something which results ask to be handled and interpreted with a particular care in respect to different statistical approaches.

### 3.1 History

*Analyse Factorielle des Correspondances* was developed by the French statistician Jean-Paul Benzécri between 1955 and 1962, and was presented in its most complete form at the Collège de France in a course in the winter of 1963. His success as a model to solve multivariate statistical problems arrived in the '70s, hand in hand with the diffusion in different research laboratories of computers able to handle, in a reasonable time of elaboration, the model's algorithms applied to large amounts of data.

What made CA model so famous was probably its ability to solve a very large band of different problems, thus, a great number of data analysts, working in very different application fields, found in CA a unified processing framework, available in a single software package. In 1976 Benzécri wrote "L'Analyse des Données" [Benzécri, 1976], that can be considered the most referential writing about his work in the two previous decades. In 1981 Fionn Murtagh completed his doctorate as a student in the statistics course of study of Benzécri, at the Collège de France of Paris. He is now a member of the Royal Irish Academy, and a professor of Computer Science at the University of London.

Murtagh conducted different works in these latter years starting from the awareness that a computer able to handle very large amounts of data as input for CA is available nowadays in every local electronics store, and new software architectures were developed since the '70s, especially in the '90s. Both Murtagh and Benzécri seem quite convinced that with this new computational abilities at humans' disposal new more advanced statistical techniques will be soon discovered and adopted by statisticians, and CA will become a deprecated technique. Anyway, in the hope that this will not happen so much soon, Murtagh spent different efforts in presenting how the CA framework can be integrated in softwares created with the last technologies and for the last hardwares: he decided in particular to keep as a reference the Java language and R to make easily available the approach of CA to new developers (you can see [Murtagh, 2005]).

It is interesting to note that CA represents for both Benzécri and Murtagh something more than a simple mathematical approach, they prefer to talk about a *philosophy of -a conceptual approach to- data, and data interpretation* ([Murtagh, 2005], Preface). They see CA as a special way to observe phenomena happening in the totality of the cases, and try to understand some aspects of the latent sense hidden between their expression.

This allows us to understand how much is important for them, after the CA algorithms have given some results, to conduct a careful interpretation of what data express in order to have the best possible understanding of the underground world of behaviors, influences, causes and concauses that generated the phenomena we observe.

## 3.2 The Framework

We are going now to specify the different algebraic passages applied by the CA technique to a context of occurrences of different modalities of two different specific variables. These steps will obtain as output some results that constitute a summary representation of the distributional behavior of the different modalities that the two variables can assume.

### 3.2.1 Context

let us now take a different example from the one presented in the introduction of the chapter, the former has probably a background easier to understand, but our new example has more to do with this thesis work: we want to observe and report the occurrences of some nouns ( $X_1$ ) and verbs ( $X_2$ ) in the tenses of a corpus of documents. Let us create a matrix  $M$  placing in the cell  $n_{i,j}$  the amount of occurrences of the  $i$ th noun as a subject of the  $j$ th verb:

$M$	eat	play	shoot
<b>man</b>	60	12	60
<b>cat</b>	20	54	5
<b>chicken</b>	32	3	2
<b>gun</b>	1	2	125

So, every  $n_{i,j}$  represents the number of tenses present in the corpus of documents with the  $i$ th noun as subject and the  $j$ th verb.

We want to see in what way the occurrence of a specific noun as subject influences the probability to occur of a verb in the same tense: it is quite evident to anyone that specific nouns tend to be subject of specific verbs with more probability in respect to others, but we want to analyze in the specific context of these 4 nouns and 3 verbs what is the measure of this influence. Also, as said in §1.5, the more similar is the behavior of the distribution of occurrences of two nouns in respect to the context of the verbs, the more we expect from these nouns to be similar from a semantic point of view. And the same can be said for the distribution of the verbs in respect to the nouns. So, the world behind the data, the underground to be uncovered is, in this case, the real world of entities and their actions expressed by the corpus of documents.

### 3.2.2 First Passages

The first step of the algorithm is to calculate the grand total of the individual observations and to divide each number in the cells for this grand total, in order to have a matrix  $M_{prob}$  expressing in the  $n_{i,j}$  cell the probability of co-occurrence of the two  $(i, j)$  modalities. In our case the grand total is 376, and the probability matrix results:

$M_{prob}$	eat	play	shoot
<b>man</b>	0.159	0.031	0.159
<b>cat</b>	0.053	0.143	0.013
<b>chicken</b>	0.085	0.007	0.005
<b>gun</b>	0.002	0.005	0.332

Then the sums of the values of each row and each column are calculated, we will call the sum of the values of the  $i$ th row  $F_i$  and the sum of the values of the  $j$ th column  $F_j$ :

$M_{prob}$	eat	play	shoot	$F_i$
<b>man</b>	0.159	0.031	0.159	35%
<b>cat</b>	0.053	0.143	0.013	21%
<b>chicken</b>	0.085	0.007	0.005	10%
<b>gun</b>	0.002	0.005	0.332	34%
$F_j$	30%	19%	51%	

Being  $\sum_{n=0}^i F_i = 1$  and  $\sum_{n=0}^j F_j = 1$ , it is preferred to report  $F_i$  and  $F_j$  as percentages, every percentage represents the contribution of the  $i$ th or  $j$ th modality to the total of the occurrences.

Here the algorithm forks to analyze separately row and column modalities. For a matter of order, I will proceed in explaining the two different paths separately, even if they are composed exactly by the same passages, applied first to what we will call *Row Profiles* and then to what we will call *Column Profiles*.

### 3.2.3 The Rows Space

#### 3.2.3.1 Row Profiles

Let us consider the rows of our matrix:

$M_{prob}$	eat	play	shoot	$F_i$
<b>man</b>	0.159	0.031	0.159	35%
<b>cat</b>	0.053	0.143	0.013	21%
<b>chicken</b>	0.085	0.007	0.005	10%
<b>gun</b>	0.002	0.005	0.332	34%
$F_j$	30%	19%	51%	



The algorithm now divides every  $n_{i,j}$  probability by the  $F_i$  value. What we obtain is a matrix whose rows represent the so-called row profiles:

$M_{rowprof}$	eat	play	shoot
<b>man</b>	45%	10%	45%
<b>cat</b>	25%	68%	7%
<b>chicken</b>	85%	7%	5%
<b>gun</b>	1%	1%	98%
$F_j$	30%	19%	51%

As you can see, row profiles are also a collection of percentages, being the sum of the values of every row equal to 1 (100 in our table, being expressed as percentages). Row profiles are a very important element of CA because they represent the pure distributional behavior of row modalities (the nouns, in our case), independently from the original amount of occurrences we were dealing with. So, we can now see that the term “man” has a probability of 45% to occur in our corpus of documents as a subject of “eat”, of 10% as a subject of “play”, of 45% as a subject of “shoot”.

The last  $F_j$  row represents the average behavior of the different row profiles. This average profile is also very important because the comparison between row profiles and this last row can inform us about how much the variable  $X_1$  depends on the variable  $X_2$ . In fact, it’s easy to understand that if all the row were exactly equal to the average behavior, all the row would have exactly the same behavior against the context represented by the variable  $X_2$  and this context would not take influence at all over the variable  $X_1$ . So, the more the single row profiles are different from the average profile, the more the table is showing us a dependency of the variable  $X_1$  (the nouns, in our case) from the variable  $X_2$  (the verbs, in our case).

The divergences of the single row profiles from the average profile can be measured with the  $\chi^2$  test of independence.  $\chi^2$  distance between the  $l$  row profile and  $k$  row profile is computed as:

$$\chi^2(l, k) = \sqrt{\sum_j \frac{(n_{l,j} - n_{k,j})^2}{F_j}}$$

The sum of all the  $\chi^2$  tests applied to all row profiles in respect to the average profile represents the *total inertia* of the matrix in respect to his rows. In our case this value is:

$$I = \chi^2(man, F_j) + \chi^2(cat, F_j) + \chi^2(chicken, F_j) + \chi^2(gun, F_j) = 3.84$$

Inertia represents the total amount of divergence of the row profiles from the assumption of independence. The higher is this number, the higher is the probability of an interdependence between the two modalities.

There’s also something to say about the  $\chi^2$  distance between two different row profiles: the obtained value represents how much two different rows diverge in their distributional behavior against the  $X_2$  context: the more similar is this behavior, the more there

should be, in some sense still to be understood, a similarity between the entities represented by the two rows (in our case these entities are the nouns appearing as subjects in tenses).

### 3.2.3.2 Projection in the Euclidean Space

The main objective of CA is to provide a summary representation of the different similarities present between the different modalities. In order to do so, the different row modalities are projected to an Euclidean space of  $\theta$  dimensions, where:

$$\theta = \min(i - 1, j - 1)$$

This space has different and very interesting properties:

- The Euclidean distance between the modalities in this space of representation is exactly equal to the  $\chi^2$  distance of their row profiles in the  $M_{rowprof}$  matrix, calling the  $\alpha$ th dimension of the  $i$  row  $F_\alpha(i)$  we can state:

$$\chi^2(l, k) = \sqrt{\sum_j \frac{(n_{l,j} - n_{k,j})^2}{F_j}} = \sqrt{\sum_\alpha (F_\alpha(l) - F_\alpha(k))^2}$$

- The origin of the axis is placed in the barycenter of the different row profiles in respect to the  $\chi^2$  distance measure, that is, as we said, the average row profile
- Axes are not chosen at random, but they are selected to have along them, from the first to the  $\theta$ th, in decreasing order, the maximum possible distribution of the projection of the elements

Let us see an example to understand this last important aspect: a cloud of points is placed in a 2-dimensional space (Figure 3.1). We want to roto-traslate this cloud of points in a new Euclidean space where the distribution of this cloud of points is maximized along the axes of the new space. We start selecting the first axis as the line on which the distribution of the projection of the points is maximized, passing over the barycenter of the cloud of points. In our case is, more or less, the line traced in Figure 3.2 in a red colour. The second axis must be selected as perpendicular to the first one (Figure 3.3). The cloud of points can then be represented in the new space at maximum distribution (Figure 3.4).

In order to create this space the algorithm proceeds as follows: let us go back to our original probabilities matrix:

$M_{prob}$	<b>eat</b>	<b>play</b>	<b>shoot</b>	$F_i$
<b>man</b>	0.159	0.031	0.159	35%
<b>cat</b>	0.053	0.143	0.013	21%
<b>chicken</b>	0.085	0.007	0.005	10%
<b>gun</b>	0.002	0.005	0.332	34%
$F_j$	30%	19%	51%	

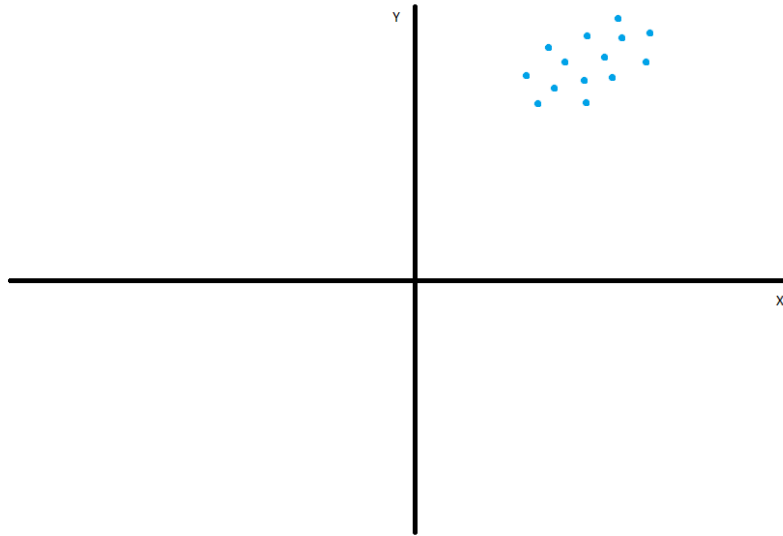


Figure 3.1: A cloud of points in a 2-dimensional space

Every number  $n_{i,j}$  is divided by the square root of the relative  $F_i$  and by the root square of the relative  $F_j$ , we obtain for our matrix  $M_{prob}$  the subsequent matrix  $M_n$ :

$M_n$	<b>eat</b>	<b>play</b>	<b>shoot</b>
<b>man</b>	0.491	0.123	0.376
<b>cat</b>	0.211	0.721	0.040
<b>chicken</b>	0.494	0.058	0.023
<b>gun</b>	0.008	0.021	0.797

$M_n$  can be seen as a collection of values normalized in respect to the population of the matrix.

In the subsequent passage the algorithm generates a similarity matrix  $i \times i$  computed as  $M_{sim} = M_n(M_n)^T$ :

$M_{sim}$	<b>man</b>	<b>cat</b>	<b>chicken</b>	<b>gun</b>
<b>man</b>	0.399	0.208	0.259	0.307
<b>cat</b>	0.209	0.566	0.147	0.049
<b>chicken</b>	0.259	0.147	0.248	0.024
<b>gun</b>	0.307	0.049	0.024	0.636

the number  $n_{i_1,i_2}$  represents a similarity index between the  $i_1$  term and the  $i_2$  term, the higher is the number, the higher is the similarity of behavior of the terms' row profiles.

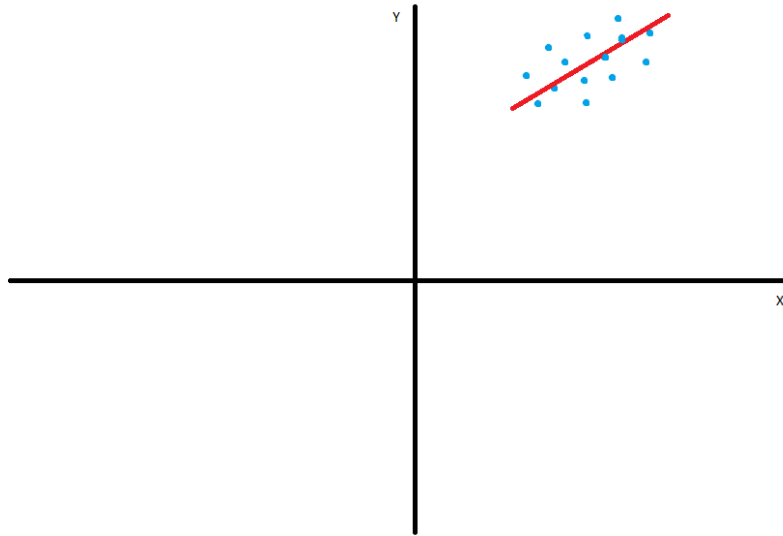


Figure 3.2: The axis that maximizes the distribution of the projection of the points on itself is selected

In fact, as you can see, the highest number in a row (or column) can be seen between a term and itself. Also, for example, a cat results more similar to a man and a chicken than to a gun.

At this point, the operation of *eigendecomposition* is applied to this matrix. Theory of linear algebra says that every square matrix  $A_{N \times N}$  can be expressed as:

$$A = Q\Lambda Q^T$$

Where  $Q$  is a  $N \times N$  matrix, with the  $i$ th column defined as the  $i$ th eigenvector of the matrix  $A$ .  $\Lambda$  is a  $N \times N$  diagonal matrix, its  $\lambda_{ii}$  value is defined as the  $i$ th eigenvalue of the matrix  $A$ .

We can represent the computed  $Q$  and  $\Lambda$  for our matrix  $M_{sim}$ :

$Q(M_{sim})$	1	2	3	4
<b>man</b>	-0.592	-0.004	0.416	0.689
<b>cat</b>	-0.458	-0.682	-0.566	-0.056
<b>chicken</b>	-0.313	-0.262	0.635	-0.655
<b>gun</b>	-0.583	0.682	-0.319	-0.303

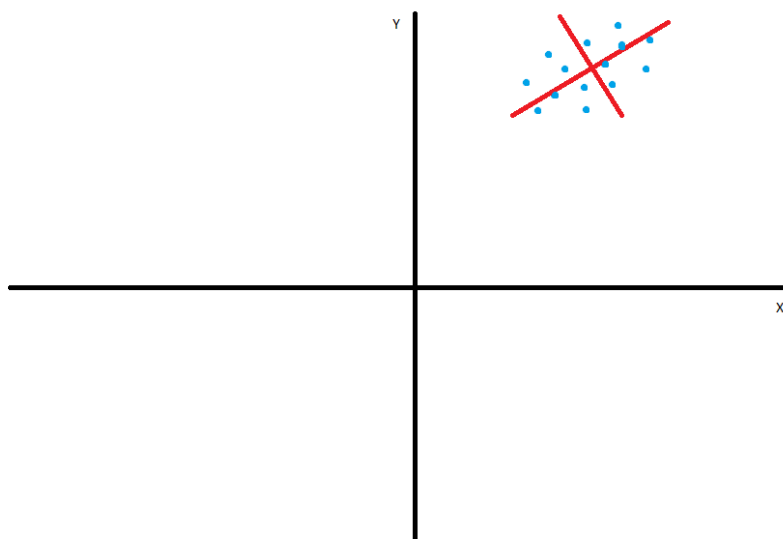


Figure 3.3: The second axis is selected perpendicular to the first one

$\Lambda(M_{sim})$	1	2	3	4
<b>man</b>	1	0	0	0
<b>cat</b>	0	0.575	0	0
<b>chicken</b>	0	0	0.275	0
<b>gun</b>	0	0	0	0

Eigenvectors are very important because in our problem each eigenvector represents one of the axes at maximum distribution that we mentioned before in this paragraph (and that can be seen in red in Figure 3.3): we can consider the row (or column, being the same) vectors of  $M_{sim}$  as coordinates in the space, and now the row vectors of  $Q(M_{sim})$  represent the same elements translated in a new space at maximum distribution.

We remember now what we are looking for: an Euclidean space where the  $\chi^2$  distance between any two row profiles is equal in this space to the Euclidean space between the same two row elements. In order to arrive to this representation, another two passages must be done.

The matrix  $Q(M_{sim})$  elements are divided by the root square of the  $F_i$  value of the row, we obtain:

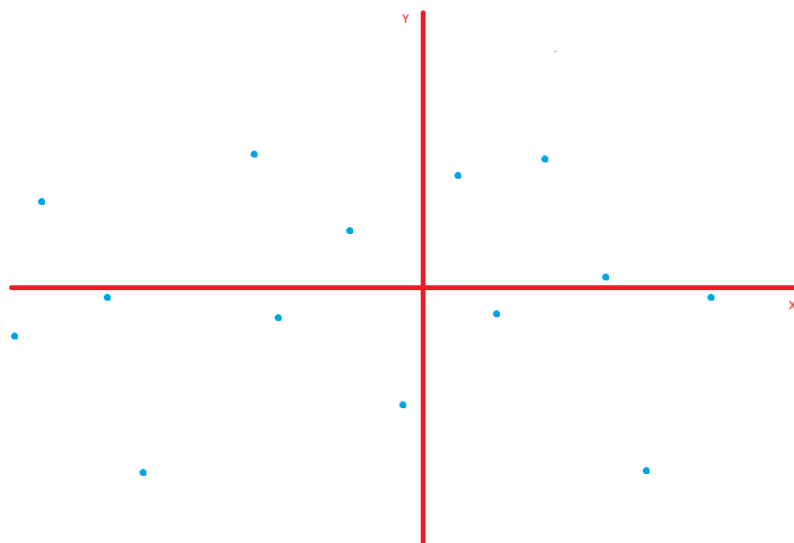


Figure 3.4: The new space

$Q(M_{sim})_{norm}$	1	2	3	4
<b>man</b>	-1	-0.008	0.703	1.163
<b>cat</b>	-1	-1.489	-1.235	-0.123
<b>chicken</b>	-1	-0.836	2.025	-2.089
<b>gun</b>	-1	1.169	-0.547	-0.520

We can also report Eigenvalues in a simpler vector form:

	1	2	3	4
$\lambda_\alpha$	1	0.575	0.275	0

You can see something important: the first eigenvector places every row element in the same coordinate, and this provides no distributional information at all. This always happens and is caused by the fact that the row profiles are not linearly independent, being the sum of all of them equal to 1. The consequence in the generation of the eigenvalues is the presence of one of them carrying no information. This first eigenvector is simply discarded.

There is also another thing to notice in this specific case: the last eigenvalue is equal to 0... what does it mean? We calculated in §3.2.2 the total inertia of the row profiles matrix. The eigenvalues represent an important information about the axis they refer to: the ratio between the  $\alpha$ th eigenvalue and the sum of all the eigenvalues is the percentage of the total inertia the  $\alpha$ th axis contributes to.

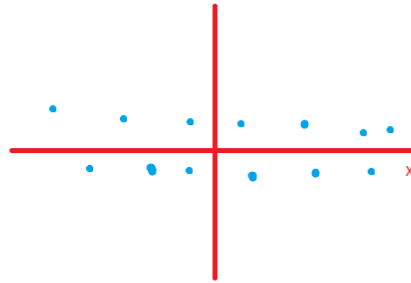


Figure 3.5: X axis carrying higher inertia

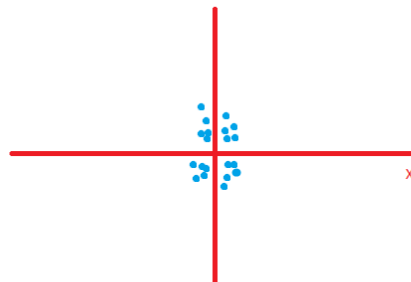


Figure 3.6: X axis carrying lower inertia

The concept of inertia is directly connected to the one of distribution of the cloud of points over the axes in the Euclidean space: we said that total inertia is the sum of the deviations of every row profile from the average profile, and this deviations are transformed in our Euclidean space in specific distances of the elements from the center of the origin. Saying that an axis carries more or less inertia means that row elements in the Euclidean space are more or less distributed over that specific axis. You can see Figures 3.5 and 3.6.

So, you can understand that even the 4th eigenvector has to be discarded, because

its contribution to the total inertia is null, being its correspondent eigenvalue equal to 0. This can be also understandable if we calculate the number of dimensions  $\theta$  that the algorithm is able to represent, as we said:

$$\theta = \min(i - 1, j - 1) = 2$$

The maximum number of dimensions representable is 2, and so exceeding dimensions simply return eigenvalues equal to 0.

Let us represent what remains:

$Q(M_{sim})_{norm}$	2	3
<b>man</b>	-0.008	0.703
<b>cat</b>	-1.489	-1.235
<b>chicken</b>	-0.836	2.025
<b>gun</b>	1.169	-0.547

	2	3
$\lambda_\alpha$	0.575	0.275

The last passage of the algorithm is to divide any  $n_{i,\alpha}$  value in the matrix  $Q(M_{sim})_{norm}$  by the square root of the  $\alpha$ th eigenvalue.

What we finally obtain are the coordinates of the rows Euclidean space we were looking for:

$F_\alpha(i)$	2	3
<b>man</b>	-0.006	0.368
<b>cat</b>	-1.129	-0.648
<b>chicken</b>	-0.634	1.062
<b>gun</b>	1.886	-0.287

let us just check the Euclidean distance between "man" and "cat" in this space:

$$\begin{aligned} dist(man, cat) &= \sqrt{[F_2(man) - F_2(cat)]^2 + [F_3(man) - F_3(cat)]^2} \\ dist(man, cat) &= \sqrt{(-0.006 + 1.129)^2 + (0.368 + 0.648)^2} = 1.515 \end{aligned}$$

And  $\chi^2$  distance between "man" and "cat" row profiles in the row profiles matrix:

$$\begin{aligned} \chi^2(man, cat) &= \sqrt{\sum_j \frac{(n_{man,j} - n_{cat,j})^2}{F_j}} \\ \chi^2(man, cat) &= \sqrt{\frac{(0.454 - 0.253)^2}{0.30} + \frac{(0.091 - 0.0683)^2}{0.189} + \frac{(0.454 - 0.063)^2}{0.511}} = 1.515 \end{aligned}$$

As you can see, the result is the same.



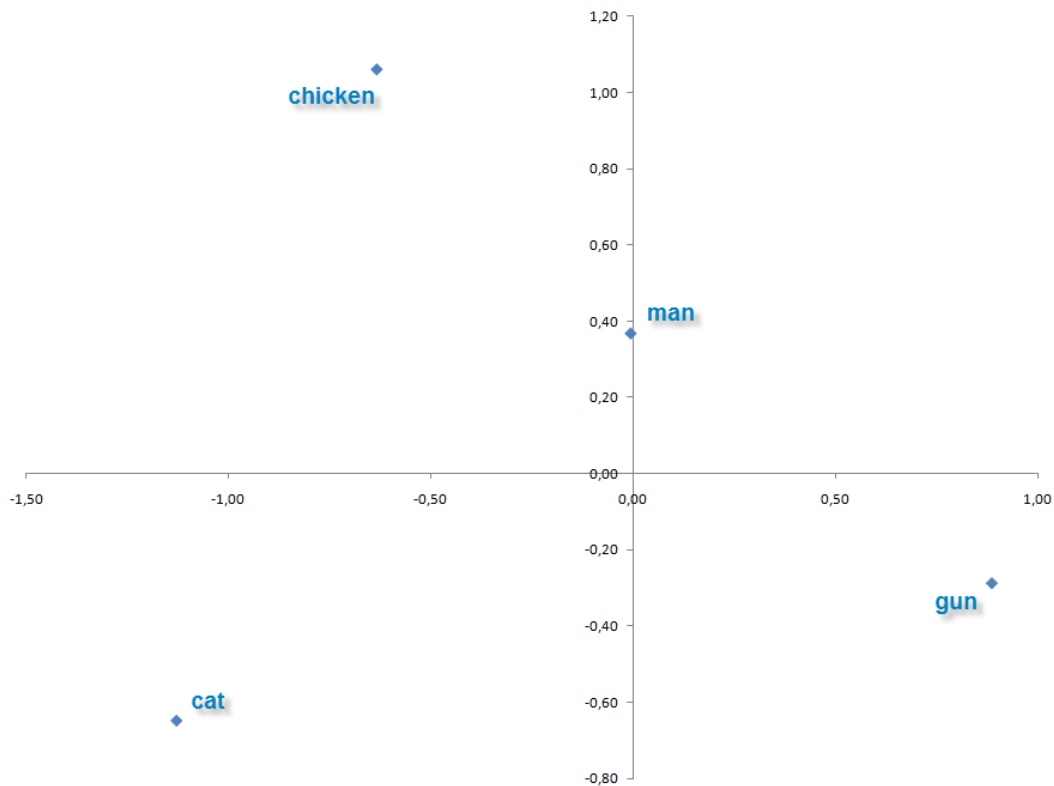


Figure 3.7: Final rows space representation

### 3.2.3.3 The Visual Representation

In order to obtain a visual representation of the the distributional behavior of the row elements it is used to represent the two axes with the highest inertia (so, highest eigenvalues) in a 2-dimensional plot (Figure 3.7). The significance of this type of plot is in proportion to the percentage of contribution to the total inertia of the two axes represented: if the two axes carry together at least 70%-80% of the inertia it would probably mean that we are looking at a good representation of the row elements distribution. Less percentages mean a probable loss of information in respect to the total amount of the axes. In our results we have just 2 axes, so we are looking at a 100% contribution to the total inertia, and the representation have no loss of precision.

