UNIVERSITÀ DI PISA

Facoltà di Lettere e Filosofia – Facoltà di Scienze M.F.N.
Corso di Laurea Specialistica in Informatica Umanistica

Tesi di Laurea

SEMANTICAL RELATION EXTRACTION AND CLASSIFICATION.

EXPERIMENTS ON WIKIPEDIA.IT

CANDIDATO
Giulia Benotto

CORRELATORE
Prof. Alessandro Lenci

Anno Accademico 2008/2009

Acknowledgements

There's no one left to take the lead
But I tell you and you can see
We're closer now than light years to go
Pick up here and chase the ride
The river empties to the tide
Fall into the ocean
(R.e.m. – Find the River)

Thanks to Marco, for something that words would never express properly, and never be enough to describe. To my parents, for always letting my choose the path to follow, even when they would have done differently, and for always being there for me. I love you both to death. To my advisor, Prof. Lenci, not only for all the help and suggestions he gave me for this Thesis, but also because I fell in love with Computational Linguistics right after attending his first lesson. To Lillian, who kindly reviewed my English, for being an amazing reviewer and even a more amazing boss. To Prof. Simi for all the very useful comments and suggestions. To all the people working on the SemaWiki project, for letting me use their material. To Sarah, for putting up with me for this past 6 years, showing me that friendship exists and being the sister I've always wanted and never had. To Luna and Ally, for all the moral support. To all the people at NURC because they have been the best colleagues a person can dream of. To all the people I should mention and I have probably forgotten. . .

Contents

List of Figures

To my grandma Fedora

And I knew that you were a truth I would rather lose
than to have never knew at all
I miss you

Chapter 1

Introduction

Semantic relations between concepts or entities exist in textual documents, keywords or key phrases, and tags generated in social tagging systems. Relation extraction refers to the identification and assignment of relations between concepts or entities. Basically, it can explore relations that are implicit to underlying data and then add new knowledge to the different domains.

The identification of semantic relations is at the core of Natural Language Processing (NLP) and many applications such as automatic text understanding. Furthermore, semantic relations represent the core elements in the organization of lexical semantic knowledge bases intended for inference purposes. In the past few years at many workshops, tutorials, and competitions this research topic has received considerable interest from the NLP community.

Semantic relation identification is the problem of recognizing, for example, the CAUSE-EFFECT (cycling, happiness) relation in the sentence *He derives great joy and happiness from cycling.* This task requires several local and global decisions needed for relation identification. This involves the meaning of the two noun entities along with the meaning of other words in context.

Previous studies have focused on relation extraction between entities from textual resources. In traditional relation extraction, the sources of entities usually come

from terms in unstructured documents such as web pages or structured documents such as relational databases. A wide variety of data sources have been used in relation extraction research, e.i., web pages (Brin, 1998), and corpus (Bunescu Mooney, 2007).
The semantic and linguistic sources for exploring relations can be a corpus containing the context of entities, and this context information can serve as the basis of relation assignment.

No matter which data sources are utilized in relation extraction, it is necessary to meet three requirements: 1) a collection of data (entity) sources from which semantic relations will be extracted, 2) a semantic or linguistic source in which the context for relations is provided, and 3) algorithms for automatic execution of processing operations. How well a relation extractor performs is determined mainly by the context sources and algorithms. Context containing entities or concepts play a critical role in ensuring the precision of text relation extraction since this provides the source in which covert relations may inhabit.

The purpose of our work was to develop a semi-unsupervised system that was able to automatically extract semantical relations between nominals in a text. In addition, we wanted it to correctly classify semantical relations between nominals. Supervised methods use manually labeled training data to achieve better performance, but we preferred limiting the laborious process of manually annotating the data.
Some algorithms for extracting lexical relations use web resources such as WordNet, or Google counts in order to facilitate the extraction using synset and the number of occurrences of a word within the web, while others do not. Since it was a semi-unsupervised system, our data did not have any annotation, except for the POS tagging, and just a little human intervention.
We did not use any web resources, preferring plain text. Our corpus was someway web related-since it was a dump of the Italian Wikipedia - but it was cleaned of all web-related tags.