### 3.2.4 The Columns Space

The passages to generate the columns space are exactly the same, but applied to the column profiles vectors.

### 3.2.4.1 Column Profiles

Let us now take in consideration the columns of our matrix:

$M_{prob}$	<b>eat</b>	<b>play</b>	<b>shoot</b>	$F_i$
man	0.159	0.031	0.159	35%
cat	0.053	0.143	0.013	21%
chicken	0.085	0.007	0.005	10%
gun	0.002	0.005	0.332	34%
$F_j$	30%	19%	51%	

We divide every  $n_{i,j}$  probability by the  $F_j$  value. What we obtain is a matrix whose columns are the so-called column profiles:

$M_{colprof}$	<b>eat</b>	<b>play</b>	<b>shoot</b>	$F_i$
<b>man</b>	53%	17%	31%	35%
<b>cat</b>	18%	76%	3%	21%
<b>chicken</b>	28%	4%	1%	10%
<b>gun</b>	1%	3%	65%	34%

Everything said for the row profiles is again valid. The  $F_i$  column is dually the column that represents the average behavior of the different profiles. We are going now to project this columns in an Euclidean space where the Euclidean distance between any two elements is equal to the  $\chi^2$  distance between their correspondent column profile.

### 3.2.4.2 Projection in the Euclidean Space

let us go back to our normalized matrix computed in 3.2.3.2:

$M_n$	<b>eat</b>	<b>play</b>	<b>shoot</b>
<b>man</b>	0.491	0.123	0.376
<b>cat</b>	0.211	0.721	0.040
<b>chicken</b>	0.494	0.058	0.023
<b>gun</b>	0.008	0.021	0.797

The similarity matrix has now dimensions  $j \times j$  and is computed as  $M_{sim} = (M_n)^T M_n$ :

$M_{sim}$	<b>eat</b>	<b>play</b>	<b>shoot</b>
<b>eat</b>	0.531	0.242	0.212
<b>play</b>	0.242	0.539	0.094
<b>shoot</b>	0.212	0.094	0.078

Eigendecomposition is applied to the matrix, computed  $Q$  and  $\lambda_\alpha$  for this matrix  $M_{sim}$  are:

$Q(M_{sim})$	1	2	3
<b>eat</b>	-0.548	-0.357	0.756
<b>play</b>	-0.434	-0.650	-0.622
<b>shoot</b>	-0.714	0.670	-0.201

	1	2	3
$\lambda_\alpha$	1	0.575	0.275

As you can notice, eigenvalues are exactly equal to the eigenvalues of the rows space. We will discuss about this in §3.2.5.

The first eigenvector is discarded again, the matrix  $Q(M_{sim})$  elements are divided by the square root of the  $F_j$  value of the column, and then any  $n_{j,\alpha}$  value is divided by the square root of the correspondent  $\lambda_\alpha$  eigenvalue. The obtained columns coordinates in the final Euclidean space are:

$G_\alpha(i)$	2	3
<b>eat</b>	-4.494	0.723
<b>play</b>	-1.136	-0.751
<b>shoot</b>	-0.711	-0.147

### 3.2.4.3 The Visual Representation

The visual 2-dimensional representation of the column elements is in Figure 3.8.

## 3.2.5 Superimposition of the spaces

Rows and Columns space obtained have lot in common:

- They have the same number of dimensions
- They have the same eigenvalues associated to every dimension
- They have the same inertia, and, so, the same distribution of elements along the same axes

They are in fact very similar under different aspects. Anyway, besides all these properties, there's a deeper relation that can be demonstrated to exist between rows and columns modalities coordinates  $F_\alpha(i)$  and  $G_\alpha(j)$  in their new Euclidean spaces, that is:

$$F_\alpha(i) = \frac{1}{\sqrt{\lambda_\alpha}} \sum_j \frac{n_{i,j}}{F_i} G_\alpha(j)$$

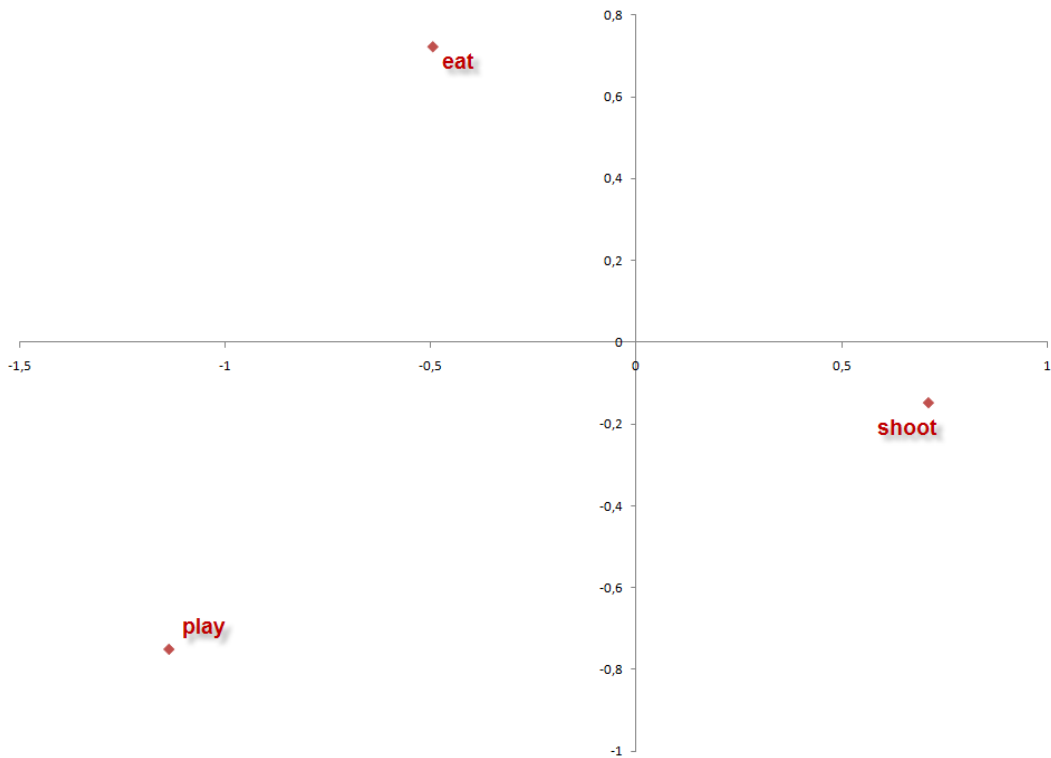


Figure 3.8: Final columns space representation

$$G_{\alpha}(i) = \frac{1}{\sqrt{\lambda_{\alpha}}} \sum_j \frac{n_{i,j}}{F_j} G_{\alpha}(i)$$

That's very interesting: we are stating that the position of a row element in the  $\alpha$ th dimension is, a part a constant, the barycenter of all the column elements, considered as masses having as weight their correspondent value in the row element's row profile.

The second formula dually says that the position of a row element in the  $\alpha$ th dimension is, a part a constant, is the barycenter of all the row elements, considered as masses having as weight their correspondent value in the column element's column profile.

Let us see for example the relation between the first coordinate of the row element "man"  $F_2(man)$  and all the first column elements coordinates  $G_2(j)$ :

$$F_2(man) = \frac{1}{\sqrt{\lambda_2}} \left\{ \frac{M_{prob}[man, eat]}{F_{man}} G_2(eat) + \frac{M_{prob}[man, play]}{F_{man}} G_2(play) + \frac{M_{prob}[man, shoot]}{F_{man}} G_2(shoot) \right\}$$

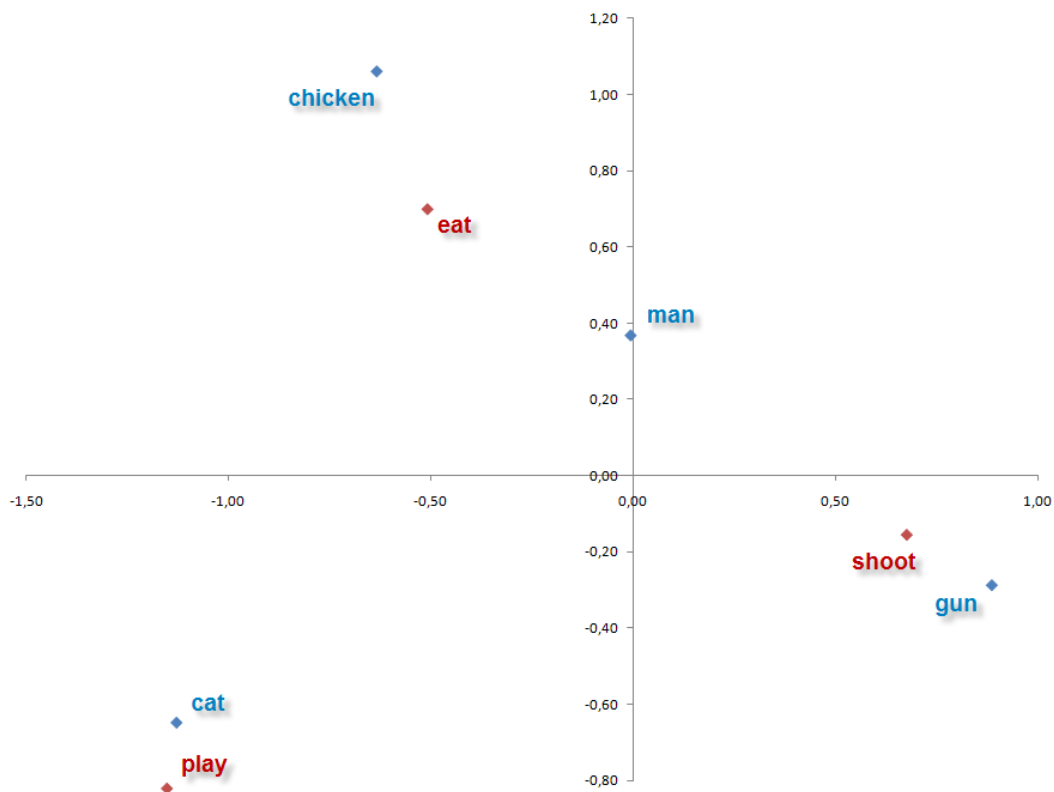


Figure 3.9: Superimposition of rows and columns space

The higher is the value associated to a verb in the row profile of “man”, the nearer “man” will be to that verb, being the barycenter of the column elements considered as masses having that precise value as weight. The same can be said for every row element in respect to *all* the column elements, and for every column element in respect to *all* the row elements.

Considering these relations, we can superimpose the two representations of the rows and columns space (Figure 3.9 and 3.10), with the consciousness that, with a particular care, we can infer informations also from this particular view.

### 3.3 Interpretation of Results

As we saw, we can obtain from the framework different pieces of information that can be examined under different aspects. The hardest part comes just right now: the ability of the statistician stands in understanding the sense of this information, identifying specific

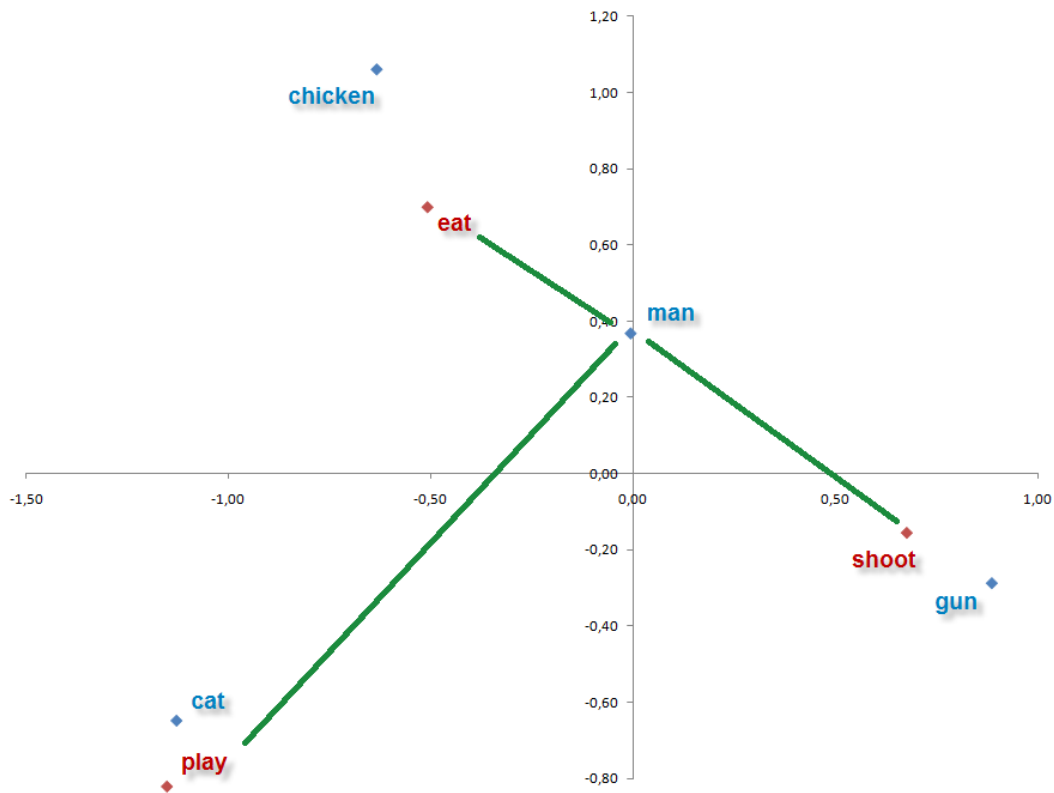


Figure 3.10: “man” is the barycenter of the weighted column elements, you can see it standing near to “eat” and “shoot”, and far away from “play”

data behaviors that have to be reconducted to specific influences and concauses in the world we are dealing with.

### 3.3.1 Proximity between modalities of the same variable

A proximity between two modalities of the same variable means a significant similarity of behavior of two different elements, indicating a connection between them: they are “similar”, in a way to be understood. In our case (Figure 3.9), for example, we can see that “chicken” is nearer to “man” than to “cat”, this means a more similar behavior (for the context we are looking at) to “man” than to “cat”. This is probably caused by the low number of verbs we are looking at, a larger collection of verbs would probably tend to reapproach “chicken” and “cat”, being both animals. Anyway, from the data we are looking at, “chicken” results more similar to “man”.

Also, positions of modalities can be examined in respect to the average profile, repre-

sented by the center of the plot. A modality far away from the origin represent a *remote* modality, indicating that his nature is, in some way to be understood, peculiar in respect to other modalities. *Typical* modalities are, instead, modalities very near to the origin and their nature is quite typical in respect to the totality of the modalities. In our case “man” is very near to the origin, meaning that its behaviour is the more typical, in fact, a man can shoot just as a gun, and can eat an play just as an animal.

### 3.3.2 Axes

Axes are the lines along which the elements have their maximum distribution. Being these lines the most important “paths” over which modalities are set, they could be re-conducted to a specific significance, or, in other words, they could imply a specific distinction that cannot immediately be seen looking at the matrix data, but it’s surely evident in our plot view.

For example, just having a look at Figure 3.7, we can desume that the X axis moves on the left the animals and on the right the objects. Man are placed in the middle because they are animals, but can execute actions that can be done also by things (in this single case, to shoot). The highest is the inertia associated to the axis, the more is probable for it to have an important significance in our hidden world of influences and concauses.

Let us make a different example in brief: just imagine an matrix  $M$  characterized by the two variables  $X_1$  and  $X_2$  being respectively the 600 students of a school and different subjects. let us put in the  $n_{i,j}$  cell the average of the votes of the  $i$ th student in the  $j$ th subject. What probably will appear making a CA over the subject profiles is a principal axis of distribution moving on a side the humanistic subjects, and on the other side the scientific ones, being probably the distributional behaviors of two humanistic subjects (as, for example, Greek and Latin) more similar to each other in respect to other scientific matters (as Maths or Physics) against the votes of the students. So, in this last example, the principal axis denotes a hidden reality, the humanistic/scientific attitude of a subject, that could hardly be identified by simply looking at the main matrix of the average votes of hundreds of students.

### 3.3.3 Proximity between modalities of different variables

Proximity between variables of different modalities imply again a strong and significant connection. There’s to say that approximation of inertia (by reductions in the representation of dimensions) could cause distances between modalities of different variables to be deceptive with a higher probability than distances between modalities of the same variables. Proximities evidence should be then checked calculating the effective proximity in the  $\theta$ -dimensional space, or, at least, in a space preserving a high grade of inertia.

As we said, our example has just two dimensions, so the visual plot carries 100% of inertia. We can see for example in Figure 3.9 that “cat” is very near to “play”: we can infer, as in fact it is, that cat is the entity between the four presented with the higher attitude to play.

That was also easy, in our simple example, to be understood looking at the original matrix  $M$ , but CA is able in more complex situations, with a larger amount of data, to extract important relations that are not immediately evident, but just latent in the values distribution in the matrix.

### 3.4 Multiple Correspondence Analysis

All we have done in this chapter, and all that can be met in this thesis work dealing with statistics, regards that part of multivariate statistics that challenges the specific problem of the simultaneous observation and analysis of 2 variables in a generic system. It's important to notice that Multivariate statistics is, in fact, the more general problem of the simultaneous observation and analysis of  $n > 1$  variables.

CA was historically born as a study of relations between 2 variables, but was then generalized to its multiple form to study the relations between  $n > 1$  variables. I summarize here the reasons why this chapter does not speak about Multiple CA:

- Simplicity of presentation of the framework
- Multiple CA is not used in any algorithm this thesis work deals with
- Fionn Murtagh's seen works themselves ([Murtagh, 2005], [Murtagh, 2007]) never deal with multiple CA

Information about Multiple CA generalized approach and its methods can be found in [Benzécri, 1976].

### 3.5 Final Comments

We saw in this chapter that Correspondence Analysis is a representation in the Euclidean space of the modalities of a variable with their respective distances representing the  $\chi^2$  distance between their row (or column) profiles.

It seems here appropriate to say something about this form of representation, what is its sense? Well, row (column) profiles represents, as said in §3.2.3.1, the pure distributional behavior of a modality against its context. Pure is said in the sense that it is independent from the population of the matrix.

Why do not projecting row (column) profiles as they are in an Euclidean space, considering their Euclidean distance as a distance measure between the terms? Well, reason is that superimposition of the spaces of different variables would be unjustified, for the principles presented in 3.2.5.

So, what is the consequence of the evaluation of  $\chi^2$  in respect to Euclidean distance? We said that  $\chi^2$  is not so different from Euclidean distance between two vectors, we presented  $\chi^2$  distance between two  $l$  and  $k$  vectors as:



$$\chi^2(l, k) = \sqrt{\sum_j \frac{(n_{l,j} - n_{k,j})^2}{F_j}}$$

The only difference in respect to the Euclidean distance, as can be noticed, is that the evaluation of a distance in a specific axis is weighted in inverse proportion to  $F_j$ , the total population of the modality that represents the  $j$ th context. This results in the consequence that *more importance is given to less populated contexts*, because they contribute more in deciding whether two modalities are near or distant. Whether this is a good feature or not, it has to be decided in the context of the application where CA is used.



## Chapter 4

# The Extraction Tool

Extraction is our own solution for the acquisition of concept hierarchies (and other fuzzy semantics, in its latest version) from corpora of unstructured text. According to what is presented in §1.5 and §1.6, the tool was developed in three different versions, that are:

- The **Version 1.0**, developed by Davide Eynard, ended in September, 2009
- The **Version 2.0**, developed by me, that extends the 1.0 Version with new features, available from December, 2009
- The **2.0 Verbal Version (2.0VV)**, always developed by me, that differs from the 2.0 Version in the fact that, according to what is presented in §1.6, it analyzes the distribution of terms under the different point of view of their action and affection attributes. It is available from February, 2010

For an ordered presentation, the three versions are described in three different separate sections, respectively, §4.1, §4.2 and §4.3. §4.4 is dedicated to a general summary and to final comments.

### 4.1 Extraction 1.0

Extraction version 1.0 is a first application of the techniques presented by Fionn Murtagh in [Murtagh, 2005] and [Murtagh, 2007] for the extraction of concept hierarchies from unstructured text. It is already characterized by many of the features presented by the model in §1.5, even if that model's last instantiation is represented by Extraction version 2.0 (§4.2).

#### 4.1.1 User Requirements

The main user requirements satisfied by the application are presented in Figure 4.1. The user has the possibility to:

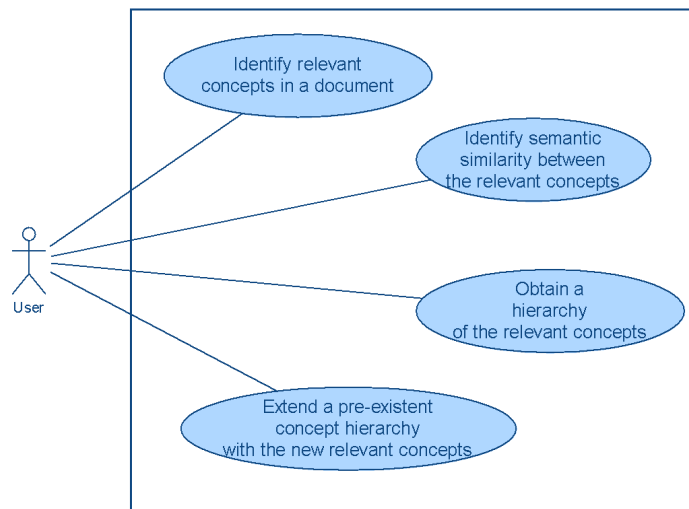


Figure 4.1: Extraction 1.0 - Main requirements, use case diagram

- **Identify** a number at his will of **relevant concepts in the document**, the more a concept is considered relevant the more it contributes to characterize the domain of knowledge which the corpus of documents belongs to
- **Identify semantic similarity between relevant concepts**
- **Obtain a hierarchy of the relevant concepts**
- **Extend a pre-existent hierarchy with the new relevant concepts**

#### 4.1.2 Main Conceptual Entities

The main conceptual entities that compose the application can be seen in Figure 4.2, they are:

- **Training Corpus of Documents:** the referential corpus of documents that the user has to provide in order to extract relevant terms from the test corpus of documents. The score of relevance of a term is, in fact, computed in respect to both the training and test corpus, considering the relevance of a term its contribution in distinguishing one of the two corpus from the other, as explained in §4.1.4.3
- **Test Corpus of Documents:** the set of documents from which the terms are identified and extracted

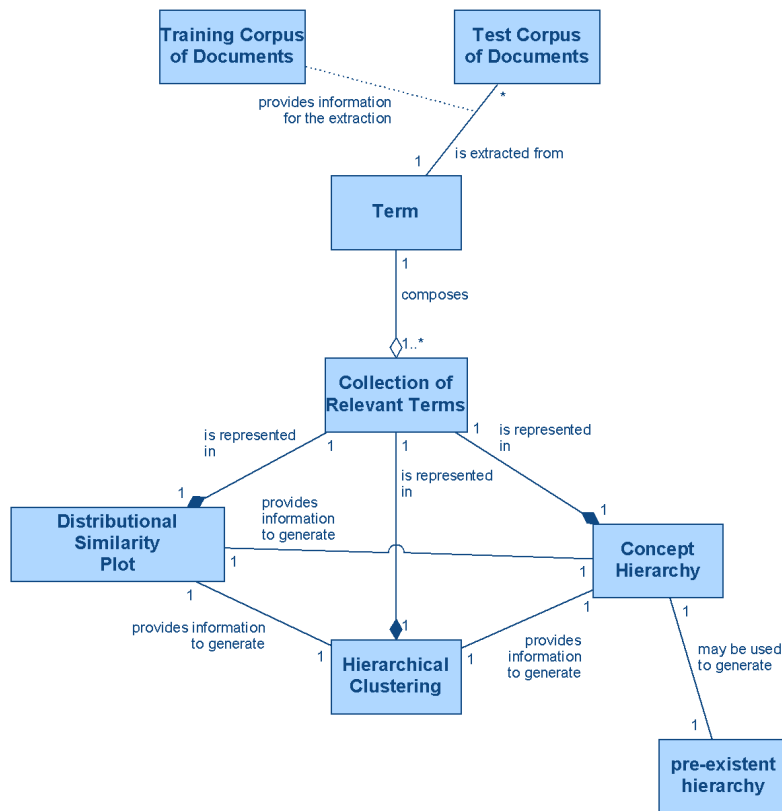


Figure 4.2: Extraction 1.0 - Main conceptual entities, class diagram

- **Term:** a term is a single string of characters expressing a concept, identified and extracted from the test corpus of documents
- **Collection of Relevant Terms:** a set of the  $n$  terms ( $n$  is specified by the user) with the highest score of relevance
- **Distributional Similarity Plot:** also referred as 2D-plot, it is a collection of the relevant terms, plus a pair of coordinates  $P_t(x, y)$  associated to each term. These coordinates represent the location of the term in the 2-dimensional euclidean space created according to the principles of Correspondence Analysis (Chapter 3). As presented in §1.5, the nearer two terms are in this space, the more we expect a semantic similarity between the concept they express
- **Hierarchical Clustering:** the hierarchical clustering graph is an ordered taxonomy of the terms, generated from the similarity measures between the terms expressed by the Distributional Similarity Plot. The process of creation of this representation is presented in §4.1.4.5

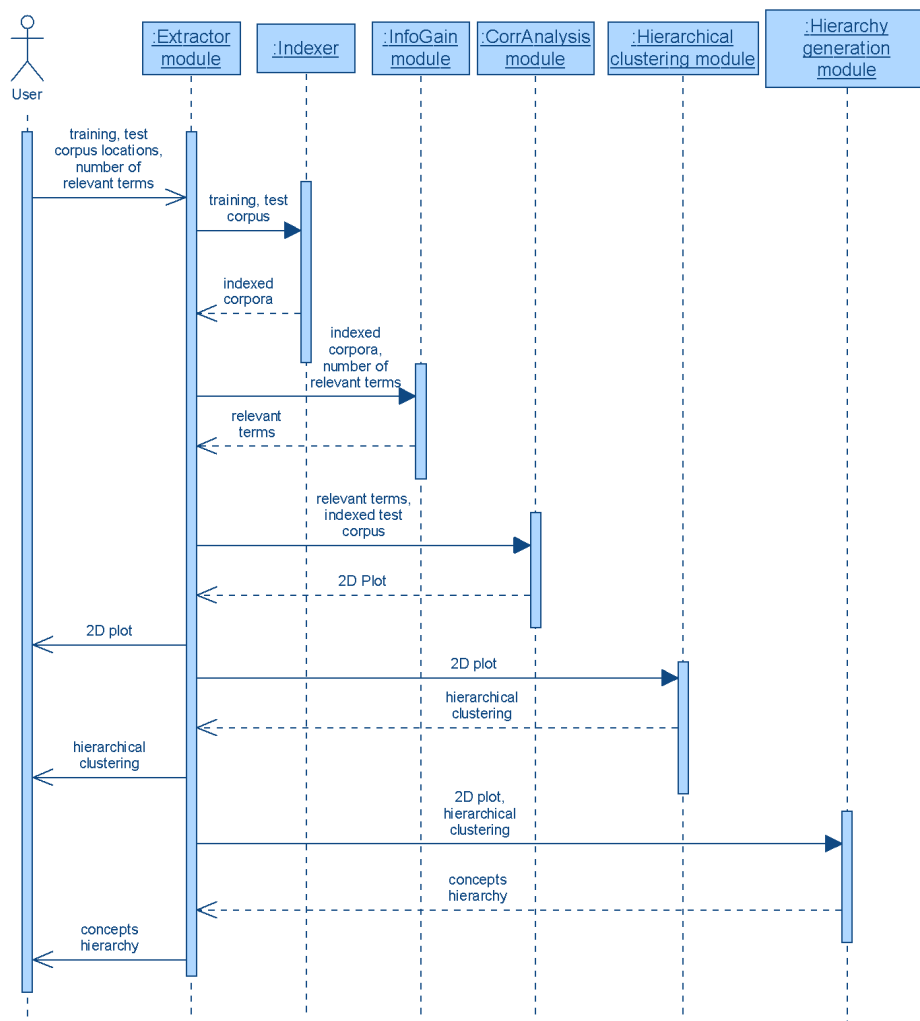


Figure 4.3: Extraction 1.0 - Standard interaction scenario, sequence diagram

- **Concept Hierarchy:** the final ontology expressing *is-a* relationship between relevant terms extracted, obtained starting from the hierarchical clustering representation, according to the process described in §4.1.4.6

### 4.1.3 Use Cases

The standard interaction scenario with the user is showed in Figure 4.3. Here follows a formal description of its main features:

#### Primary Actor

A user

#### Scope

- Identify relevant concepts in the document
- Identify semantic similarity between relevant concepts
- Obtain a hierarchy of the relevant concepts

#### Success Guarantees

The subsequent informations are textually printed on the video:

- Relevant terms
- 2D coordinates of every term in the euclidean space
- Hierarchical clustering of the terms
- Hierarchy of the terms

#### Preconditions

- Training and test corpora of documents are composed by ASCII-8 only-text documents, no other format is allowed
- Training and test corpora of documents are composed by at least 2 documents for each

#### Main Successful Scenario

- The user indicates the path of the training corpus, the test corpus and the number of relevant terms he wants to extract
- The user visualizes on the video the relevant terms, the 2D coordinates of every term in the euclidean space, the hierarchical clustering of the terms, the hierarchy of the terms

#### Alternatives

There are few alternatives to the standard scenario, the main different possibilities, for the user, are:

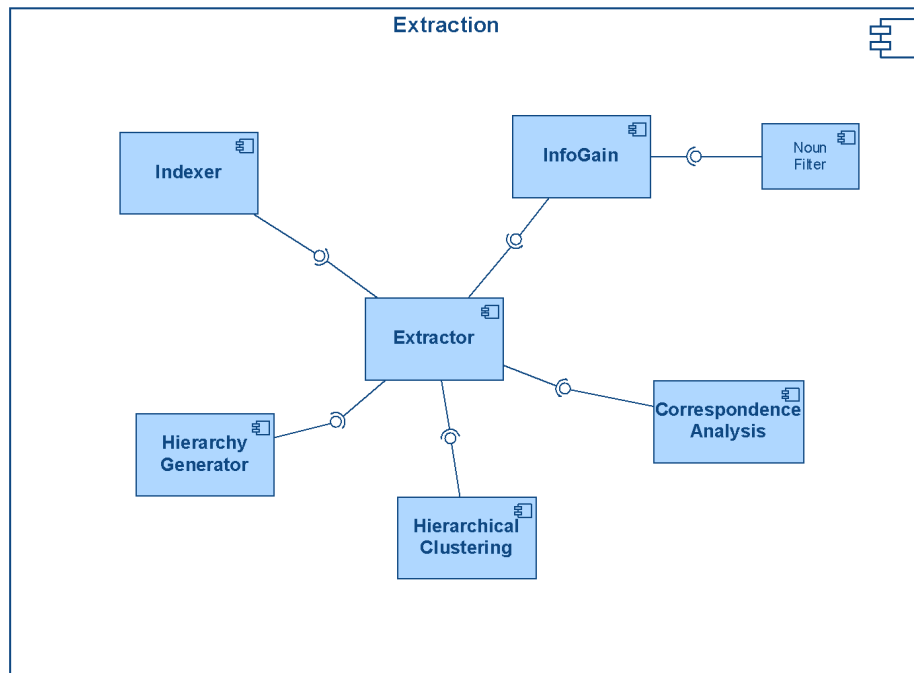


Figure 4.4: Extraction 1.0 - Main modules of the application, component diagram

- To express, for the test or the training corpus, instead of the path of the folder where the documents are located, the path of the folder where an index of a corpus (previously indexed) is located, avoiding the time-consuming process of documents indexing
- To select a file name where to save the generated hierarchy in different ontology specification format, selectable between N3 [W3C, 1998], OWL [W3C, 2004], and TURTLE [W3C, 2001]
- To select a previous existent ontology to be expanded by the algorithm
- To select an option that activates a pass-noun filter, that is a filter applied by the InfoGain module to the relevant terms that discards every term not recognized as a noun, using as a reference an archive of concepts obtained from WordNet [Miller, 1995]



#### 4.1.4 Main Modules of the Application

The main modules that compose the application are shown in Figure 4.4; you can refer also to Figure 4.3 for their role in the application.

##### 4.1.4.1 Extractor

This is the backbone of the application, it acquires the input of the user, returns the output on the video, manages the shared data and orchestrates the tasks. These latter, summarizing, are:

- Indexing of test and training corpus
- Acquisition of relevant terms
- Projection of the terms in a 2-dimensional graph, according to their distributional similarity
- Hierarchical Clustering of the terms
- Generation of the final hierarchy

##### 4.1.4.2 Indexer

Indexer generates indexes of the test and training corpora of documents in order to rapidly handle the research of terms in the documents. The indexes are saved in a pre-selected location, and they can be indicated to the program in a subsequent execution, in order to not execute again the indexing of the same corpus.

##### 4.1.4.3 InfoGain

The InfoGain module calculates an InfoGain score of relevance for the terms found in the documents, and returns the  $n$  most relevant terms (with  $n$  specified by the user). The measure of InfoGain is based on the concept of entropy. As already presented in §2.1.2.3 (Distance Metric), the entropy of a random variable  $X$ , distributed as the probability function  $p(X)$ , can be defined as a measure of its uncertainty, and it can be computed as follows:

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$

In our case, we want to calculate the entropy of a group of documents distributed in the two training and test corpora. We can identify the  $P(X)$  function as the probability for a

document to appear in a corpus  $i$ , computed as the number of the documents in  $i$  divided by the total number of documents of  $i$ :

$$p(i) = \frac{\text{documentsIn}(i)}{\text{totalDocuments}(i)}$$

Where  $i$  can be training or test. So, our entropy measure for a group of documents  $D_g$  distributed in the two  $i$  corpora is measured as:

$$H_{D_g} = - \sum_i p(i) \log_2 p(i)$$

In this way, we have associated to a group of documents an entropy measure. We identify three different groups of documents:

- The group of all documents,  $D_{total}$
- Documents presenting the  $t$  term in them,  $D_t$
- Documents not presenting the  $t$  term in them,  $D_{not}$

We define  $entropy(D_g)$  function as the entropy value of the  $D_g$  group, and we specify that  $|D_g|$  indicates the number of documents in the  $D_g$  group. InfoGain score of term  $t$  is then calculated as:

$$IG(t) = entropy(D_{total}) + \frac{|D_t|}{|D_{total}|} entropy(D_t) - \frac{|D_{not}|}{|D_{total}|} entropy(D_{not})$$

According to what is written in §1.5, the InfoGain measure is a function that returns low results for terms that are very common in both training and test corpora. Terms that, instead, are very common in just one of the two corpora make InfoGain return high results. So, training corpus covers the role of confrontation context in respect to the test corpus: terms that are very frequent in test corpus can both be characterizing terms of the corpus, or very useless terms as conjunction, common adverbs, common verbs... If a term  $t$  is very frequent also in the training corpus, which refers to a topic different from the test corpus,  $t$  is supposed to be a useless term, and, in fact, it will receive a lower InfoGain value. Vice versa, if  $t$  is very frequent in test corpus but not in training corpus it is supposed to be a characterizing term of the test corpus, and in fact, it will receive a higher InfoGain value.

#### 4.1.4.4 Correspondence Analysis

According to the techniques explained in Chapter 3, a (test documents)  $\times$  (relevant terms) matrix is generated, placing in the cell  $n_{i,j}$  the number of occurrences of the  $j$ th term in the  $i$ th document, as, for example:

	<b>caligula</b>	<b>city</b>	<b>group</b>	<b>...</b>
<b>D1</b>	34	120	0	...
<b>D2</b>	0	10	120	...
<b>D3</b>	200	160	4	...
...	...	...	...	...

Distributional similarity between terms is calculated and a collection  $P_t(x, y)$  of coordinates is collected, that represent the position of every  $t$  term in the euclidean space generated by the CA algorithm.

#### 4.1.4.5 Hierarchical Clustering

This module uses a bottom-up approach for the generation of the Hierarchical Clustering representation, according to what is shown in 2.1.2.3 (Hierarchical Clustering). Let us see a simple example: we have just obtained from the projection the subsequent four terms:

- pen(0,0)
- pencil(2,2)
- eraser(-8,-8)
- orange(20,20)

The algorithm starts looking for the two nearest elements. They are pen and pencil, so, they are grouped in a single cluster (Figure 4.5, A). This cluster is considered a new single element located in the barycenter of the elements that compose it ((1, 1), in our case). pen and pencil are no longer elements to be considered. So, the elements now are:

- pen, pencil(1,1)
- eraser(-8,-8)
- orange(20,20)

The algorithm then looks again for the two nearest elements: they are the barycenter of pen and pencil (1,1), and eraser(-8,-8). a new cluster is created and its location is the barycenter of the points (1,1) and (-8,-8), that is (-3.5,-3.5), the new clustering step is traced in the graph (Figure 4.5, B). The elements are now:

- (pen, pencil), eraser(-3.5,-3.5)
- orange(20,20)

In the last step the algorithm looks again for the two nearest elements, they are the two elements left, and the last clustering step is done (Figure 4.5, C).

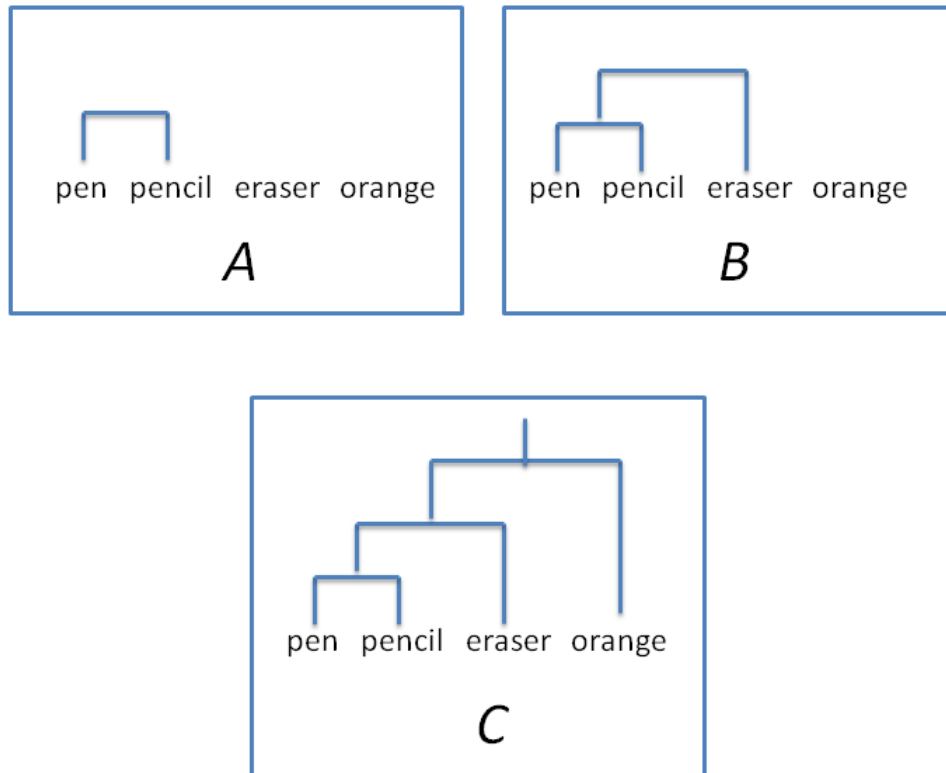


Figure 4.5: Simple example of Hierarchical Clustering

This view (Figure 4.5, C) can be considered a summary view of semantic similarity between terms, the more two terms are near in the 2-dimensional graph, the less steps are to be made in this graph to go from one term to the other (a step is considered as a move from a node to another node in the graph). Anyway, distance measures are no longer kept, so, this graph, even if it preserves a distance representation, carries lower information respect to the 2-dimensional graph.

#### 4.1.4.6 Hierarchy Generator

This module applies the technique presented by Fionn Murtagh in [Murtagh, 2007] for the generation of a hierarchy of concepts from a hierarchical clustering representation. Let us continue with the previous example, to understand how it works: we have four terms in their hierarchical clustering representation (Figure 4.6, A).

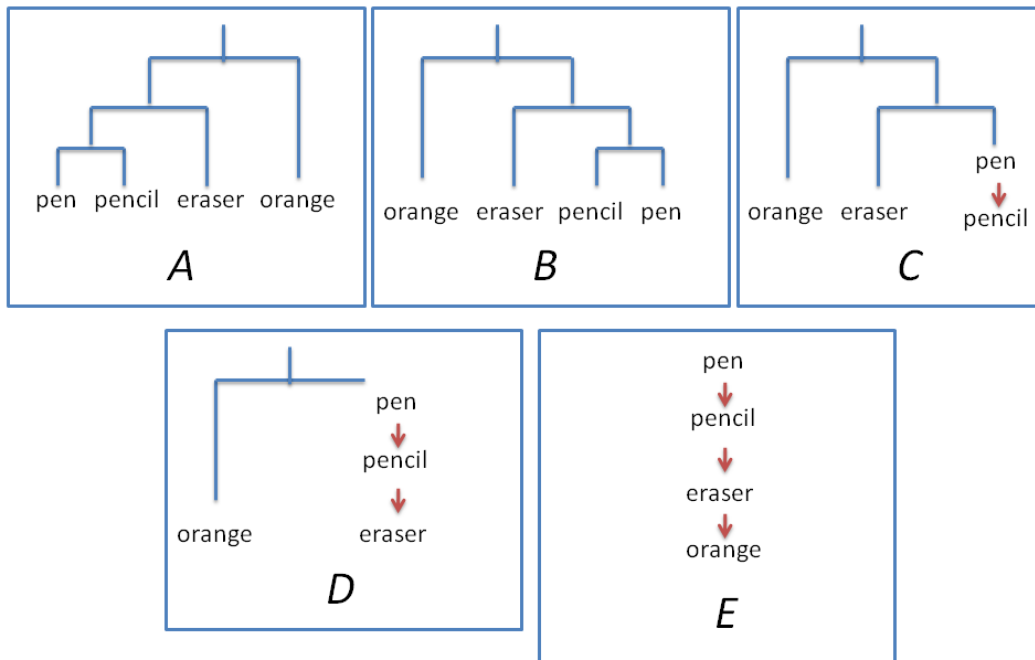


Figure 4.6: Simple example of Hierarchy generation, according to Fionn Murtagh's approach

The algorithm starts ordering from right to left the clusters according to their proximity to the origin in the 2-dimensional representation. This is done because terms appearing closer to the origin are expected to be more general terms, being their occurrences distributed over the maximum number of documents (see §3.3.2), so, terms are organized in order to use this information in the algorithm, wanting from these more general terms to be in a higher position in the hierarchy. In our case, the representation is exactly specular to what we had (Figure 4.6, B).

The approach is then very simple: starting from the first created cluster (by the hierarchical clustering algorithm), going on, what is found on the right is an hyponym of what is found on the left. The first created cluster (it is the lowest in the representation) presents `pen` and `pencil`. Then, `pen` is an hypernym of `pencil` (Figure 4.6, C).

Again, what is found on the right, `pen` and `pencil`, is hypernym of what is found on the left, `eraser` (Figure 4.6, D).

At last, what is found on the right, `pen`, `pencil` and `eraser`, is hypernym of what is found on the left, `orange` (Figure 4.6, E). This final representation is considered our hierarchy.

### 4.1.5 Execution Environment and Deployment

The program is developed in Java language and can be executed by any Java Virtual Machine version 1.6. According to what noticed by Fionn Murtagh (§3.1), an ordinary PC is sufficient in its performances to execute Extraction, even if very huge corpora of documents could make one consider to export the application on more efficient machines. Some performance tests are presented in §5.10.

## 4.2 Extraction 2.0

This version of the program is the complete instantiation of the model presented in 1.5.

### 4.2.1 User Requirements

The main user requirements are the same as the 1.0 version, you can refer to §4.1.1.

### 4.2.2 Specification of the Interface

The tool was provided with a Graphic User Interface model that allows the user to settle the options, to have a visual representation of the 2D-plot generated by the Correspondence Analysis module, and to automatically open a generated hierarchy with a provided external tool (GrOwl, see §4.2.5.2).

The interfaces provided by the application are:

- the **Main Window**, where the user can indicate the location of the corpora of documents or the indexes of the corpora, the number of relevant terms to be visualized and filtering options (Figure 4.7)
- the **2D-Plot Window**, that represents in a 2-dimensional space the distribution of the terms, according to the results of Correspondence Analysis (Figure 4.8)
- a **Term Summary Window**, that provides information on the visualized terms, as location and distance from the origin (4.9)
- a **Hierarchy Algorithm Selector**: where the user can select the algorithm to apply for the generation of the hierarchy (according to what presented in §4.2.2 (Use Cases))
- the **Hierarchy GrOwl Window**: generated by the GrOwl external tool provided with the Application (see 4.2.5.2), it allows the visualization and modification of a created hierarchy (Figure 4.10)

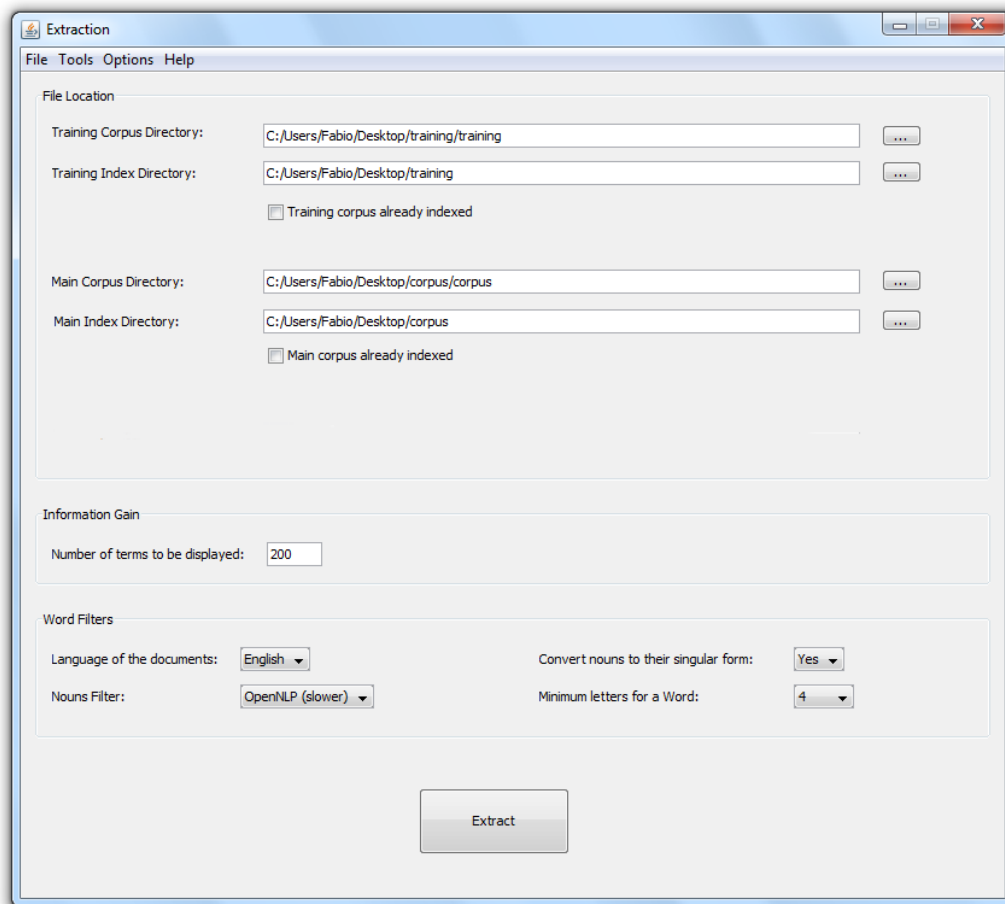


Figure 4.7: Extraction 2.0 - Graphic User Interface main window

### 4.2.3 Main Conceptual Entities

Conceptual entities are the same as in version 1.0, you can refer to §4.1.2.