Our algorithm used seeds as the starting point for relation extraction, and used a pattern based approach, which led us to deal with the ambiguity of the extracted patterns.
Seeds are words, terms representing the relation that should be extracted. In fact our algorithm selected seeds (nominals) known to be related by the semantic relation we were harvesting for (the two selected relations being *meronymy* and *location*) and extracted patterns intercourring between them. We did not have annotated data, except for the pos tagging, but that was all we needed, since the Perl script we used for pattern extraction, implemented a regular expression that needed only to discriminate between different part of speech, in order to select the ones we wanted to find between the nominals used as seeds, and dismissed the unwanted ones.
Once we gathered a conspicuous number of patterns, we had to analyze their correlation with the relation they were extracted for. We wanted to know the degree of association between the extracted patterns and the relation they are bound to represent.
To do this, we used two associational measure, Mutual Information and Local Mutual Information, which let us infer which pattern was more representative for every relation, according to the ones having a higher MI or LMI value.

Once this task was completed, having selected the more representative patterns for every relation according to our corpus, we wanted to do something slightly different: we then developed a relation classifier. Basically, given a couple of nominals, the system should be able to correctly classify the semantical relation occurring between these nominals.
There is growing interest in the task of classifying semantic relations between pairs of words. However, many different classification schemes have been used, which makes it difficult to compare the various classification algorithms.
For our algorithm, we decided to implement a variation of the Vector Space Model commonly used in Information Retrieval.
Algorithms for classifying semantic relations have potential applications in Information Retrieval, Information Extraction, Summarization, Machine Translation,

Question Answering, Paraphrasing, Recognizing Textual Entailment, Thesaurus Construction, Semantic Network Construction, Word Sense Disambiguation, and Language Modeling, so it seemed pretty useful building our own relation classifier.

We first examined the already harvested *meronymy* and *location* relation. We built n-dimensional vectors for every relation, where each dimension represented an extracted pattern, and which weight was given by the degree of association between the pattern and the relation the vector was bound to represent. This way the results of the previous experiment were used as training set for this one. We then built a n-dimensional vector for each seed couple we wanted to discriminate, and measure the cosine of the angle between the vector representing the seeds and the two vectors representing the semantical relations, in order to determine the one it was closer to.

Motivated by the goals achieved with this experiment, we decided to try our algorithm on different relation, to better evaluate its performance. We were inspired by task 4 of SemEval 2007, so we chose three relation between the seven they proposed for this task, as in: *cause-effect*, *instrument-agent* and *product-producer* relations.
We then repeated the same experiments, using the same relation but implementing automatically extracted seeds, instead of the manually selected ones used before, discovering that it provided even better results.

In Chapter 2, we will go through the state of the art regarding relation extraction and classification. We will examine all the different techniques and approaches commonly used to develop relation extraction algorithms, and how each one of these different approaches is applied to build different algorithm implemented by different linguists. We also examine how algorithms performing relation classification are developed, especially examining SemEval 2007 task 4, since it was an important inspiration for us.

In Chapter 3 we will give a brief description of each step of our algorithm. In addition, we give a description of all the treatment we have submitted our data to, in order to make them ready for the algorithm, and all the treatment our data had already undergone to before we even received them.

In Chapter 4 we will show how we developed the first step of our algorithm, the one in which we extracted semantic relations using manually selected seeds, and how we tested it by making it extract two different types of relations: the standard *part-of* and the *location* relation.

In Chapter 5 we will talk about how we adapted the Vector Space Model often used in Information Retrieval, to develop our own relation classifier. We will show how we used the previously extracted patterns as a training set to teach our modified VSM how to classify nouns that shared a *part-of* or a *location* relation.