### 4.2.4 Use Cases

The user interaction changes from the first version (§4.1.3) in the fact that it is mediated by a Graphic User Interface, anyway, the standard interaction scenario remains the same, except for the fact that it provides some additional **alternatives**:

- It is possible to modify an option for each one of three different filters, in order to activate them or not: they are a pass-noun filter (different from version 1.0's

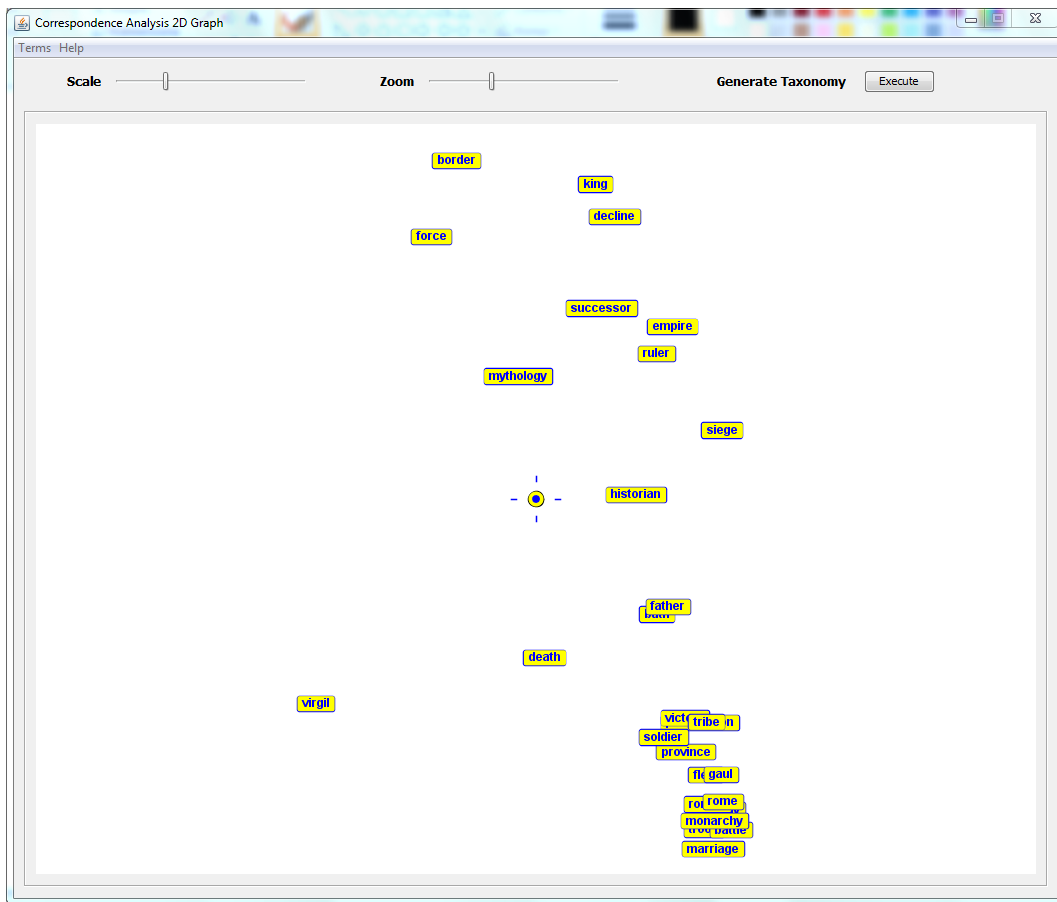
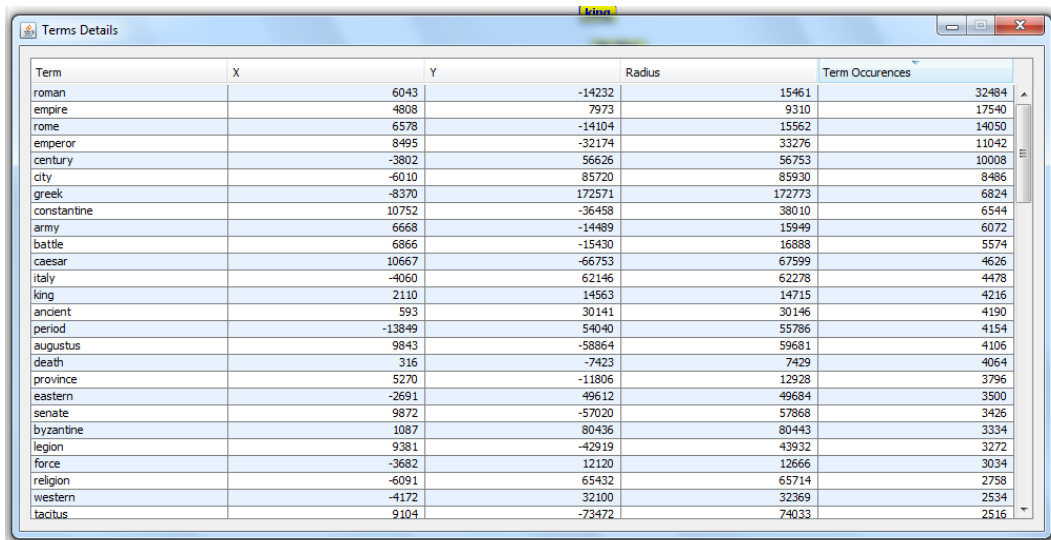


Figure 4.8: Extraction 2.0 - Example of 2D-plot window

WordNet filter), a number-of-letters filter and a singularizing filter. Their role is explained in-depth in §4.2.5.4

- It is possible to select between different methods for the creation of the hierarchy of concepts, the options are described in §4.2.5.6
- It is possible to select whether to generate the hierarchy from all the terms visualized in the plot, or the  $n$  most frequent terms, or a group of selected terms
- It is possible to activate an option for opening the generated hierarchy in an external tool, as described in §4.2.5.6
- It is possible to activate an option to have the possibility, during the execution of some hierarchy generator approaches, to be asked to suggest a hypernym for a term if no hypernyms have been found for it.





Term	X	Y	Radius	Term Occurrences
roman		6043	-14232	15461
empire		4808	7973	9310
rome		6578	-14104	15562
emperor		8495	-32174	33276
century		-3802	56626	56753
city		-6010	85720	85930
greek		-8370	172771	172773
constantine		10752	-36458	38010
army		6668	-14489	15949
battle		6866	-15430	16888
caesar		10667	-66753	67599
italy		-4060	62146	62278
king		2110	14563	14715
ancient		593	30141	30146
period		-13849	54040	55786
augustus		9843	-58864	59681
death		316	-7423	7429
province		5270	-11806	12928
eastern		-2691	49612	49684
senate		9872	-57020	57868
byzantine		1087	80436	80443
legion		9381	-42919	43932
force		-3682	12120	12666
religion		-6091	65432	65714
western		-4172	32100	32369
tactus		9104	-73472	74033

Figure 4.9: Extraction 2.0 - Example of terms summary information window

## 4.2.5 Main Modules of the Application

Main modules set changes from version 1.0 because three new modules have been added (in yellow in the component diagram in Figure 4.11). Also, the functionalities of some modules were deeply modified.

Here follows the description of both new and modified modules, the ones not described here remain the same as version 1.0, you can refer for them to §4.2.5:

### 4.2.5.1 Graphic User Interface

This module handles the visualization of the GUI and the interaction between the Extractor module and the user inputs.

### 4.2.5.2 GrOwl

GrOwl [University of Vermont, 2003] is a tool for visualizing and modifying ontologies, which is provided with Extraction 2.0 as an external component. When Extraction generates a hierarchy, it stores it in a file and executes grOwl to visualize the hierarchy that has just been created.

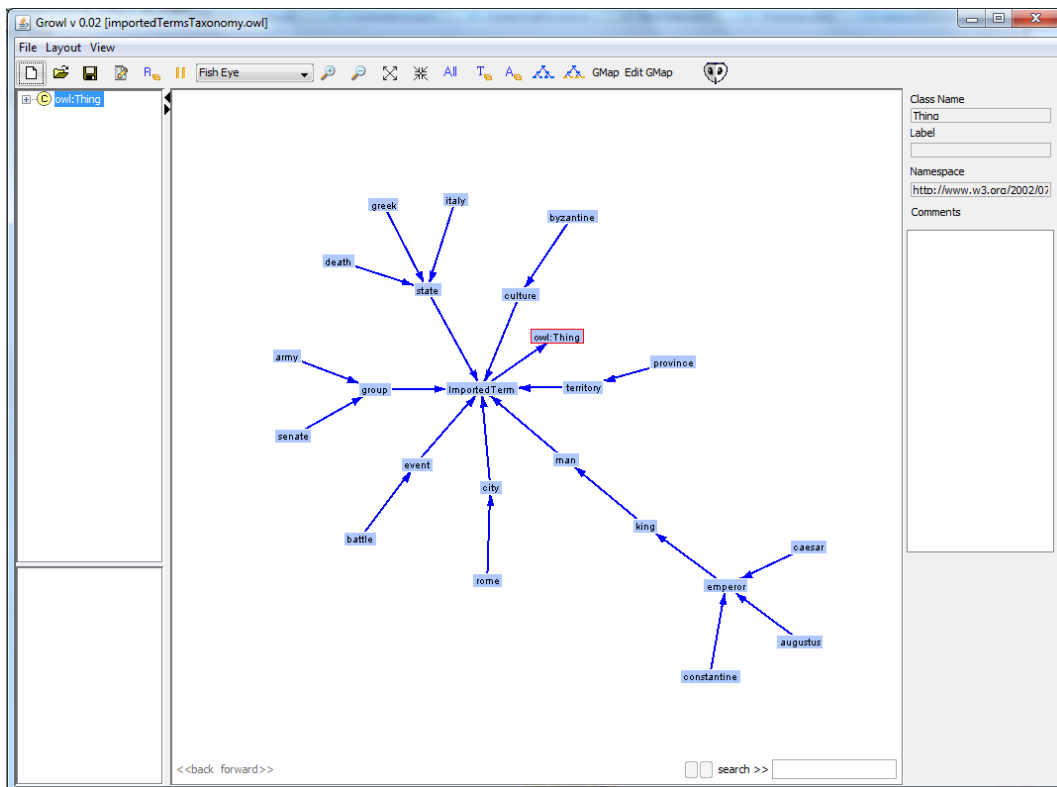


Figure 4.10: Extraction 2.0 - Example of window for the visualization of a hierarchy

#### 4.2.5.3 Indexer

Indexer functionalities were extended to support parsing of:

- PDF files
- HTML pages
- RTF documents
- Microsoft “.doc” ’97-2003 documents

Indexer was also provided with an interface towards an NLP filter, in order to apply to the words in the documents some filters at this level of processing.

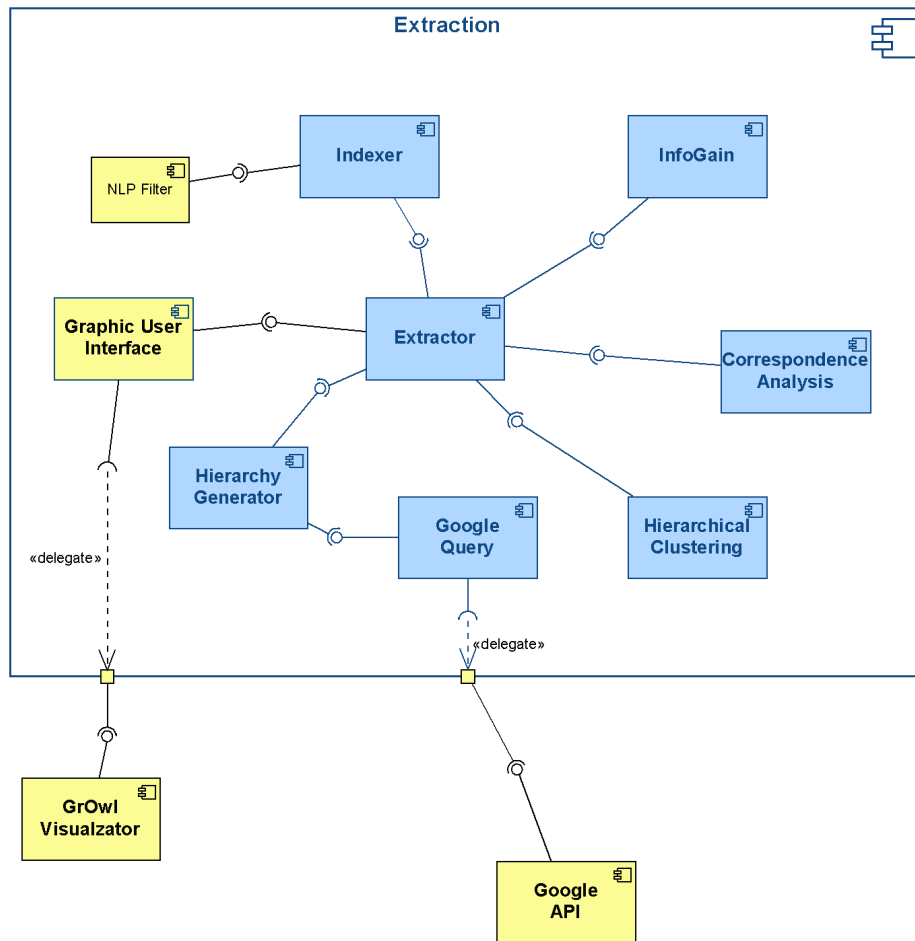


Figure 4.11: Extraction 2.0 - Main modules of the application, component diagram, components in yellow are new in respect to the 1.0 version

#### 4.2.5.4 NLP Filter

This module is a collection of three different filters, each of which can be activated at will by the user, and applied during the indexing process:

- **Pass-noun Filter:** NLP algorithms are applied in order to distinguish nouns looking at the context of the sentence in which they are placed. Every term that is not recognized as a noun is discarded.
- **Number-of-letters Filter:** the words composed by less than  $n$  letters (with  $n$  specifiable by the user) are discarded. This is often useful because words composed by

few letters (as 2 or 3) usually do not express interesting concepts

- **Singularizing Filter:** this filter can be activated only if pass-noun filter is activated. It is able to convert a noun to its singular form, allowing the program to count the occurrences of, for instance, `pencil` and `pencils` as the occurrences of the same concept

It is easily understandable that pass-noun and singularizing filters are strongly language-dependent: Extraction 2.0 supports only English language, but the architecture is built to allow easy extensions for the support of different languages.

#### 4.2.5.5 Google Query

As presented in §4.2.5.6, the Hierarchy Generator can generate queries to look for popularity of regular patterns, in order to infer relations of hyponymy between terms. A single query is done via an HTTP connection, sending an exact search query and waiting for a standard Google result that has, between its information, the number of pages that present the exact string requested. This number is called the number of *Google Hits* of the query.

The Google Query module handles the concurrent execution of these queries, in order to make the most of the Internet connection speed at disposal of the hardware. More information about Google API services can be found at [Google Inc., 2010].

#### 4.2.5.6 Hierarchy Generator

The Hierarchy Generator was extended to offer to the user different possibilities for creating a hierarchy or expanding a pre-existent one. I describe here the 5 different options provided:

##### **Generating a Hierarchy with Fionn Murtagh's Algorithm**

This is the algorithm also provided by Extraction 1.0 and described in §4.2.5.6.

##### **Generating a Hierarchy looking for Hearst Patterns on Web**

This approach is inspired to the application PANKOW and to the general principles of the Learning by Googling approach (§2.1.2.6).

The algorithm starts with an empty hierarchy. The first term  $t_1$  is simply put under the root element. Then, for every new term  $t_n$  to be added, its possibility to be an hyponym of any term  $t_i$  already in the hierarchy is checked as follows:

- 5 different pre-defined strings based on as many Hearst patterns are built, they are:
  - *pluralize*( $t_i$ ) such as  $t_n$

- *pluralize*( $t_i$ ) including  $t_n$
  - *pluralize*( $t_i$ ) especially  $t_n$
  - *pluralize*( $t_i$ ) like  $t_n$
  - $t_n$  is a/an  $t_i$
- 6 Google queries are executed: the 5 generated Hearst patterns plus a research of the single  $t_n$  term, obtaining the number of Google hits for every query
  - The score obtained by every string is defined as the ratio between the number of its Google hits and the number of Google hits of the hyponym searched by itself

$$score_{hp_i} = \frac{googleHits(hp_i)}{googleHits(t_n)}$$

- The total score is obtained as a sum of the different 5 scores

Checked the hypernymy scores of every  $t_i$  in the hierarchy,  $t_n$  is placed as hyponym of the  $t_i$  with the highest score between all the scores that surpass a pre-defined threshold level. The threshold level was empirically identified after many experiments, trying to define a score under which the hyponymy hypothesis is unfounded with certainty.

If no  $t_n$  hypernyms are found, the user (if she had activated the option) is asked for the suggestion of a hypernym. The user can also select to discard the term or to add it in the hierarchy as an hyponym of the root element. If no user interaction option is activated, the algorithm directly puts the  $t_n$  term as an hyponym of the root element.

After that  $t_n$  is placed, all its found siblings are checked in the same manner for an hyponymy relation with  $t_n$ ; if the relation surpasses the threshold, the sibling element is put as a hyponym of  $t_n$ .

### Expanding a Hierarchy with Maedche and Staab's Bootstrapping Process

This module is an instantiation of the model described in §2.1.2.7, defined by Alexander Maedche and Steffen Staab in their article "On Discovering Taxonomic Relations from the Web" [Maedche A., 2003]. According to this model, a concept hierarchy is expanded adding a new  $t_n$  term according to the hypernyms of its  $q$  nearest neighbors in the 2D-plot.  $q$  can be specified by the user. I summarize here the calculus of the score for every  $f$  candidate hypernym (refer to §2.1.2.7 for more information):

The notion of Least Common Superconcept between two concepts  $a$  and  $b$  in a hierarchy is defined as:

$$lcs(a, b) = c \text{ such that } \delta(a, c) + \delta(b, c) + \delta(ROOT, c) \text{ is minimal}$$

where  $\delta(a, b)$  is the distance between  $a$  and  $b$  in terms of the number of edges which need to be traversed. Then the taxonomic similarity  $\sigma$  between two concepts in a hierarchy is defined:

$$\sigma(a, b) = \frac{\delta(ROOT, c) + 1}{\delta(ROOT, c) + \delta(a, c) + \delta(b, c) + 1}$$

where  $c = lcs(a, b)$ .

The  $W(f)$  score for a certain candidate hypernym  $f$  is finally computed as:

$$W(f) = \sum_{h \in H(f)} sim(t_n, h) \cdot \sigma(n, h)$$

where  $t_n$  is the term to be classified and  $H(f)$  is the set of hyponyms of candidate hypernym  $f$  that are also nearest neighbors of  $t_n$  in the 2D-plot.

If no neighbors are found in the pre-existing hierarchy, the user (if he had activated the option) is asked for the suggestion of a hypernym. The user can also select to discard the term or to add it in the hierarchy as an hyponym of the root element. If no user interaction option is activated, the algorithm directly puts the  $t_n$  term as an hyponym of the root element.

#### **Expanding a Hierarchy looking for Hearst patterns on the Web**

This is the same approach presented in 4.2.5.6 (Creating a Hierarchy looking for Hearst Patterns on Web), with just the difference that the algorithm does not start with an empty hierarchy, but with a pre-existing one.

#### **Expanding a Hierarchy with both Bootstrapping and Hearst Patterns**

This approach is a combination of the two previous algorithms seen for expanding a hierarchy. The fact is that the pre-existing hierarchy could be very huge, and a new  $t_n$  term to be added with the Hearst patterns algorithm would generate a very huge amount of connections to the Google servers. This, in fact, depends on the Internet connection speed available, but the execution of a very large number of HTTP queries could be very time consuming.

Thus, the idea is to look for the  $n$  nearest neighbors in the 2D-plot of  $t_n$  in the pre-existing hierarchy, and collect all their ancestors. These ancestors are considered the candidate hypernyms of  $t_n$  and they are checked according to the algorithm seen in the previous section: (Creating a Hierarchy looking for Hearst Patterns on Web).

If no hypernyms are found, the user (if she had activated the option) is asked for the suggestion of a hypernym. The user can also select to discard the term or to add it in the hierarchy as an hyponym of the root element. If no user interaction option is activated, the algorithm directly puts the  $t_n$  term as an hyponym of the root element.

### **4.2.6 Execution Environment and Deployment**

Everything described in §4.1.5 is still valid for this second version of the program.

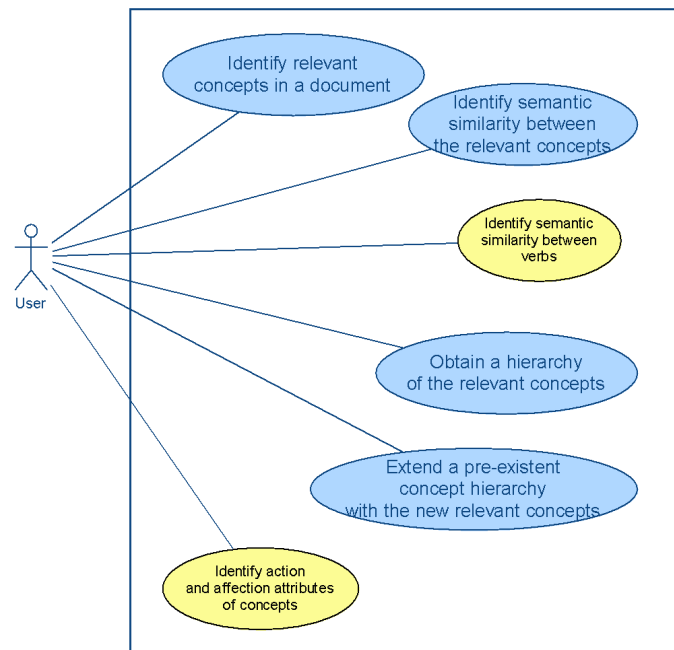


Figure 4.12: Extraction 2.0VV - Main requirements, use case diagram

## 4.3 Extraction 2.0 - Verbal Version

This version of the application is based on the principles presented in §1.6.

### 4.3.1 User Requirements

The requirements that we satisfy in this version are presented in Figure 4.12. As it can be noticed, two new requirements are present:

- **Identify semantic similarity between verbs**
- **Identify action and affection attributes of concepts**

As already presented in §1.6, action and affection attributes are properties of an entity expressing, respectively, what that entity can do, and what can be done with that entity. They can be easily recognized in a sentence being often represented by the verb of

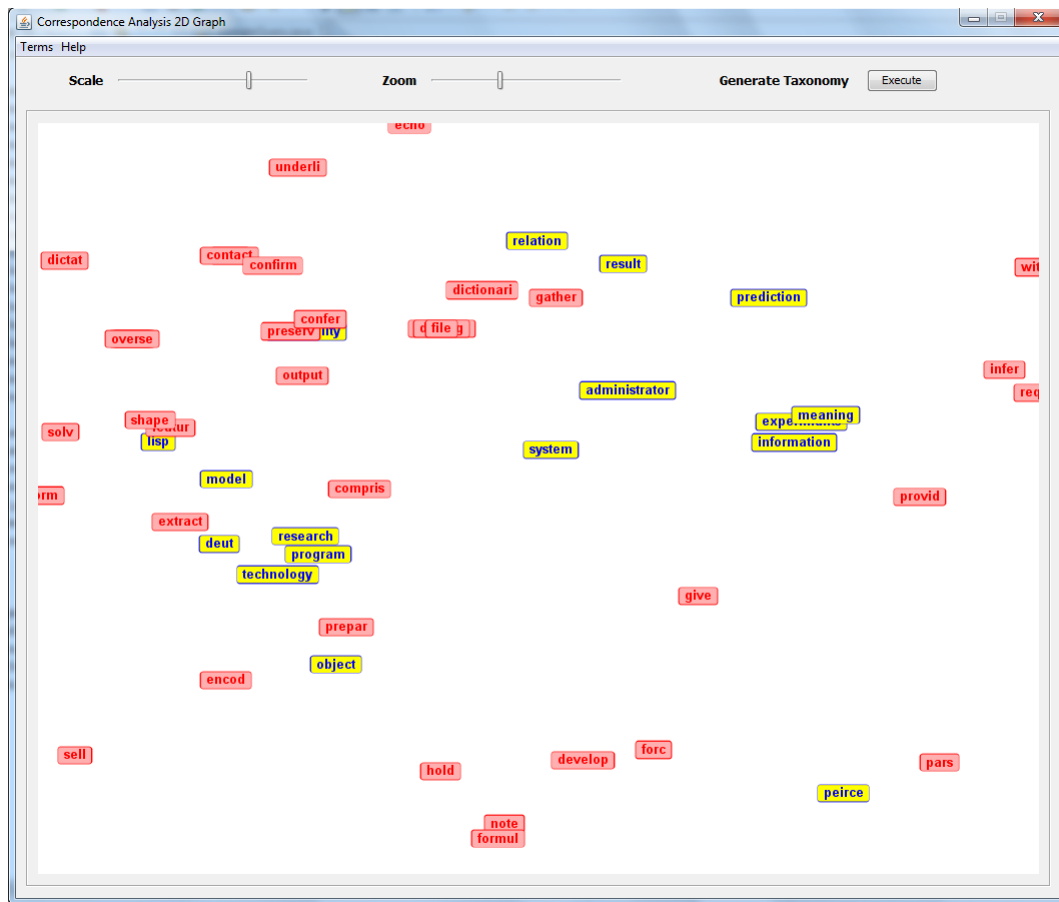


Figure 4.13: Extraction 2.0VV - Example of 2D-plot window, concepts (in blue over yellow) and action attributes (in red over pink) are represented

the sentence of which the entity is subject (for an action attribute) or object (for an affection attribute). Examples of action attributes are *eat*, *play*, *jump*; examples of affection attributes are *eatable*, *playable*, *jumpable*.

### 4.3.2 Specification of the Interface

The GUI specification is the same as the 2.0 version, you can refer to §4.2.2. The only difference here is that the visualization of the 2D-plot has to be specified as extended with the representation of the attributes (Figure 4.13).



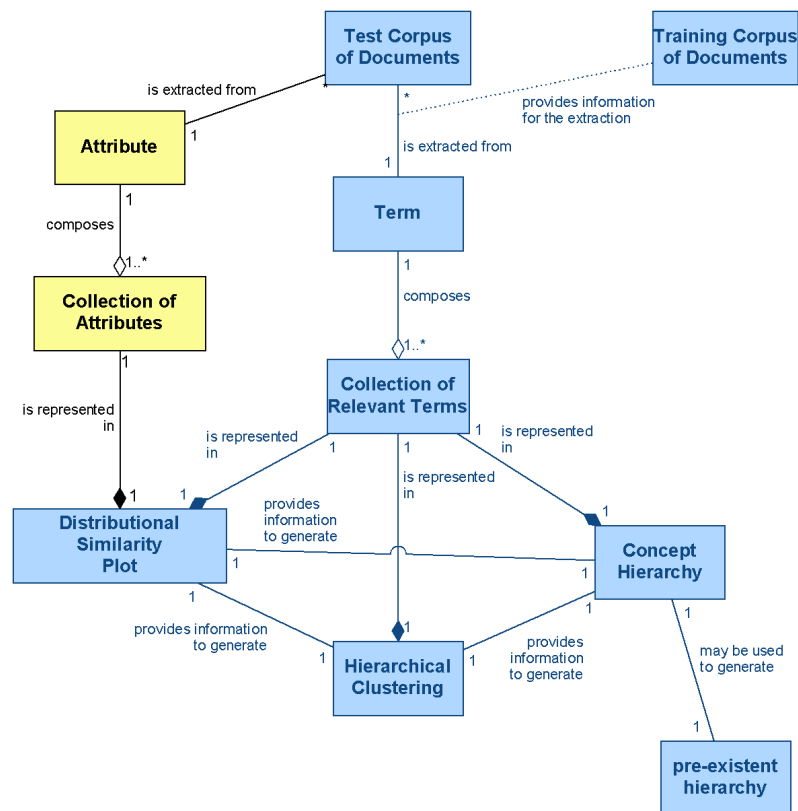


Figure 4.14: Extraction 2.0VV - Main conceptual entities, class diagram

### 4.3.3 Main Conceptual Entities

The main conceptual entities of the application can be seen in Figure 4.14. As it can be noticed, two new entities are added:

- the **Attribute**, of action or affection type, extracted from the test corpus of documents.
- the **Collection** of all the **Attributes** that occur in the test corpus at least 3 times. Attributes with less occurrences are not considered significant and they are discarded

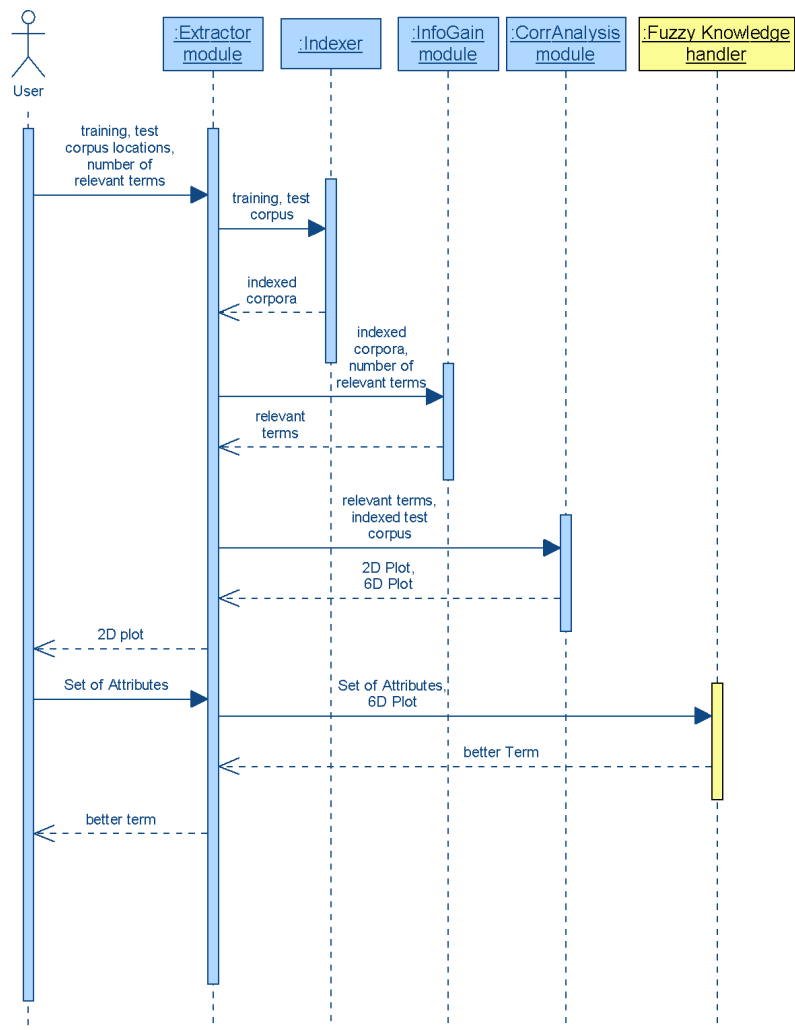


Figure 4.15: Extraction 2.0VV - Interaction scenario of a user providing some attributes to receive back the concept that has the higher probability to be characterized by all those attributes, sequence diagram

#### 4.3.4 Use Cases

The standard user scenario is the same respect to version 2.0 (§4.2.4): a user can obtain a concept hierarchy with all the techniques seen in §4.2.4, starting from the different representation of the terms analyzed in their similarity in respect to their attributes, instead that in respect to their occurrences in specific documents. Furthermore, some other **alternatives** are added to the standard scenario in §4.1.3:

- Instead of creating a hierarchy, the user can textually input a term and an attribute, to print their plot distance. According to what I wrote in §1.6 and §3.2.5, this measure is expected to be in inverse proportion to the attitude of the attribute to characterize the term
- Instead of creating a hierarchy, The user can textually input the name of some attributes, and the tool return the concept that has the higher probability to be characterized by all those attributes (or the 5 better concepts found)
- Instead of creating a hierarchy, The user can textually input a concept, and the 10 nearest action attributes and 10 nearest affection attributes are printed According to what is said in §1.6 and §3.2.5, they are expected to be the attributes that more probably characterize the input concept, according to what is expressed by the test corpus of documents

In Figure 4.15, as an example, an alternative scenario is represented where attributes are specified as an input and the concept more characterized by those attributes is returned, as it is found in the representation.

#### 4.3.5 Main Modules of the Application

As it can be seen in Figure 4.16, the only new component respect to version 2.0 is the Fuzzy Knowledge Handler module. Also, Indexing is applied in a different manner in order to extract both attributes and terms, Correspondence Analysis has a different approach in the generation of the euclidean space, for the same reasons of presence and confrontation of attributes and terms.

##### 4.3.5.1 Indexer

The indexer uses NLP algorithms in order to identify, for every sentence present in a document, the so-called SVO Triplet, a triplet of (subject, verb, object) of the sentence, according to the technique presented by a Romanian university research by different students [Rusu D. et al., 2005]: for every English verbal phrase  $VP$  identified by the algorithm, the previous noun phrase  $NP_s$  and the subsequent noun phrase  $NP_o$  are taken, obtaining the structure:

$$NP_s - VP - NP_o$$

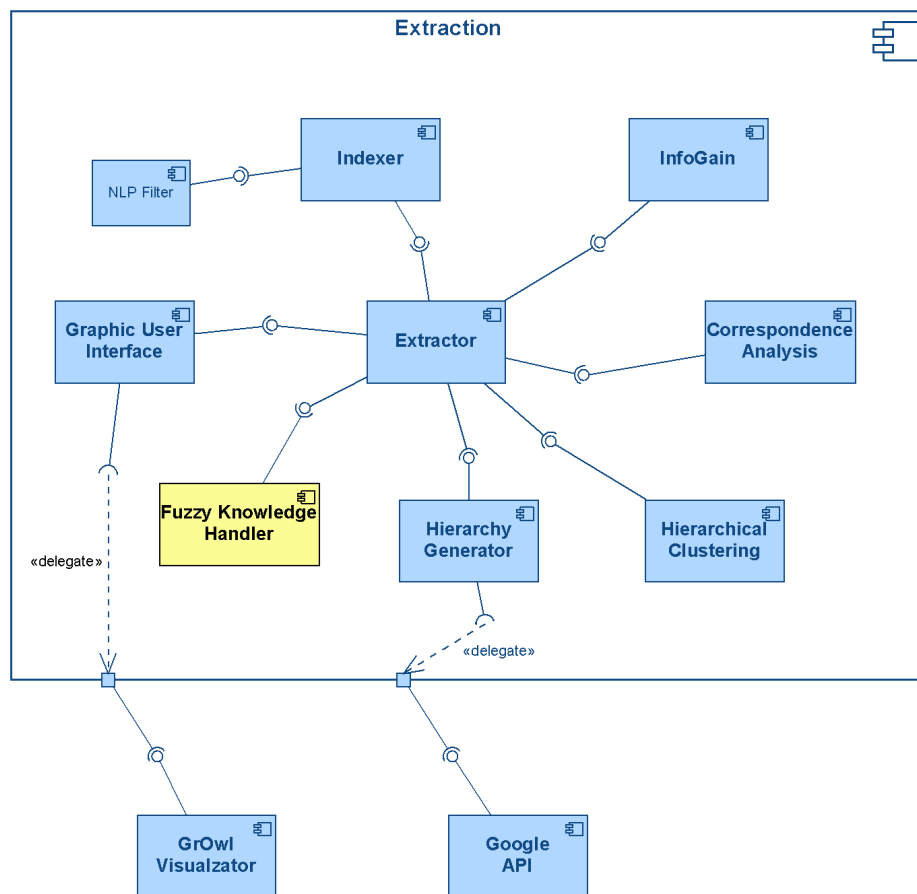


Figure 4.16: Extraction 2.0VV - Main modules of the application, component diagram, components in yellow are different in respect to version 2.0

According to [Rusu D. et al., 2005], the first noun appearing in  $NP_s$  can be considered the subject of the sentence. This choice was made by the Romanian students because they wanted to arrive at a unique solution, they had to extract a single noun, and they decided to take into consideration the first appearing one. In fact, there are many occasions in which the subject of the action expressed by the verb in  $VP$  is not the first noun appearing in  $NP_s$ , but the second, the third or (less probably) more, being my approach a statistical one, I preferred to extract all the nouns in  $NP_s$  and keep them as subject, expecting from the subsequent Correspondence Analysis to give few importance to noises generated by nouns that are not in fact subject of the action expressed by the verbal phrase's verb.

The first verb appearing in the verbal phrase different from "be", "have", "can", "may" and all their inflected versions is the verb of the sentence. A stemming transformation is then applied to the verb, in order to reconduct any inflected form to a common

string.

If, in the way just explained, no verb is found, the sentence is discarded. If the Verbal Phrase, according to some heuristics, appears to be a in a passive form, the `subject` of the sentence is moved to the `object` slot, the `subject` is left empty and the algorithm continues with the next sentence.

If the last two conditions are not verified,  $NP_o$  is taken in consideration and all the nouns appearing in it are stored as `object` in the triplet. The algorithm then continues with the next sentence.

For every SVO triplet, the `verb` is stored as an action attribute of all the terms appearing in `subject`, and then it is stored as an affection attribute of all the terms appearing in `object`.

#### 4.3.5.2 Correspondence Analysis

As presented in §1.6 the Correspondence Analysis is applied to a table of cross-occurrences of attributes referred to determinate nouns, for example:

	march	conquer	devastable	...
caligula	34	120	0	...
city	0	10	120	...
troop	200	160	4	...
...	...	...	...	...

Where in the  $n_{i,j}$  cell is located the number of occurrences of the  $j$ th attribute referred to the  $i$ th noun, according to the extraction procedure explained in §4.3.5.1. The program handles, in fact, the attributes as a couple (`verb`, `type`), where `verb` is a verb root obtained from the the indexer (it is a root because of the stemming procedure, see §4.3.5.1), and `type` is one between action and affection. Both nouns and attributes are then projected in the 2-dimensional euclidean space, according to what is presented in §3.2.5.

Also, as explained in §3.3.3, the confrontation between modalities of different variables (in our case, nouns and attributes), must be handled with a particular care. In §3.2.3.2 we explain the possibility to generate euclidean spaces of more than 2 dimensions; in fact, we are able to generate a space of up to  $\theta$  dimensions, with:

$$\theta = \min(i - 1, j - 1)$$

With  $(i, j)$  the number of rows and columns of the matrix. Being the matrix usually something very huge (with  $i$  and  $j$  of the order of a thousand), we have absolutely no problems of dimensions limits.

After some empirical tests about inertia carried by the plot axes, we decided that it was appropriate to generate also a 6-dimensional euclidean hyperspace where to compare the noun-attribute distances, in order to obtain a better measure, continuing to make

the Graphic User Interface visualize the 2-dimensional plot. So, it can happen that a term and an attribute that are represented not so near in the visualization are then computed as very near, or vice versa: this is due to this augmented precision in the distance computation respect to the visualization.

#### 4.3.5.3 Fuzzy Knowledge Handler

This module handles all the procedures necessary to return results to requests of the user that ask for a fuzzy form of computation. Here the three functionalities of the application that ask something in this sense follow:

##### Noun - Attribute Distance

The user textually inputs a noun and an attribute (specifying `verb`, and `type`), the value returned is their distance in the 6-dimensional space, and it is expected to be the lower the higher the probability is for the attribute to be a characterizing aspect of the noun.

This simple operation is an expression of a strong semantic: the number is in inverse proportion to a value of probability to be given to a proposition, for instance, giving as input:

```
chicken, (eat, affection)
```

We can obtain a number that what a corpus of documents have to say about the probability that a chicken is eatable. Arranging some heuristics, a 0 to 1 probability value could be inferred by the distance metric, making this tool a real generator of fuzzy propositions from corpora of documents.

##### Input of Attributes to Obtain the Most Characterized Concept

The user can input a group of attributes  $a_i$  at will, in the form (`verb`, `type`). The Fuzzy Knowledge Handler module calculates the distances between the  $a_n$  attributes and every  $t_i$  term in the corpus, giving a score to the  $t_i$  term in inverse proportion to the sum of the squared distances of between the  $t_i$  term and every  $a_n$  attribute:

$$score(t_i) = \frac{1}{\sum_n distance(t_i, a_n)^2}$$

The concept with the best score is then returned. Alternatively, the first five concepts can also be seen, with their respective scores.

##### Nearest Attributes of a Concept

The user inputs a concept, and the 10 nearest action attributes and 10 nearest affection attributes are identified in the 6-dimensional plot and returned.

### 4.3.6 Execution Environment and Deployment

The main characteristic that differentiates this version of the program from the previous two is a large usage of NLP algorithms in the indexing process. NLP procedures are very expensive in term of computational resources, so, the best deployment for this application would be a machine with more computational power than an ordinary PC. Some performance tests are presented in §5.10.

## 4.4 Considerations and Conclusions

All the versions of the application are characterized by an approach with a strong statistical basis. About Extraction 1.0 there is to say, anyway, that it presents many limits under different aspects:

- The only parseable documents are the ASCII-8 only-text documents (the “.txt” documents in Windows), no formatted text is allowed. This is a great limit for the application, because many corpora of documents are represented by Web Pages, PDF, but also RTF or other document formats
- No GUI is present, the user communicates his choices textually, and he receives back textual communication via the system standard output. The final hierarchy can be stored in a file, but it has to be opened with an external tool to be viewed. Also, having as an important output a 2-dimensional graph, receiving in output a group of numerical coordinates is not of immediate visualization. The minimum we would expect from a usable tool is to have a visual representation of the graph
- The pass-noun filter, as based on the WordNet archive, does not turn out to be so useful, because we expect from a corpus of documents to extract new and domain-specific concepts, in other words, concepts that are not expected to be already present in a generalized ontology like WordNet. A pass-noun filter should be based on a different approach, trying, for instance, to understand the grammar role of a word with respect to the context of the sentence where it is placed using NLP algorithms
- As you can see in the example in §4.1.4.6, the obtained hierarchy is not conceptually correct. It can be thought that probably a real group of terms, obtained from a real corpus of documents, could return better results, but, in fact, this is not true. The main problem of this approach is that it wants to generate *at any cost* an hyponymy relation for *every* term, not considering the possibility for a term to simply have no hypernyms, as in fact it happens for the great part of the extracted terms. As we expected, and as we obtained during the test procedures (see §5.2), precision results for the hierarchies generated by this algorithm are awful

I primarily worked, with the development of version 2.0, to improve these different functionalities of the tool, in order to provide a more valid and usable service. In version 2.0, in fact, some of the the best hierarchy extraction solutions of our current state of the art have been added to a strong statistical basis. As different results encourage to think (§5.9),

Extraction can represent a good solution to extract concepts from a corpus of documents, to examine semantic similarity between them, and to organize them in standard hierarchies to be manually refined and made available to of large knowledge-based systems.

About the different Euclidean hyperspace represented by Extraction 2.0 VV, there is to say that it could represent a good source of knowledge for systems more complex than ordinary semantic applications, and they should be designed to handle this different type of fuzzy logic. Anyway, if we want to understand something more about the possibilities for this model to be a usable representation of knowledge, we have to evaluate it against four parameters, that are scalability, usability, expandability and validity:

- **Scalability:** it is guaranteed by the Correspondence Analysis approach, that handles at the same manner little and very huge amounts of data
- **Usability:** about usability nothing certain can be said, it should be measured in relation to the system in which the representation would be applied, and hypotheses at this level of development of its possibilities in relation to any type of system would be rash
- **Expandability:** it represents, conceptually, the main problem, because any time new terms/attributes are added to the collection, a new euclidean space should be generated from the whole amount of data. Anyway, a first possibility is to generate a new euclidean space for just the new terms/attributes and then add the space obtained as new dimensions of the old one. However, The possibility for the creation of an algorithm of expansion of the euclidean space with new data is not to be discarded, even if it should be noted that it would definitely request a huge amount of conceptual work
- **Validity:** the few amount of tests presented in §5.7 and §5.8 should return a first idea of the validity of the information carried by our term/attributes euclidean space, precision of propositions is evaluated in about 68% for the identification of nearest attributes of a name



## Chapter 5

# Tests and Results

According to what is presented in §2.1.4, a reliable evaluation of a knowledge representation has to proceed in two distinct dimensions:

- a **Functional Dimension**, where the people who are responsible of the quality assurance have to evaluate the quality and usability of the knowledge representation in respect to the software environment of which it serves the purpose
- a **Structural Dimension**, where the structural validity, coherence and consistency of the representation is checked

**Functional Dimension** is an aspect that, at this actual state, we can not evaluate, being us in relation with no system which actually needs a knowledge repository. We can, however, evaluate the **Structural Dimension** of both ontologies and fuzzy spaces generated with the different algorithms seen in Chapter 4. To perform this evaluation, three different corpora of documents were selected as test corpora:

- A set of 847 Wikipedia articles about **Artificial Intelligence** and related arguments
- A set of 1464 Wikipedia articles about **Roman Empire** and related historical articles
- A set of 1364 Wikipedia articles about **Biology**

As referential Training corpus is always used the same collection of 1414 Wikipedia articles about **Wikipedia** itself. This choice was made according to what stated about InfoGain measure of relevance of terms in §4.1.4.3: we expected from the confrontation between any of the three test corpora and this Wikipedia training corpus that less importance would have been given to terms typical of every Wikipedia article. This, in fact, proved to be correct and terms as, for example “Wikipedia”, “Wikimedia”, or “article”, that have a high importance in all the four corpora of documents in themselves, received a lower InfoGain value respect to other characterizing terms of each test corpus.

§5.1 is devoted to some arbitrary observations made on 2D-plot representations generated by the Correspondence Analysis model of the tool, according to what stated about the information understandable from this type of view in §3.3.

In §5.1-§5.6, according to what is presented in §2.1.4, we present different tests where the precision of ontologies generated by the tool are evaluated by a human judge that selects the valid subsumption relationships according to his knowledge. The precision is computed then as the ratio between the valid subsumption and the totality of the subsumption. Recall measure (it is presented in §2.1.4 as the ratio between valid subsumptions found and all valid subsumption identifiable between all the terms in the hierarchy), was avoided. It would have been a good measure of evaluation, but its acquisition turned out to be too much time-consuming, having a human judge to manually analyze every single couple of terms in an ontology for the possibility of a subsumption relation.

In §5.7 and §5.8 is presented a different approach of evaluation established for the noun/attributes Euclidean space. In §5.9 is a summary view of the different results obtained and some final comments. Finally, §5.10 is devoted to the different evaluation of performance of the tool in terms of time requested for the execution of the algorithms.

## 5.1 Distributional Similarity View Evaluation

### 5.1.1 2D-Plot Concepts Distribution against Documents

As presented in §4.2, the Extraction tool generates a view of the distributional similarity of the concepts extracted from a corpus of documents: we want to evaluate here if this representation give us in fact some early information about semantics of the extracted concepts, according to the lecture dimensions presented in §3.3.

#### Test 1

Research of semantic information in the documents/terms graph - corpus about Artificial Intelligence

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 847 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 200

The term `philosopher` was underlined, and then names of different philosophers and philosophical movements: note that they are all placed in a circumscribed and very specific area of the graph (Figure 5.1, 5.2).

12 terms arbitrarily considered of general interest were identified:

- science
- philosophy
- theory
- computer

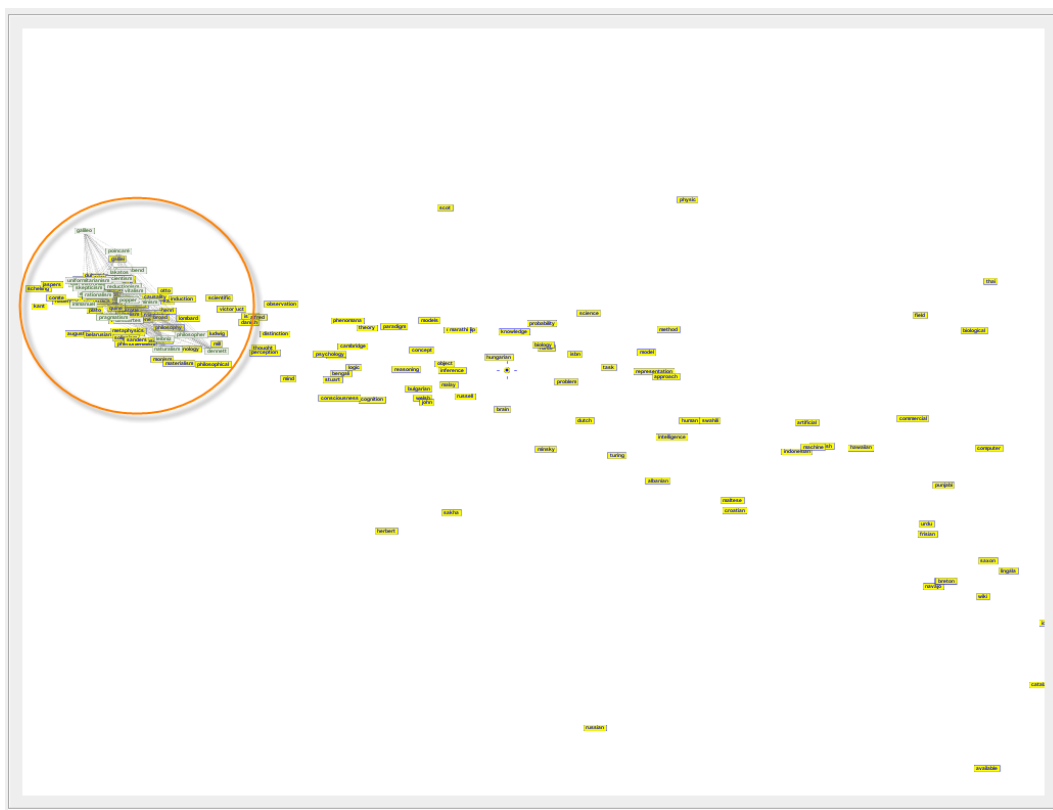


Figure 5.1: Corpus about Artificial Intelligence, 2D-plot - Semantic area of philosophers names and philosophical movements

- intelligence
- mind
- problem
- logic
- machine
- knowledge
- concept
- object
- psychology

It is noteworthy how all these terms (Figure 5.3) are distributed with a high probability near the origin, except for three terms: philosophy, machine, computer. However,

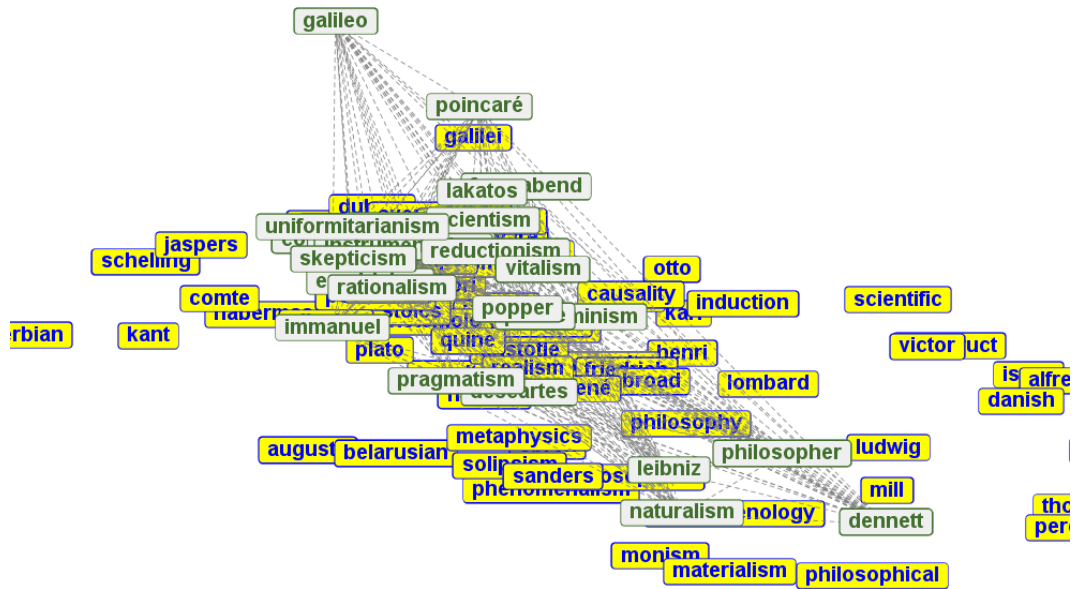


Figure 5.2: Corpus about Artificial Intelligence, 2D-plot - Semantic area of philosophers names and philosophical movements, zoom

they appear central respect to their effective semantic areas: in particular it seems that everything about mathematics and engineering is moved on the right, while philosophical problems are moved on the left, so, we can identify with not so much doubts the main X axis as the influence factor +philosophical/+mathematical.