In Chapter 6 we test the previously released model to classify manually selected nominals that shared three possible relations: *Cause-Effect*, *Instrument-Agency* and *Product-Producer*. In addition, we try and automatically extract from our corpus other nominals to test, and we use them for both the training and the test set. In the end we compare the results obtained using the almost unsupervised system (which used the automatically extracted words) and the results obtained using the supervised one, which used words that were human-selected.

In Chapter 7 we will talk about our conclusions, and what we obtained with our algorithm.

Chapter 2

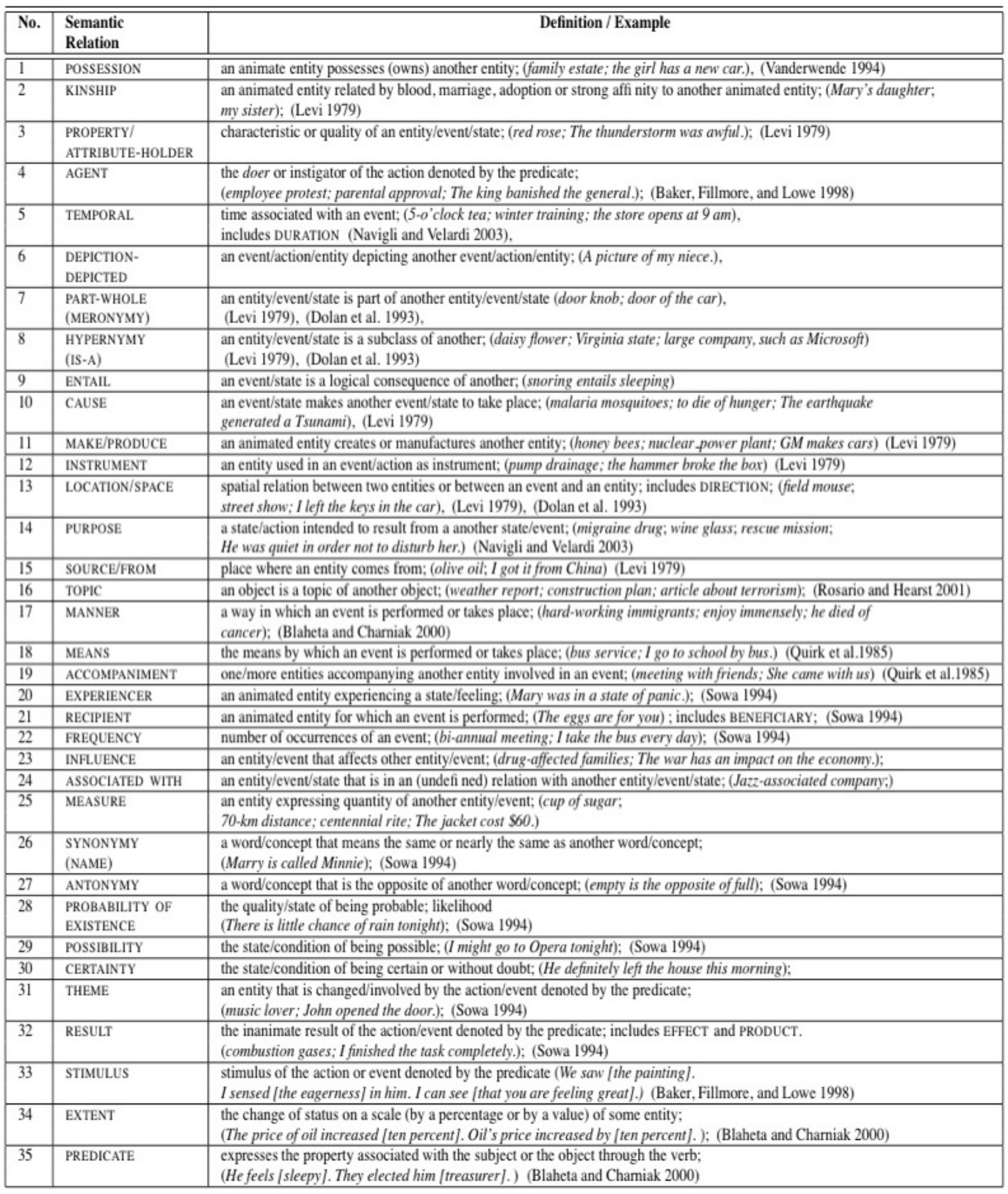
State of the Art

2.1 Information and Relation Extraction

Information Extraction (IE) is an important unresolved problem in natural language processing (NLP). It is a type of information retrieval whose goal is to automatically extract structured information, i.e. categorized and contextually and semantically well-defined data from a particular domain, from unstructured machine-readable documents. Basically it is a shallow form of text understanding that extracts substrings about pre-specified types of entities or relationships from documents and web pages. Example of entities could be people, organizations and locations, while relations could be part-of and location. The location relation is the one that relates a particular name with a certain other name the former is located in. For instance, the sentence "the book is on the shelf" contains the location relation between the object "book" and the place "bookshelf".

In their 2004 paper [1], Moldovan et al. identified a set of semantic relations that cover the majority of text semantics. The relations they identified are listed in the following table:

[1]Dan Moldovan, Adriana Badulescu, Roxana Girju, Marta Tatu, and Daniel Antohe. 2004. *Models for the Semantic Classication of Noun Phrases.* In HLT-NAACL 2004: Workshop on Computational Lexical Semantic, May



No.	Semantic Relation	Definition / Example