These are some examples of terms moved on the right:

- turing
- intelligence
- machine
- problem
- science
- approach

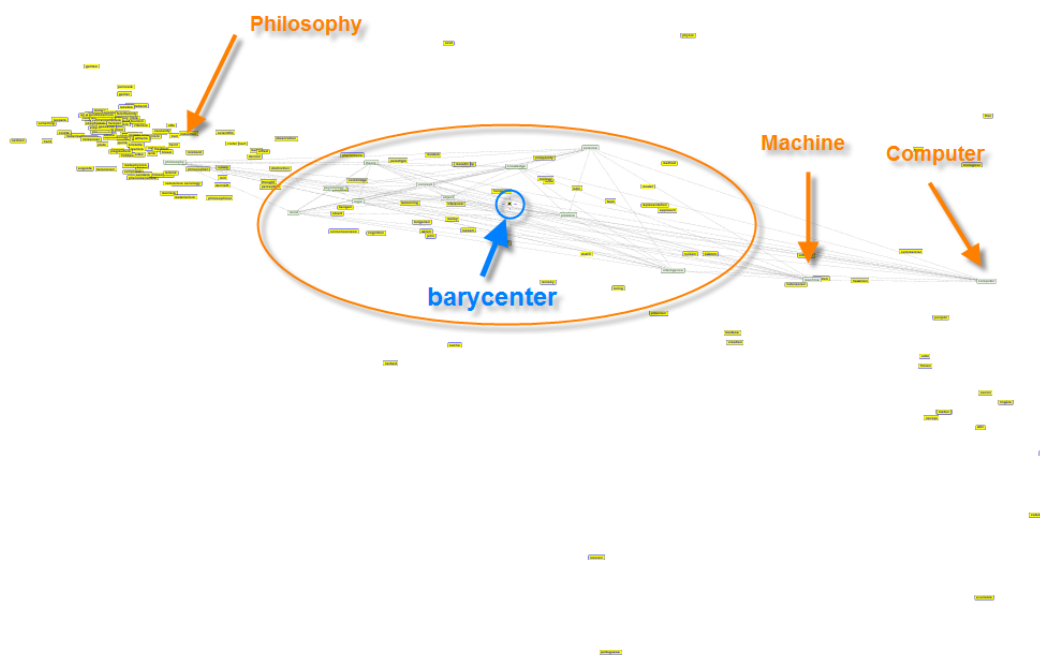


Figure 5.3: Corpus about Artificial Intelligence, 2D-plot - Distribution of terms of more general interest for the corpus, mostly located near the origin

- representation
- problem

How it can be seen from Figure 5.1, the terms are less distributed along the Y axis. Thus, we can hypothesize that this second factor does not represent a significant axis of distribution of the terms.

## Test 2

Research of semantic information in the documents/terms graph - corpus about Roman Empire

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects

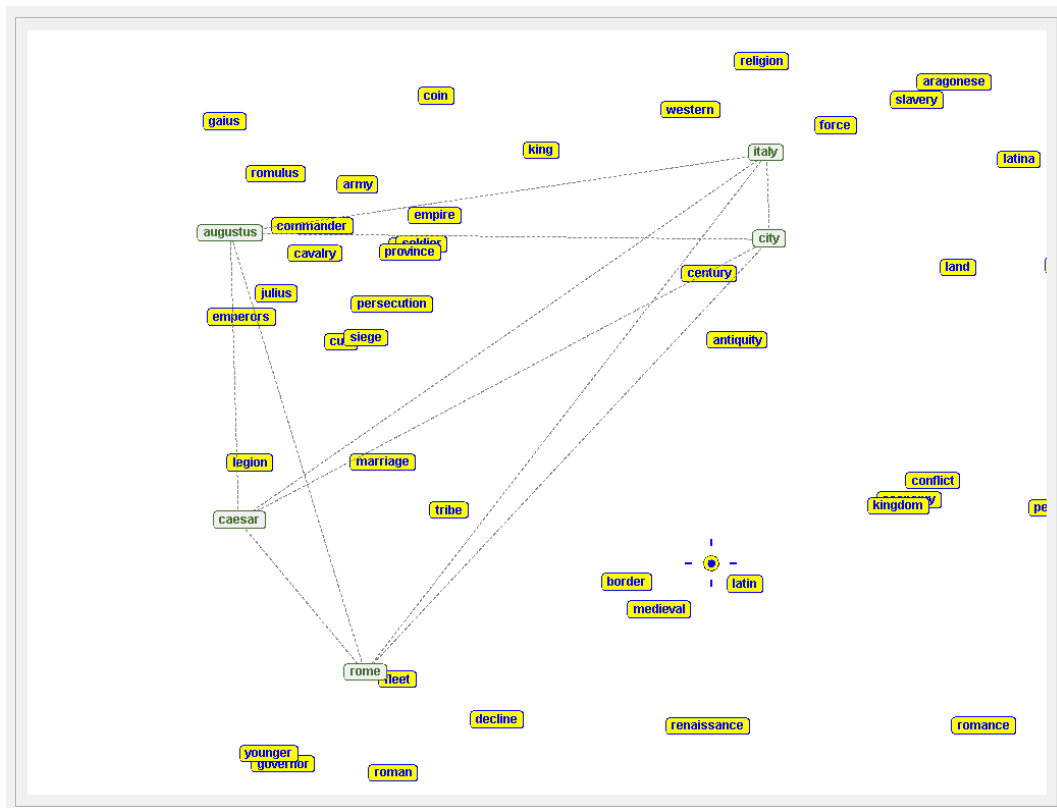


Figure 5.4: Corpus about Roman Empire, 2D-plot - Terms of more general interest

**Total number of acquired terms: 200**

Again the terms that appear to be of general interest were selected:

- augustus
- caesar
- rome
- city
- italy

All of them were found very near to the origin (Figure 5.4). The greatest distribution of the terms appears to be over the Y axis. I tried to find out a semantic motivation to this distribution, but no conjecture seems to be so strong to justify it in itself. Terms at the top seem to refer to the historical period of the empire, while terms at the bottom tend to be

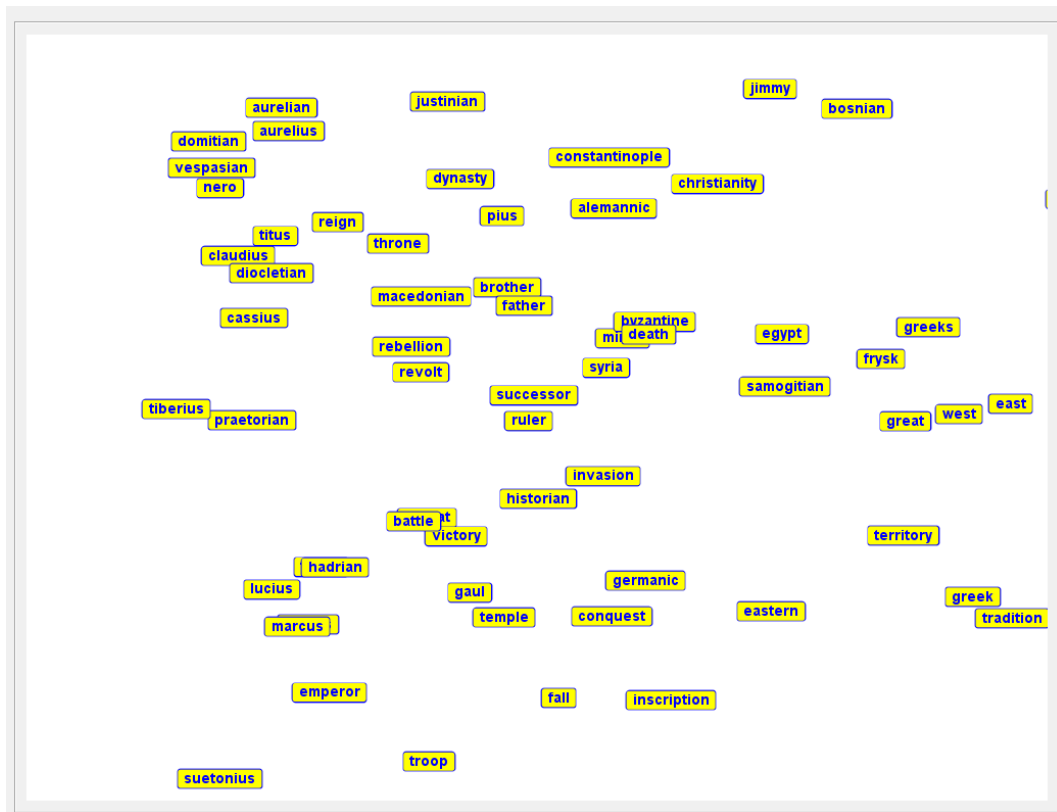


Figure 5.5: Corpus about Roman Empire, 2D-plot - Top area of the graph

the ones that refer to the republican period and to practices and customs of the roman people (Figure 5.5, 5.6).

### Test 3

Research of semantic information in the documents/terms graph - corpus about Biology

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1364 Wikipedia articles about Biology

**Total number of acquired terms:** 300

Selected terms considered of general interest:

- plant
- specie

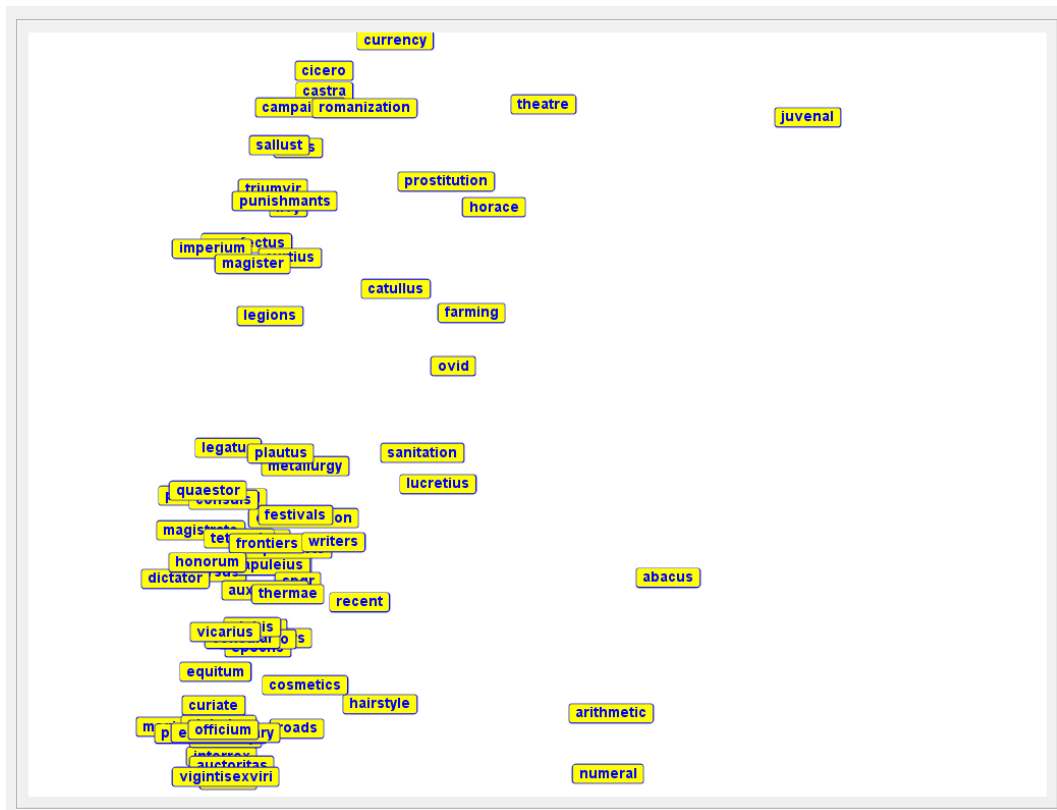


Figure 5.6: Corpus about Roman Empire, 2D-plot - Bottom area of the graph

- animal
- water
- organism
- classification
- biology

All the terms are near the origin but they seem a little far from the center of the cloud of terms, they are all moved to the left side of the representation (Figure 5.7, 5.8).

The center of the cloud is moved to the right respect to the barycenter of the same cloud, and the main terms are moved to the left respect to the barycenter. The position of the main terms can justify the movement of the barycenter to the left, because they have a larger weight and so the barycenter moves according to this. So, main terms have a little cloud of terms around them, instead a large cloud is positioned right away from what was supposed to be the barycenter, let us have a look at them: Figure 5.9, the cloud





Figure 5.7: Corpus about Biology, 2D-plot - Terms of general interest appear moved on the left respect to the origin

is full of terms that have nothing to do with biology, probably the amount of document was contaminated by articles that had nothing to do with biology during the acquisition. It's impressive how this can be easily understood just having a look at this representation instead of reading here and there with no clear intentions between a thousand and more documents.

Having a look at the terms we cannot have a clear idea of what the spurious documents are about. In conclusion, we can label the X axis as +biology/+spurious. Biology's terms are all concentrated in a very small portion of the Y axis, spurious terms on the contrary have a very large distribution. So, as Y axis is a dimension of distribution just for the spurious documents, it is difficult to give an interpretation, as we cannot semantically understand the distribution of terms appearing in scarcely correlated documents (Figure 5.10).

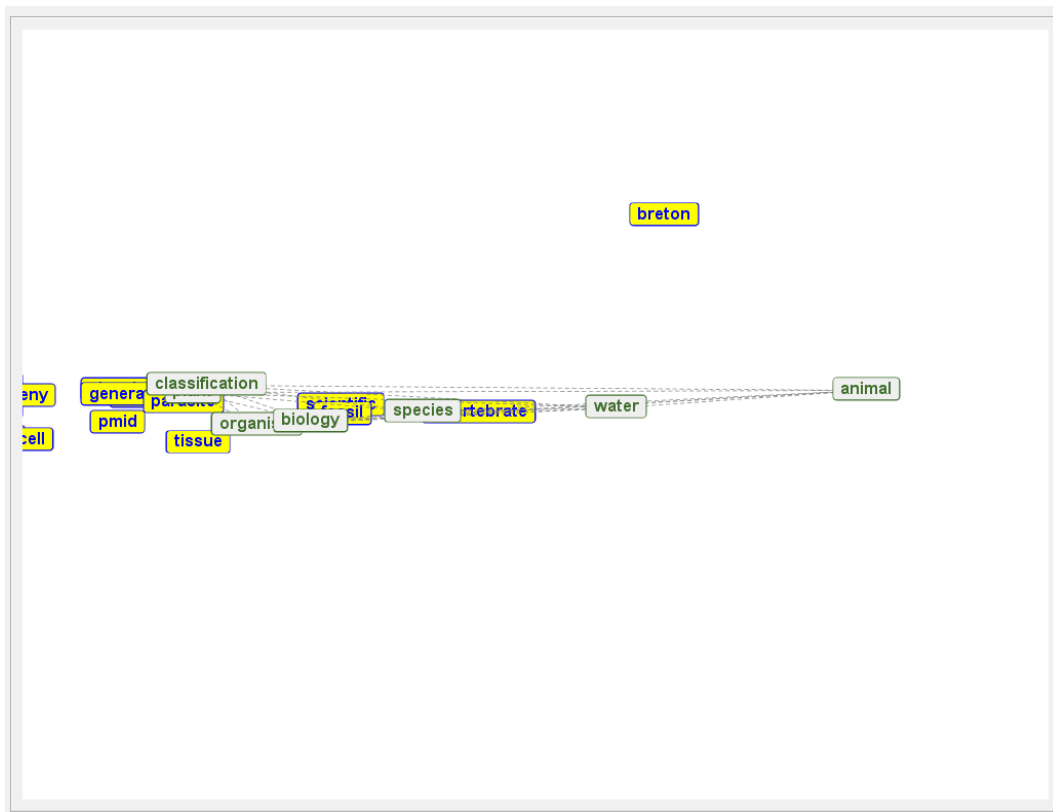


Figure 5.8: Corpus about Biology, 2D-plot - Zoom on terms of general interest

### 5.1.2 2D-Plot Concepts Distribution against Attributes

As presented in §4.3, the Extraction tool also generates a view of the distributional similarity of the concepts created considering their distribution respect to their action and affection attributes. We want to evaluate here if the representation of the concepts in this space gives us some first information about their semantics, according to the interpretation modalities presented in §3.3.

#### Test 4

Research of semantic information in the documents/terms graph - corpus about Artificial Intelligence

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 847 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 150

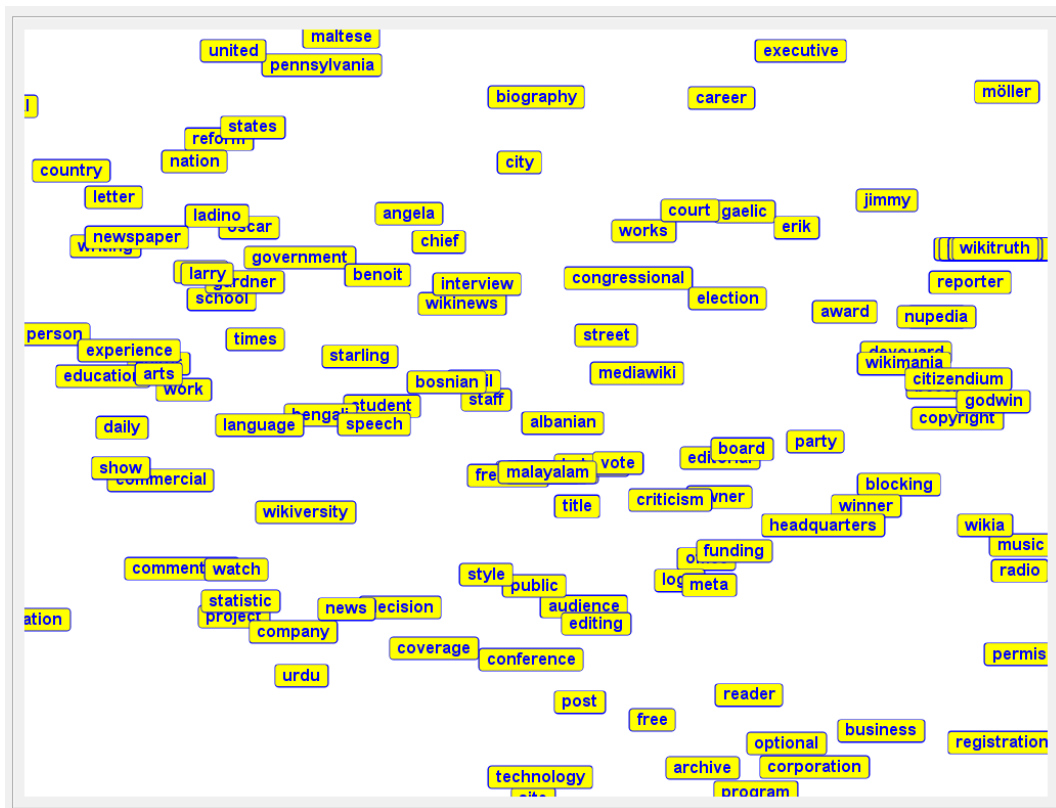


Figure 5.9: Corpus about Biology, 2D-plot - Cloud of many terms on the right of the origin: they have, for the most part, nothing to do with Biology

The same 12 terms selected in Test 1 are considered, be able to compare their new positions in this different distribution:

- science
- philosophy
- theory
- computer
- intelligence
- mind
- problem
- logic

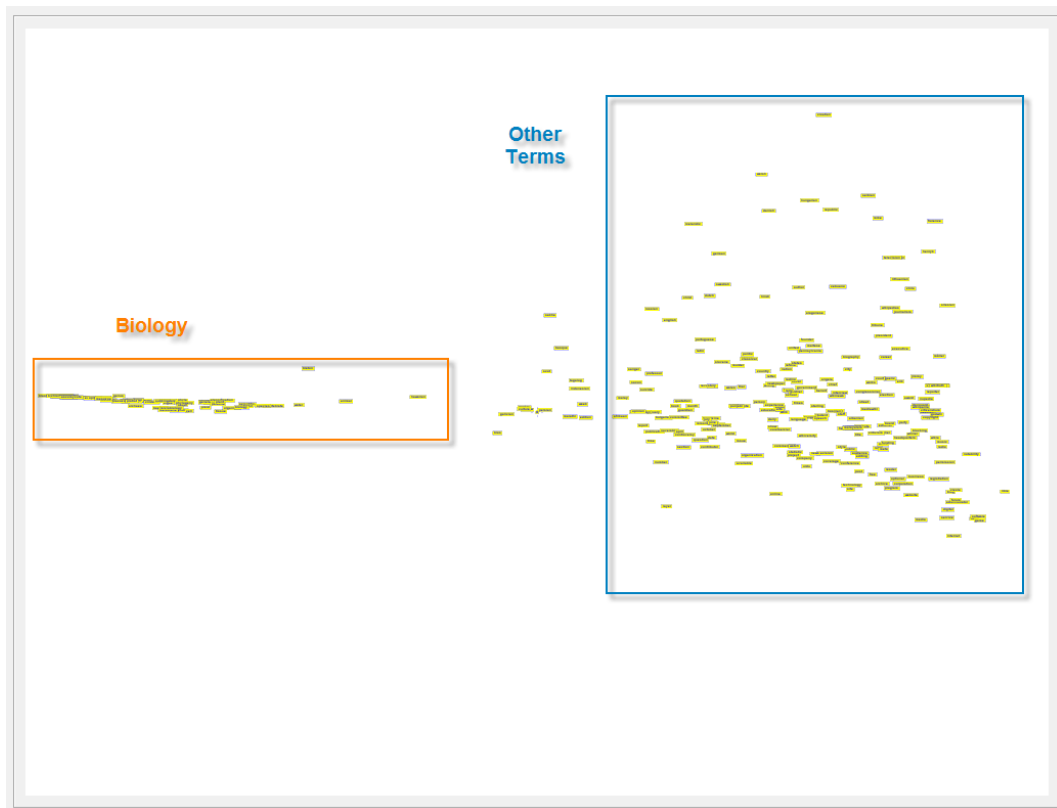


Figure 5.10: Corpus about Biology, 2D-plot - The greatest part of the terms is on the right and has not so much to do with biology

- machine
- knowledge
- concept
- object
- psychology

This representation seems quite more confuse than the one in Test 1. Anyway, the most important concepts are moved on the left side (Figure 5.11), leaving to the center of the representation more conceptually general terms as rule, principle, value, datum, gene (Figure 5.12), but less specific for the corpus of documents.

The axes seem not to distinguish identifiable semantic information. In the graph just some “area of interest” can be identified: they are areas where the terms near one another

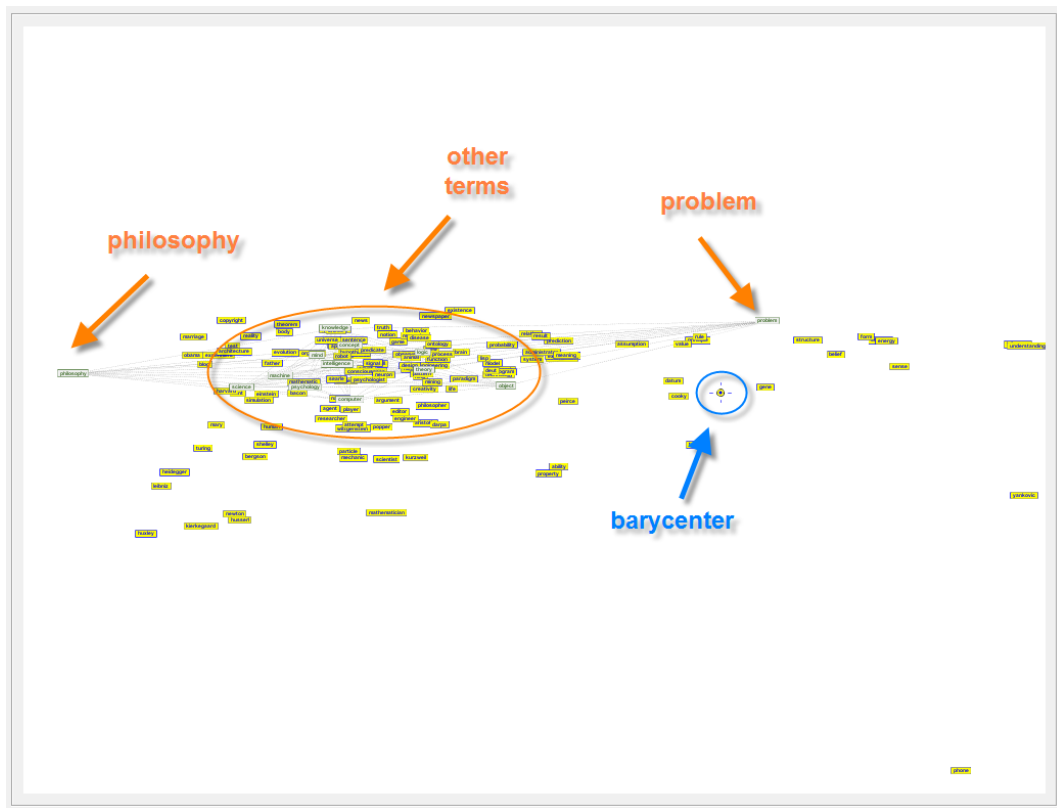


Figure 5.11: Corpus about Artificial Intelligence, 2D-plot - Representation of the terms of more general interest, they seem to be moved on the left respect to the origin

are in fact semantically similar, but a general semantical distribution does not seem to be distinguishable.

### Test 5

Research of semantic information in the terms/attributes graph - corpus about Roman Empire

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects

**Total number of acquired terms:** 150

The terms central in the representation of test 2 were:

- augustus

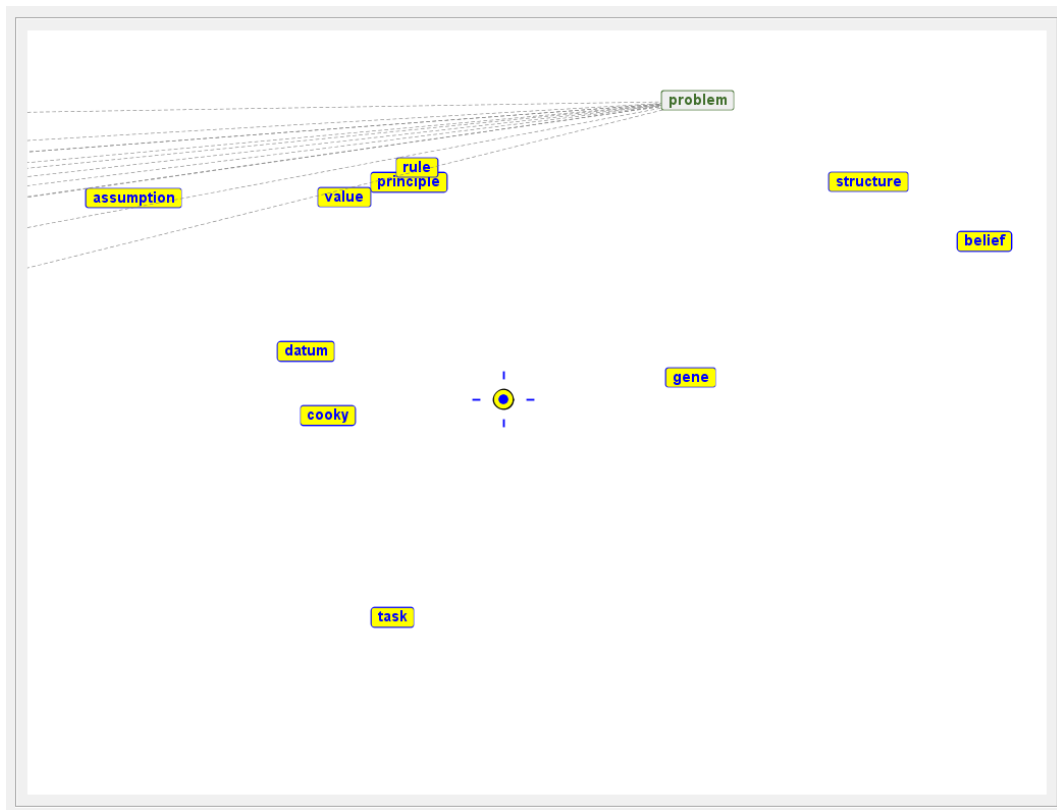


Figure 5.12: Corpus about Artificial Intelligence, 2D-plot - Central view

- caesar
- rome
- city
- italy

Something strange appears, that is difficult to be interpreted: all these terms are near, but far from the origin (Figure 5.13). In fact, *rome*, *city* and *italy* are semantically similar, and so *augustus* and *caesar*. Probably the proximity of these two groups is a consequence of the condision of some attributes that we are, by now, not able to understand.

Figure 5.14 is a zoom on these five terms. It is interesting to notice how in fact *rome* and *italy* are closer to each other respect to *augustus* and *caesar*, and it is interesting also that around these last two names different other names of emperors, generals, leaders are located.



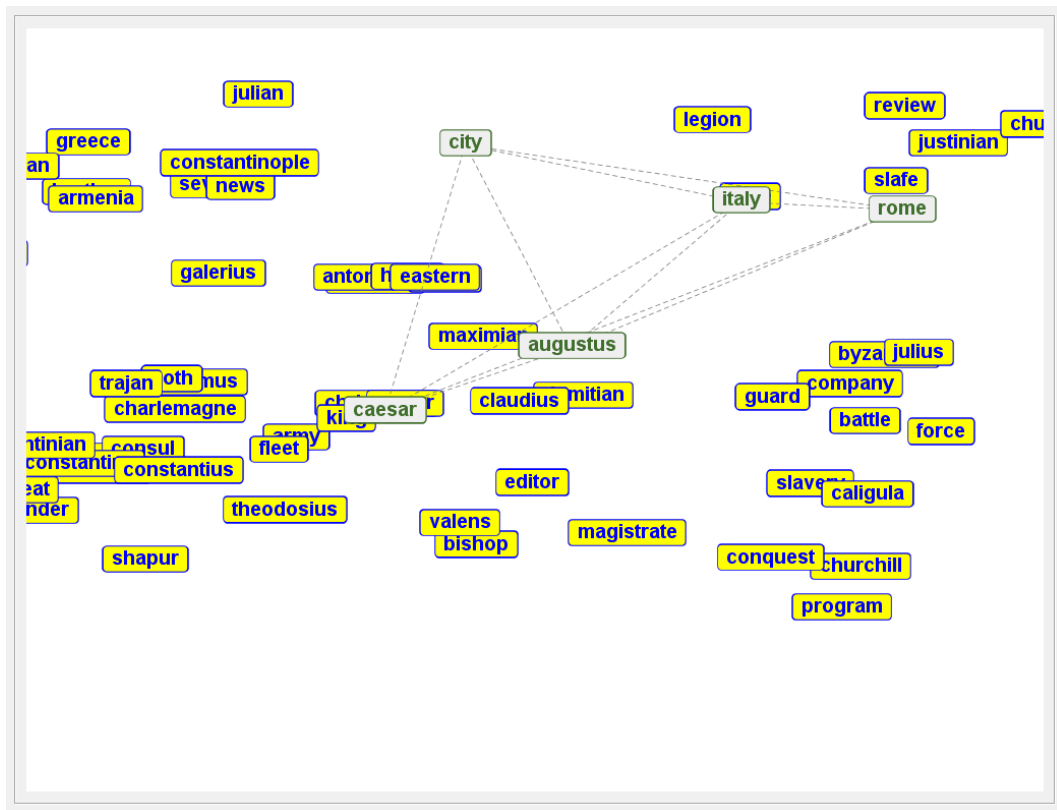


Figure 5.14: Corpus about Roman Empire, 2D-plot - Terms of more general interest, zoom

## 5.2 Fionn Murtagh Hierarchy Generator Evaluation

In this section the precision tests for the algorithm of Murtagh for the extraction of concept hierarchies are documented. the description of this algorithm can be found in §4.1.4.6.

### 5.2.1 Tests

#### Test 6

Concept Hierarchy Generation with Fionn Murtagh's algorithm - corpus about Artificial Intelligence

**Training Corpus:** General document set about Wikipedia, 1414 documents



**Test Corpus:** 847 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 106

Number of subsumptions considered valid by a human judge: 7

Precision ratio: 6.6%

### Test 7

Concept Hierarchy Generation with Fionn Murtagh's algorithm - corpus about Roman Empire

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 115

Number of subsumptions considered valid by a human judge: 6

Precision ratio: 5.22%

An excerpt from the generated hierarchy can be seen in Figure 5.15.

### Test 8

Concept Hierarchy Generation with Fionn Murtagh's algorithm - corpus about Biology

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1364 Wikipedia articles about Roman Empire and historical related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 107

Number of subsumptions considered valid by a human judge: 2

Precision ratio: 1.87%

## 5.2.2 Comments

**Average precision** is **4.56%**. As we expected (see §4.4), this value is very low. Also the little amount of right relations seems to be caught by chance, being formed between near terms in the 2D-plot.

There is to say that Fionn Murtagh in [Murtagh, 2007] proposes some measures to be computed from the 2D-plot in order to verify how much this algorithm is applicable to the terms represented in that plot. Anyway, the measures proposed did not seem to

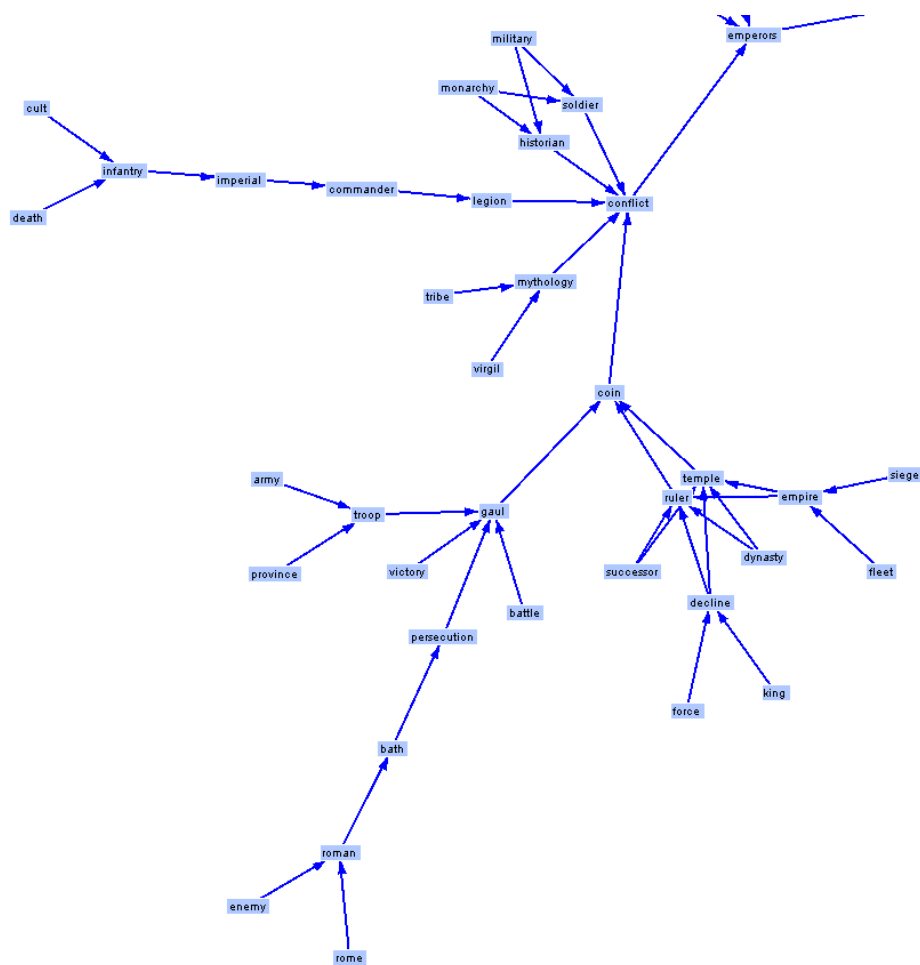


Figure 5.15: An excerpt from a hierarchy created by Fionn Murtagh's algorithm, terms are extracted from a corpus about Roman Empire

return high values for any complex representation of terms. So, either the algorithm was thought for just trivial representation, or there is something that we do not understand. I did not go more in depth, for more information you can directly see [Murtagh, 2007].

### 5.3 Hearst Patterns on Web Hierarchy Generator Evaluation

In this section the precision tests for the algorithm of Hearst Patterns on Web for the extraction of concept hierarchies are reported, the description of this algorithm can be found in §4.2.5.6 (Generating a Hierarchy looking for Hearst Patterns on Web).



**Test Corpus:** 847 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 25

Number of subsumptions considered valid by a human judge: 15

Precision ratio: 60%

An excerpt from the generated hierarchy can be seen in Figure 5.16, the large rose of terms is constituted by the ones put as hyponyms of the root element.

### Test 10

Concept Hierarchy Generation with Hearst Patterns on Web - corpus about Roman Empire

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 8

Number of subsumptions considered valid by a human judge: 6

Precision ratio: 75%

### Test 11

Concept Hierarchy Generation with Hearst Patterns on Web - corpus about Biology

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1364 Wikipedia articles about Biology and historical related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 8

Number of subsumptions considered valid by a human judge: 3

Precision ratio: 37.5%

### Test 12

Concept Hierarchy Generation with Hearst Patterns on Web - corpus about Artificial Intelligence

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 847 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 200

Number of subsumptions created: 30  
Number of subsumptions considered valid by a human judge: 12  
Precision ratio: 45.45%

### Test 13

Concept Hierarchy Generation with Hearst Patterns on Web - corpus about Roman Empire

**Training Corpus:** General document set about Wikipedia, 1414 documents  
**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects  
**Total number of acquired terms:** 200

Number of subsumptions created: 12  
Number of subsumptions considered valid by a human judge: 9  
Precision ratio: 75%

### Test 14

Concept Hierarchy Generation with Hearst Patterns on Web - corpus about Biology

**Training Corpus:** General document set about Wikipedia, 1414 documents  
**Test Corpus:** 1364 Wikipedia articles about Roman Empire and historical related subjects  
**Total number of acquired terms:** 200

Number of subsumptions created: 10  
Number of subsumptions considered valid by a human judge: 5  
Precision ratio: 50%

## 5.3.2 Comments

**Average precision** is 57.08%, it is a good precision, anyway, there is to say that domains of knowledge more specific than ours can be requested for a knowledge-based system, so, less information is expected to be found on the Web about them, and, consequently, this precision measure would reasonably decrease in proportion to the specificity of the domain of knowledge.

Also, as you can see, the number of subsumptions found is very low. Probably a larger number of terms, should be considered, and also the recall measure should be evaluated to see if, in fact, the subsumptions found are the only subsumptions that could have been individuated.

## 5.4 Hearst Patterns on Web Hierarchy Generator Evaluation (Semi-Automatic Acquisition)

In this section the precision tests for the algorithm of Hearst Patterns on Web with Human Interaction for the extraction of concept hierarchies are reported, Hearst Patterns on Web procedure is used, as the previous paragraph. The difference here is that, as described in §4.2.5.6 (Generating a Hierarchy looking for Hearst Patterns on Web), when no hyponymy relation is found the user is asked to suggest one, making of this approach a semi-automatic procedure.

### 5.4.1 Tests

#### Test 15

Concept Hierarchy Generation with Hearst Patterns on Web and semi-automatic procedure - corpus about Artificial Intelligence

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 847 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 50

Number of subsumptions considered valid by a human judge: 45

Precision ratio: 90%

#### Test 16

Concept Hierarchy Generation with Hearst Patterns on Web and semi-automatic procedure - corpus about Roman Empire

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects

**Total number of acquired terms:** 100

Number of subsumptions created: 73

Number of subsumptions considered valid by a human judge: 69

Precision ratio: 94.52%

The generated hierarchy can be seen in Figure 5.17.

#### Test 17

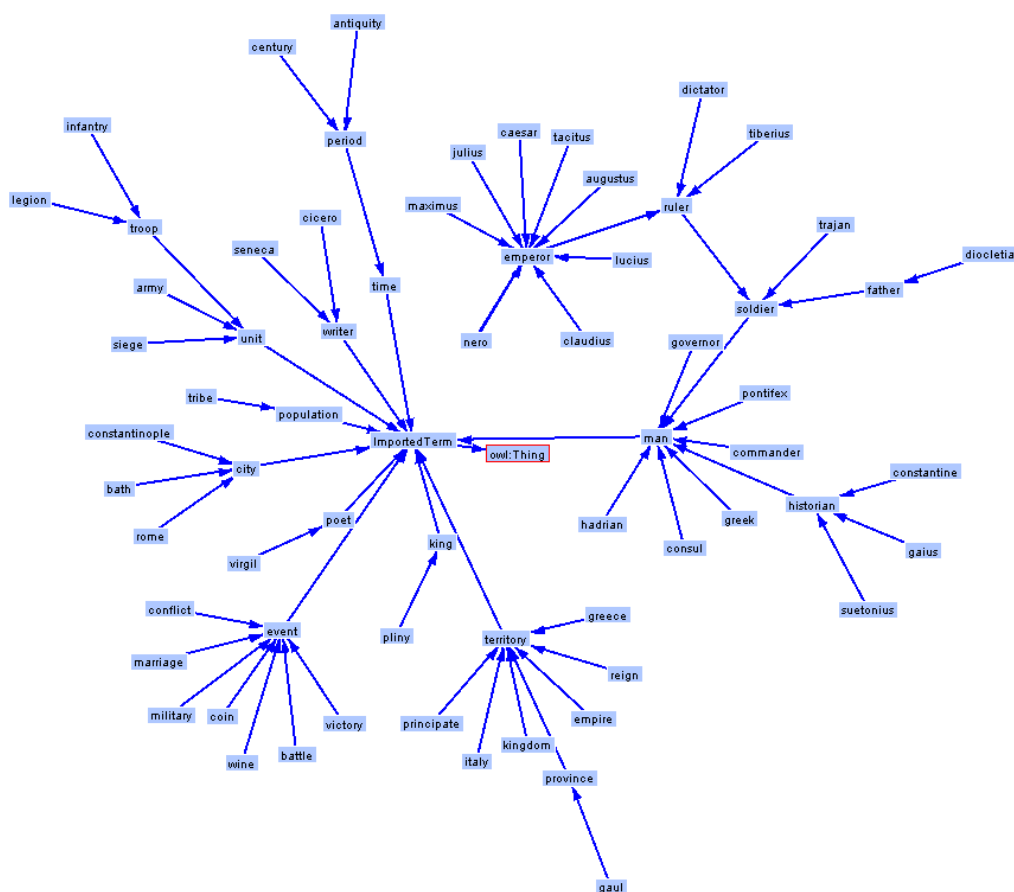


Figure 5.17: A hierarchy created by the Hearst Patterns on Web algorithm in its semi-automatic procedure, from 100 terms of a corpus about Roman Empire

Concept Hierarchy Generation with Hearst Patterns on Web and semi-automatic procedure - corpus about Biology

- Training Corpus:** General document set about Wikipedia, 1414 documents
- Test Corpus:** 1364 Wikipedia articles about Roman Empire and historical related subjects
- Total number of acquired terms:** 100

Number of subsumptions created: 67  
 Number of subsumptions considered valid by a human judge: 57  
 Precision ratio: 85.07%

### 5.4.2 Comments

**Average precision** is 89.86%. It is a very good precision, however, there is to remember that it is obtained with a semi-automatic and time-consuming procedure, interacting with the user for the suggestion of the subsumption relationships.

## 5.5 Bootstrapping Hierarchy Extension Algorithm

Maedche and Staab's Bootstrapping algorithm is described in §4.2.5.6 (Expanding a Hierarchy with Maedche and Staab's Bootstrapping Process). This algorithm was thought to expand large concept hierarchies, however, as we did not have one available, no valid attempts could be done to test this algorithm.

Anyway, tests with pre-existent valid hierarchies of about 70 terms were done and the precision of the algorithm was about 20%, but this measure can not be taken as a valid evaluation.

## 5.6 Bootstrapping plus Hearst Patterns Hierarchy Extension Algorithm

Maedche and Staab's bootstrapping combined with Hearst Patterns algorithm description can be found in §4.2.5.6 (Expanding a Hierarchy with both Bootstrapping and Hearst Patterns). What stated in the previous paragraph is valid also for this algorithm. Early tests show an improvement of 10% respect to Bootstrapping algorithm (we are around 30%), but, still, these measures of precision can not be considered so much referential, while larger hierarchies have not been tested. Anyway, from these approximate observations it can be noticed that the application of the research for Hearst patterns seems to improve the results of the algorithm.

## 5.7 Euclidean Space of Terms and Attributes - 20 Nearest Attributes

In this section the precision tests for the 20 nearest returned attributes of a noun are evaluated, as presented in §4.3.5.3.

In the subsequent tests some concepts have been selected between the ones extracted by the algorithm, and the 10 nearest action attributes and 10 nearest affection attributes have been identified. A concept for the test have been selected as follows: it was extracted randomly from the set of concepts, but then a human evaluated whether the term was an important concept of the domain of knowledge of the corpus of documents. If the term did not pass the test, another random term was selected, and evaluated again.



Three human judges evaluate whether or not the attributes found can represent typical actions that the term can do (for the action attribute) and that can be done with the term (for the affection attribute). The judges express their impressions with an integer number between 0 (impossible to refer the attribute to the concept) and 5 (attribute very typical of the concept), the average of the three votes is reported.

Differently from the evaluation of hierarchies, where the human judge was a member of the development program and aware of the origin of the data and the purposes of the tests, these tests were conducted with a single-blind control procedure, with the human judge unaware of the origin of the data and unaware of the purposes of the test: the judge is just informed of the domain of knowledge of origin of names and attributes (necessary, for the best understanding of the propositions).

### 5.7.1 Tests

#### Test 18

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1464 Wikipedia articles about Roman Empire and historical related subjects

**Total number of acquired terms:** 200

**Total number of identified attributes:** 2288

- Randomly selected term: "city". Nearest action attributes:

Attribute	Human Judgment	Distance from "city"
acquire	4.3	0.02124
continue	5	0.03439
saint	0	0.03633
retell	0.3	0.03798
taunt	1.3	0.03947
oper	0	0.04082
harass	2.7	0.04187
attract	5	0.04220
become	5	0.04422
resist	5	0.04552

**Score:** 28.60/50, 57.2%

- Randomly selected term: "city". Nearest affection attributes:

Attribute	Human Judgment	Distance from "city"
agreeable	0	0.0277
revivable	4.3	0.0318
controllable	5	0.0338
regardable	3.7	0.0367
callable	5	0.0409
developable	4	0.0413
remain(?)	0	0.0414
definable	4.6	0.0429
disturbable	4.3	0.0438
perceivable	4.3	0.0438

Score: 35.2/50, 70.4%

- Randomly selected term: "attila". Nearest action attributes:

Attribute	Human Judgment	Distance from "attila"
elect	4.7	0.0189
demand	5	0.0195
approve	4.7	0.025
subdue	5	0.0268
head	5	0.0281
blend	3	0.0286
further	4	0.0286
contemplate	4.3	0.0286
create	3.6	0.0288
institute	4.3	0.0311

Score: 43.6/50, 87.2%

- Randomly selected term: "attila". Nearest affection attributes:

Attribute	Human Judgment	Distance from "attila"
joinable	4.7	0.0157
meetable	5	0.0177
recallable	4	0.0209
sendable	3.3	0.0258
precedable	4.3	0.0264
honourable	5	0.0286
renderable	3.7	0.0286
aros(?)	0	0.0286
relinquishable	5	0.0286
defeatable	5	0.03

Score: 40/50, 80%

- Randomly selected term: "senate". Nearest action attributes:

Attribute	Human Judgment	Distance from "senate"
summon	4.7	0.018
wall	0	0.018
resign	4.3	0.0183
reverse	4	0.0203
despatch	4.3	0.0208
institute	5	0.0236
pass	5	0.024
embrace	4	0.0241
instruct	3.3	0.0241
neglect	5	0.0241

Score: 39.6/50, 79.2%

- Randomly selected term: "senate". Nearest affection attributes:

Attribute	Human Judgment	Distance from "senate"
awaitable	4.7	0.0176
selectable	3	0.0199
emphasizable	2.3	0.0241
extollable	0	0.0241
overthrowable	4.7	0.0245
signalable	4	0.0248
nominable	5	0.0249
hateable	4.7	0.0284
acclaimable	5	0.0291
compellable	5	0.0329

Score: 38.4/50, 76.8%

- Randomly selected term: "troop". Nearest action attributes:

Attribute	Human Judgment	Distance from "troop"
prefer	3.7	0.0349
overwhelm	2.7	0.0381
believe	3	0.0422
contrast	5	0.046
endorse	5	0.046
guide	3.7	0.046
dampen	3.3	0.0512
hail	4.3	0.0538
study	1	0.0542
deem	2.7	0.0573

Score: 39.5/50, 79%

- Randomly selected term: "troop". Nearest affection attributes:

Attribute	Human Judgment	Distance from "troop"
exhortable	4.7	0.0381
necessitable	4	0.0381
stretchable	4	0.0381
targetable	4.7	0.046
intensifyable	1.7	0.046
occupable	0	0.046
confessable	0	0.046
instructable	4.3	0.046
workable	1.3	0.0505
referrable	4.7	0.0562

Score: 29.4/50, 58.8%

### Test 19

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 867 Wikipedia articles about Artificial Intelligence and related subjects

**Total number of acquired terms:** 150

**Total number of identified attributes:** 1682

- Randomly selected term: "computer". Nearest action attributes:

Attribute	Human Judgment	Distance from "computer"
assess	4	0,0452
steer	2	0,0506
screen	4,7	0,0506
define	4	0,0506
signal	5	0,0506
compare	4	0,0552
prove	3,7	0,0558
play	5	0,0573
meet	2,7	0,0581

Score: 35.1/50, 70.2%

- Randomly selected term: "computer". Nearest affection attributes:

Attribute	Human Judgment	Distance from "computer"
signalable	3,3	0,0506
leavable	3,7	0,0678
stoppable	5	0,0772
editable	4,7	0,0797
collectable	2,3	0,0983
assistable	4	0,0985
programmable	5	0,0992
fundable	3,7	0,101
travellable	0,3	0,1116
preservable	2,7	0,1121

Score: 34.7/50, 69.4%

- Randomly selected term: "chess". Nearest action attributes:

Attribute	Human Judgment	Distance from "chess"
learn	2	0,1005
automate	3	0,1204
achieve	3,7	0,1215
sequence	0	0,1291
inaugurate	3,3	0,1291
spark	1,7	0,1291
ignore	0,7	0,1366
design	0,7	0,1454
trigger	3,7	0,1508
warp	2,7	0,1674

Score: 21.5/50, 42%

- Randomly selected term: "chess". Nearest affection attributes:

Attribute	Human Judgment	Distance from "chess"
reinforcable	2,3	0,1091
undertakeable	1,7	0,1291
hearable	1	0,1291
playable	5	0,151
conceptable	0	0,1525
seemable	4,7	0,1577
distinguishable	3,3	0,1635
declarable	4	0,1653
referencable	4,3	0,1733
promisable	0,3	0,1849

Score: 26.6/50, 53.2%

- Randomly selected term: "intelligence". Nearest action attributes:

Attribute	Human Judgment	Distance from "intelligence"
comprise	4,3	0,0415
browse	5	0,0481
diversify	3,7	0,0663
exhibit	3,7	0,0727
enable	4,3	0,0784
train	4,6	0,0827
remove	4	0,083
reproduce	3,3	0,0945
seem	3	0,1015
specify	3	0,1027

Score: 38.9/50, 77.8%

- Randomly selected term: "intelligence". Nearest affection attributes:

Attribute	Human Judgment	Distance from "intelligence"
protectable	4	0,0494
synthesizable	4,7	0,0522
reconstructable	4,3	0,0542
spinnable	2	0,0639
sizeable	0	0,0705
helpable	5	0,0721
completable	3	0,0726
attributable	2,3	0,0741
adoptable	4	0,08
loadable	5	0,0804

Score: 34.3/50, 68.6%

- Randomly selected term: "newton". Nearest action attributes:

Attribute	Human Judgment	Distance from "newton"
favour	4	0,074
sponsor	2,3	0,0848
voic	0	0,0922
authorize	3,3	0,0922
expand	3	0,101
lend	3,3	0,1123
leave	4	0,1149
theorize	5	0,1158
live	5	0,1224
offer	4,3	0,126

Score: 34.2/50, 68.4%

- Randomly selected term: "newton". Nearest affection attributes:

Attribute	Human Judgment	Distance from "newton"
bearable	4	0,1773
echoable	1	0,2019
notifyable	4,3	0,235
attractable	4,7	0,2418
creditable	5	0,2427
replenishable	3,7	0,2526
suspectable	3,3	0,2623
killable	4,3	0,2798
provokable	4,7	0,2818
meetable	5	0,2846

Score: 40/50, 80%

### Test 20

**Training Corpus:** General document set about Wikipedia, 1414 documents

**Test Corpus:** 1364 Wikipedia articles about Biology

**Total number of acquired terms:** 150

**Total number of identified attributes:** 1682

- Randomly selected term: "fish". Nearest action attributes:

Attribute	Human Judgment	Distance from "fish"
advance	4,7	0,0969
modify	4	0,0981
extract	4	0,1525
group	5	0,1592
embryo	0	0,162
trace	2,7	0,162
tend	4,3	0,1739
inherit	5	0,1806
encounter	5	0,1983
catch	5	0,1991

Score: 39.7/50, 79.4%

- Randomly selected term: "fish". Nearest affection attributes:

Attribute	Human Judgment	Distance from "fish"
appointable	2,7	0,1117
defendable	2	0,1221
spreadable	4	0,1527
eliminable	5	0,155
livable	1,3	0,1595
presumable	0,7	0,162
constitutable	2,7	0,1775
devastable	3,7	0,1776
benefitable	3,3	0,182
filamentable	0	0,1928

Score: 25.4/50, 50.8%

- Randomly selected term: "bird". Nearest action attributes:

Attribute	Human Judgment	Distance from "bird"
tend	5	0,035
encount	5	0,0796
advance	4,7	0,1055
detect	5	0,112
embryo	0	0,1129
trace	4	0,1129
relax	4,3	0,1269
pump	1,3	0,1269
inherit	4,7	0,1308
derive	3,7	0,1497

Score: 37.7/50, 75.4%

- Randomly selected term: "bird". Nearest affection attributes:

Attribute	Human Judgment	Distance from "bird"
harmable	5	0,0542
benefitable	4,7	0,0624
spreadable	4,7	0,0782
filamentable	0	0,0862
presumable	1,7	0,1129
findable	3,7	0,1471
breakable	4,7	0,1572
considerable	5	0,1587
constitutable	4	0,1797
livable	0	0,1822

Score: 33.5/50, 67%

- Randomly selected term: "extinction". Nearest action attributes:



Attribute	Human Judgment	Distance from "extinction"
decimate	5	0,1593
devast	4,3	0,1593
decrease	5	0,1842
encapsulate	2,3	0,1896
emulate	3,7	0,1896
limit	4,3	0,2085
affect	4	0,2156
preserve	2	0,2171
drive	1	0,218
retain	3,3	0,2188

Score: 34.9/50, 69.8%

- Randomly selected term: "extinction". Nearest affection attributes:

Attribute	Human Judgment	Distance from "extinction"
survivable	1,3	0,0974
controllable	4	0,1655
releasable	3	0,1864
blockable	5	0,1896
giveable	2,7	0,1905
causable	5	0,1952
influencable	4,7	0,1959
leavable	3,7	0,201
extendable	4,3	0,2139
displayable	2,7	0,2242

Score: 36.4/50, 72.8%

- Randomly selected term: "protein". Nearest action attributes:

Attribute	Human Judgment	Distance from "protein"
construct	4	0,0648
reproduce	5	0,0695
exclude	3,3	0,0892
characterize	5	0,091
occur	4	0,1228
attain	3,7	0,1349
estimate	1,3	0,1349
copy	4,3	0,1349
sustain	3,3	0,1349
undergo	4	0,1349

Score: 37.9/50, 75.8%

- Randomly selected term: "protein". Nearest affection attributes:

Attribute	Human Judgment	Distance from "protein"
gainable	4	0,0238
synthesizable	5	0,0465
restrictable	2,7	0,0479
fusable	3,3	0,0614
evolvable	4	0,0671
restorable	4,6	0,0768
comparable	4,3	0,0776
supplyable	5	0,0819
cookable	0,3	0,0835
derivable	4	0,0865

Score: 37.2/50, 74.4%

### 5.7.2 Comments

We summarize here the different results for the evaluated concepts:

Concept	Action attributes score	Affection attributes score
city	70.4%	57.2%
attila	87.2%	80.0%
senate	79.2%	76.8%
troop	79.0%	58.8%
<b>Roman Empire average</b>	<b>78.95%</b>	<b>68.2%</b>
computer	70.2%	69.4%
chess	42.0%	53.2%
intelligence	68.6%	61.8%
newton	68.4%	80.0%
<b>IA average</b>	<b>62.3%</b>	<b>66.1%</b>
fish	79.4%	50.8%
bird	75.4%	67.0%
extinction	69.8%	72.8%
protein	75.8%	74.4%
<b>Biology average</b>	<b>75.1%</b>	<b>66.25%</b>
<b>Attribute type average</b>	<b>72.12%</b>	<b>66.85%</b>

**Total average: 69.485%**

These seem to be good results: for affection attributes they are slightly lower, however, we can state that the knowledge representation generated by the tool is able to recognize action and affection attributes of concepts with a confidence of about  $\frac{2}{3}$ . Anyway, having no other referential works, we can not state that these are good results in respect to other approaches, techniques. . . We can just say that they *seem* good, in the sense that they appear usable in respect to a hypothetical systems that could want to use this model as a repository of knowledge, but, in fact, we can not even state this with certainty.

## 5.8 Euclidean Space of Terms and Attributes - Best Concept for a Group of Attitudes

Some tests for the fuzzy knowledge representation were made also with the functionality presented in §4.3.5.3 (Input of Attributes to Obtain the Most Characterized Concept).

In fact the feature can be represented as a sort of a game: the user covers the role of an entity, and provides the tool some information about what this entity can do (action), or what can be done with it (affection). The tool, just based on what it extracted from the corpus of documents, tries to guess the entity. Let us see some example conversations:

Corpus about Roman Empire:

User:

I can persecute  
I can burn, who am I?

Tool:

You are a search

5 better concepts found:

search, score: 15.0910945363155  
student, score: 3.160172198933044  
soldier, score: 1.6199649623492582  
slafe, score: 1.123169417677151  
michael, score: 0.993288794997991

User:

I can persecute  
I can burn  
I can conquer  
I can command  
I can ead  
I can decide  
I can kill, who am I?

Tool:

You are pompey

pompey, score: 5.097153490759302  
vespasian, score: 2.548333865285382  
valentinian, score: 2.4307313967123556  
caesar, score: 2.115809184444728  
sassanid, score: 1.839062421227323

**Note:** Pompey is a Roman general, do not mistake him for the Italian city

Corpus about Artificial Intelligence:

User:  
I can think  
I can evolve  
I can play  
I can grow  
I can infer, who am I?

Tool:  
You are a neuron

5 better concepts found:  
neuron, score: 9.070922751723742  
intelligence, score: 2.7318488636916243  
predicate, score: 1.909471310898176  
mechanic, score: 1.470693690893001  
process, score: 1.469673091701366

#### Corpus about Biology:

User:  
I can be killed  
I can be observed  
I can be fought  
I can be born  
I can be fed, who am I?

Tool:  
you are a sponge

5 better concepts found:  
sponge, score: 8.238901230171537  
tree, score: 2.898485009974354  
fish, score: 1.874004072506257  
soil, score: 1.742540336826821  
clade, score: 1.4611176061108737

I show some between the results both good and bad. Tests were made with different numbers of action and affection attributes, there is to say that these attributes are not chosen at random, but they are selected by a human judge that knows that probably the corpus can speak about something that is characterized by that attributes. In fact, we are looking for something using a sense, random selection of attributes would have no sense at all, and so its results.

In 30 conversations, providing 4-5-6 concepts, the algorithm shows about the 35% of times the first word as an acceptable concept for the attributes indicated, and about the 60% of times at least one acceptable concept in the top 5. Providing less attributes lead to worse results, but it is understandable, because less attributes are not sufficient to characterize a specific entity and noise in the representation take more easily to wrong identifications.

## 5.9 Summary and Conclusions

### 5.9.1 Ontology Learning from Text

I summarize here the performances of the different Ontology Learning algorithms:

Algorithm	Measured precision
Fionn Murtagh's Algorithm	4.56%
Hearst Patterns on Web Algorithm	57.08%
Hearst Patterns on Web Algorithm (semi-automatic)	89.86%
Maedche and Staab's Algorithm	about 20% (unvalidated)
Maedche and Staab's plus Hearst Patterns Algorithm	about 30% (unvalidated)

While Murtagh's algorithm seems to be useless, the research for Hearst patterns via Web seems to be a good option for the generation of concept hierarchies. Semi-automatic generation provides nearly ready-to-go ontologies.

We expect from the hierarchy extension algorithms to improve with their performances when applied to large ontologies. Anyway, nothing can be said while something in this sense will not be done.

### 5.9.2 Fuzzy Knowledge Model

I summarize here the performances of the tested features:

Algorithm	Score
20 nearest attributes	69.485%
Best concept for some attributes	about 35%
Best concept for some attributes (top 5)	about 60%

The results are encouraging: what is presented in §5.8, in particular, is not a tricky game but represents a well-founded rough draft of a Natural Language Understanding application: the tool here simulates an intelligent agent that in a short time can read thousands of documents and give proof to be able to express learned semantics. If this can, anyway, lead to somewhere, goes far beyond my ability to understand this model's potentialities. Further considerations about how the performances of this model could be improved can be found in §6.2.

## 5.10 Performance Tests

We present here some performance tests in term of time consumed for the execution of the algorithms by the tool versions 2.0 and 2.0 Verbal Version.

### 5.10.1 Extraction 2.0

We noticed that the bottleneck in the execution is represented by the indexing task. Its complexity, anyway, is just  $O(n)$ , with  $n$  the number of the documents; furthermore, as said in (§4.1.4.2), a corpus index can be generated one time and re-used in many executions of the tool.

Tests on a PC with Windows 7 operating system, 2 GB of RAM and an AMD Sempron 3000+ processor, required, for indexing a corpus of 1464 documents, about 8 hours.

Tests on a Server with the Linux Ubuntu operating system, 8 GB of RAM and an Intel Xeon 2.33 GHz processor used, asked, for indexing a corpus of about 1464 documents, more or less 4 hours.

Also, there is to notice the necessity of a good Internet connection speed for the Hearst Patterns on Web algorithm. An ordinary ADSL 2+ 8Mb connection asks for the generation of a hierarchy of 100 terms about 10 minutes, but notice that the complexity of the algorithm is  $O(n \cdot \log(n))$ , where  $n$  is the number of the terms, having the algorithm to check the hyponymy hypotheses of a new term respect to all the terms already put in the hierarchy.

### 5.10.2 Extraction 2.0 Verbal Version

The main characteristic that differentiates this version of the program from the other two is a large usage of NLP algorithms in the indexing process. NLP procedures are very expensive in term of computational resources, and it is useful to talk about the task of extracting SVO triplets: its complexity is  $O(n)$ , with  $n$  the number of the documents. Anyway, it is very time-consuming. Normal PC performances for a corpus of 5 documents is 20 minutes.

Normal PC performances for a corpus of 1464 documents is unreported, taking too long to allow an evaluation.

A Server with the Linux Ubuntu operating system, 8 GB of RAM and an Intel Xeon 2.33 GHz processor used, asked, for indexing a corpus of about 1464 documents, about 14 hours.

## Chapter 6

# Conclusions and Future Work

I present in this chapter, in §6.1, a summary presentation of the results, and the conclusions about this work of thesis. §6.2 is devoted to future work.

### 6.1 Conclusions

We have seen in this thesis work the CA statistical model applied to study, with different approaches, distributional behaviors of words in corpora of documents, in order to extract ontological *is-a* relationships between concepts and their action and affection attributes (this last type of information is provided with an associated distance measure, expressing a probability information).

Precision of the best automatic Ontology Learning from Text approach, Hearst Patterns on Web, is around 60%. Other automatic distributional similarity approaches (§2.1.2.3) demonstrated to have a lower precision, around 40%, but there is to say that they not rely on information acquired from the Web. Anyway, these results demonstrate that our tool represents a valid solution for the automatic generation of hierarchies of concepts. The precision evaluation of Hearst Patterns on Web in its semi-automatic version is around 90%, demonstrating that a user can be reliably assisted by our tool in the creation of concept hierarchies. This assisted procedure represents a good alternative to the more time-consuming option of manually notating hierarchies from scratch.

The nouns/attributes Euclidean space, generated by the tool, results to be valid with a precision of about 70%. We can state, in last analysis, having a look also at the simple conversational game attempted with the tool (§5.8), and the future improvements that can be applied (§6.2), that this model could represent a good source of knowledge for different systems: for instance, for systems simulating abilities of natural language understanding, or systems simulating intelligent agents that can demonstrate to have a good knowledge of specific domains of interest.

## 6.2 Future Work

The Extraction tool is still an experimental application and its features and performances certainly lend themselves to future improvements. Also, as explained in §5.9.2, the Fuzzy Knowledge model can be defined as just a draft of a real model of knowledge, and new researches and experiments should be done in order to understand its real potentialities. I present here the most immediate and important future works that emerge from the current development state of the tool.

### More Tests for the Ontology Generation Algorithms

As it can be seen in Chapter 5, evaluation of precision of ontologies generated by the tool were in little number or, for the hierarchy extension algorithms, also missing. The fact is that manually evaluating ontologies is a very time-consuming operation. More tests should be done with larger ontologies, and recall measure should also be evaluated.

### Support for Further Natural Languages

The tool supports only the English language, but the framework was designed to be easily extended with other languages, provided that Natural Language Processing libraries for Java are available for every specific language.

### User Interface

The actual version of the user interface was created to offer a simpler interaction with the user respect to the textual interaction of Extraction 1.0, but after its creation it has never been evaluated against any standard of usability. It definitely represents a good starting point, but efforts should be made in order to improve it according to some usability standards, increasing the general software quality.

### Relevant Attributes

As seen, while relevant concepts are extracted in respect to a measure of relevance (InfoGain, see §4.1.4.3), no relevance measure is provided for attributes extracted by Extraction 2.0VV. Attributes found in a corpus referring to less than 3 terms are discarded, and the others are kept. A good evaluation measure also for the attributes should be given in order to obtain better performances from the Fuzzy Knowledge model.

### Verb Inflections handler

During the extraction of attributes by Extraction 2.0VV (§4.3.5.1), in order to reduce every inflected form of a verb to the same string, a stemming algorithm was applied to a verb before storing it as an attribute. A better choice could probably be a handler of verbs inflected forms, in order to visualize them in their base form, instead of their root, as it happens now. I did not find anything like that for the Java language, even if probably something could have been done by someone in this sense, maybe a proprietary solution. An inflection handler could also be a good basis for the realization of new Natural Language Understanding features for the tool.



### Multiple Correspondence Analysis

As we have seen in §3.4, CA also has the possibility to concurrently analyze the occurrences of more than two variables. A first step in this sense could be done with our algorithm in order to study the occurrences of SVO triplets in sentences of the documents.

The result would be a tool able to understand the probability of concurrent appearances of both action and affection attributes in a sentence, being not just be able to evaluate sentences, as it happens now, as:

“The chicken is eatable”

but also to handle more complex expressions like:

“John eats the chicken”

Also, the addition to the CA representation of different types of Aristotelian attributes in the same way (we have handled, in this thesis work, just two out of ten, as presented in §1.6), would make our model represent a more complete knowledge map.



# Bibliography

- [Alfonseca E., 2002] Alfonseca E., M. S. (2002). Extending a lexical ontology by a combination of distributional semantics signatures. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*.
- [Alshawi, 1987] Alshawi, H. (1987). Processing dictionary definitions with phrasal pattern hierarchies. In *Computational linguistics - Special issue of the Lexicon*.
- [Amsler, 1981] Amsler, R. (1981). A taxonomy for english nouns and verbs. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*.
- [Aristotle, 340] Aristotle (340). The categories. In *Organon*. 340 BC.
- [Benzécri, 1976] Benzécri, J.-P. (1976). *L'Analyse des Données*. Dounod.
- [Benzécri, 1990] Benzécri, J. P. (1990). Programs for linguistic statistics based on merge sort of text files. In *Les Cahiers de l'Analyse des Données, vol. XIV*.
- [Brunzel, 2008] Brunzel, M. (2008). The xtream methods for ontology learning from web documents. In *Bridging the Gap Between Text and Knowledge*.
- [Buitelaar P., 2004] Buitelaar P., Olejnik D., S. M. (2004). A protégé plug-in for ontology extraction from text based on linguistic analysis. In *Proceedings of the 1st European Semantic Web Symposium*.
- [Calzolari, 1984] Calzolari, N. (1984). Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics*.
- [Caraballo and Charniak, 1998] Caraballo and Charniak (1998). New figures of merit for best-first probabilistic chart parsing. In *Computational Linguistics*.
- [Caraballo, 1999] Caraballo, S. (1999). Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- [Chomsky, 1956] Chomsky, N. (1956). Three models for the description of language. In *I.R.E. Transactions on Information Theory*.
- [Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structures*.
- [Chomsky, 1965] Chomsky, N. (1965). *Aspects of the Theory of Syntax*.
- [Chomsky, 1966] Chomsky, N. (1966). *Topics in the Theory of Generative Grammar*.

- [Chomsky, 1986] Chomsky, N. (1986). *Knowledge of Language*. Praeger.
- [Ciaramita M. et al., 2005] Ciaramita M., Gangemi A., R. E. et al. (2005). Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*.
- [Cimiano, 2006] Cimiano, P. (2006). *Ontology Learning and Population from Text*. Springer.
- [Cimiano P., 2004a] Cimiano P., S. S. (2004a). Learning by googling. In *SIGKDD Explorations*.
- [Cimiano P., 2004b] Cimiano P., Handschuh S., S. S. (2004b). Towards the self-annotating web. In *Proceedings of the 13th World Wide Web Conference*.
- [Cimiano P., 2005] Cimiano P., Ladwig G., S. S. (2005). Gimme' the context: context-driven automatic semantic annotation with c-pankow. In *Proceedings of the 14th World Wide Web Conference*.
- [Cimiano P., 2003] Cimiano P., Pivk A., S. S. (2003). *Learning Taxonomic Relations from Heterogeneous Sources of Evidence*.
- [Dellschaft K., 2004] Dellschaft K., S. S. (2004). Strategies for the evaluation of ontology learning. In *Bridging the Gap Between Text and Knowledge*.
- [Dennett, 1991] Dennett, D. (1991). *Consciousness Explained*. Little, Brown and Co.
- [Dolan W., 1993] Dolan W., Vanderwende L., R. S. (1993). Automatically deriving structured knowledge bases from online dictionaries. In *Proceedings of the Pacific Association for Computational Linguistics*.
- [Eco, 1997] Eco, U. (1997). *Kant e L'Ornitorinco*. Bompiani.
- [Etzioni, 2004] Etzioni, O. (2004). Web-scale information extraction in knowitall. In *Proceedings of the 13th World Wide Web Conference*.
- [Faure D., 1998] Faure D., N. C. (1998). A corpus-based conceptual clustering method for verb frames and ontology. In *Proceedings of the LREC Workshop on Adapting lexical and corpus resources to sublanguages and applications*.
- [Fellbaum, 1998] Fellbaum, C. (1998). *WordNet, an electronic lexical database*. MIT Press.
- [Gamallo P. et al., 2002] Gamallo P., Gonzalez M., A. A. et al. (2002). Mapping syntactic dependencies onto semantic relations. In *Proceedings of the ECAI Workshop on Machine Learning and Natural Language Processing for Ontology Engineering*.
- [Golub and Van Loan, 1996] Golub, G. and Van Loan, F. (1996). *Matrix Computations*. Johns Hopkins University Press.
- [Google, 2009] Google (2009). Google code. <http://code.google.com/intl/it-IT/>.
- [Google Inc., 2010] Google Inc. (2010). Google code. <http://code.google.com/intl/en/>.
- [Grefenstette, 1994] Grefenstette, G. (1994). *Explorations in Automatic Thesaurus Construction*. Kluwer.
- [Gruber, 1992] Gruber, T. (1992). Toward principles for the design of ontologies used for knowledge sharing. In *International Journal Human-Computer Studies*.

- [Gruber, 1993] Gruber, T. (1993). A translation approach to portable ontology specifications. In *Knowledge Acquisition*.
- [Harris, 1968] Harris, Z. (1968). *Mathematical Structures of Language*. Wiley.
- [Hearst, 1992] Hearst, M. (1992). Automatic acquisition of hyponyms from a large text corpora. In *Proceedings of the 14th International Conference of Computational Linguistics*.
- [Hearst M., 1993] Hearst M., S. H. (1993). Customizing a lexicon to better suit a computational task. In *Proceedings of the ACL SIGLEX Workshop on Acquisition of Lexical Knowledge from Text*.
- [Kant, 1781] Kant, I. (1781). *Critique of Pure Reason*.
- [Maedche A., 2003] Maedche A., Pekar V., S. S. (2003). On discovering taxonomic relations from the web. Technical report, Institute AIFB - University of Karlsruhe, Germany.
- [Markert K., 2003] Markert K., Modjeska N., N. M. (2003). Using the web for nominal anaphora resolution. In *Proceedings of the EACL workshop on the Computational treatment of Anaphora*.
- [Miller, 1995] Miller, G. (1995). Wordnet: A lexical database for english. In *Communications of the ACM*.
- [Miller G., 1991] Miller G., C. W. (1991). Contextual correlates of semantic similarity. In *Language and Cognitive Processes*.
- [Milonakis, 2005] Milonakis, A. (2005). My little wubbie.  
<http://www.youtube.com/watch?v=x4BS7PmE-zg>.
- [Murtagh, 2005] Murtagh, F. (2005). *Correspondence Analysis and Data Coding with Java and R*. Chapman & Hall.
- [Murtagh, 2007] Murtagh, F. (2007). Ontology from hierarchical structure in text. Technical report, University of London Egham.
- [Pasca, 2004] Pasca, M. (2004). Acquisition of categorized named entities for web search. In *Proceedings of the Conference on Information and Knowledge Management*.
- [Peirce, 1883] Peirce, C. S. (1883). A theory of probable inference. In *Studies In Logic*.
- [Poesio M., 2005] Poesio M., A. A. (2005). Identifying concept attributes using a classifier. In *Proceedings of the ACL Workshop on Deep Lexical Acquisition*.
- [Pustejovsky, 1991] Pustejovsky, J. (1991). The generative lexicon. In *Computational Linguistics*.
- [Ravichandran D., 2005] Ravichandran D., Pantel P., H. E. (2005). Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics*.
- [Rusu D. et al., 2005] Rusu D., D. L. et al. (2005). Triplet extraction from sentences. Technical report, Technical University of Cluj-Napoca - Romania.

- [Schutze, 1993] Schutze, H. (1993). Word space. In *Advances in Neural Information Processing Systems 5*.
- [University of Vermont, 2003] University of Vermont (2003). Growl.  
<http://www.uvm.edu/skrivov/growl/>.
- [W3C, 1998] W3C (1998). Notation 3.  
<http://www.w3.org/DesignIssues/Notation3.html>.
- [W3C, 2001] W3C (2001). Terse rdf triple language.  
<http://www.w3.org/TeamSubmission/turtle/>.
- [W3C, 2004] W3C (2004). Owl web ontology language.  
<http://www.w3.org/TR/owl-features/>.
- [Widdows, 2003] Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of the Human Language Technology Conference*.
- [Wittgenstein, 1919] Wittgenstein, L. (1919). *Tractatus Logico-Philosophicus*.
- [Wittgenstein, 1953] Wittgenstein, L. (1953). *Philosophical Investigations*. G. Anscombe and R. Rhees.