1	POSSESSION	an animate entity possesses (owns) another entity; (*family estate; the girl has a new car.*), (Vanderwende 1994)
2	KINSHIP	an animated entity related by blood, marriage, adoption or strong affi nity to another animated entity; (*Mary's daughter; my sister*); (Levi 1979)
3	PROPERTY/ ATTRIBUTE-HOLDER	characteristic or quality of an entity/event/state; (*red rose; The thunderstorm was awful.*); (Levi 1979)
4	AGENT	the *doer* or instigator of the action denoted by the predicate; (*employee protest; parental approval; The king banished the general.*); (Baker, Fillmore, and Lowe 1998)
5	TEMPORAL	time associated with an event; (*5-o'clock tea; winter training; the store opens at 9 am*), includes DURATION (Navigli and Velardi 2003),
6	DEPICTION-DEPICTED	an event/action/entity depicting another event/action/entity; (*A picture of my niece.*),
7	PART-WHOLE (MERONYMY)	an entity/event/state is part of another entity/event/state (*door knob; door of the car*), (Levi 1979), (Dolan et al. 1993),
8	HYPERNYMY (IS-A)	an entity/event/state is a subclass of another; (*daisy flower; Virginia state; large company, such as Microsoft*) (Levi 1979), (Dolan et al. 1993)
9	ENTAIL	an event/state is a logical consequence of another; (*snoring entails sleeping*)
10	CAUSE	an event/state makes another event/state to take place; (*malaria mosquitoes; to die of hunger; The earthquake generated a Tsunami*), (Levi 1979)
11	MAKE/PRODUCE	an animated entity creates or manufactures another entity; (*honey bees; nuclear power plant; GM makes cars*) (Levi 1979)
12	INSTRUMENT	an entity used in an event/action as instrument; (*pump drainage; the hammer broke the box*) (Levi 1979)
13	LOCATION/SPACE	spatial relation between two entities or between an event and an entity; includes DIRECTION; (*field mouse; street show; I left the keys in the car*), (Levi 1979), (Dolan et al. 1993)
14	PURPOSE	a state/action intended to result from a another state/event; (*migraine drug*; *wine glass*; *rescue mission*; *He was quiet in order not to disturb her.*) (Navigli and Velardi 2003)
15	SOURCE/FROM	place where an entity comes from; (*olive oil*; *I got it from China*) (Levi 1979)
16	TOPIC	an object is a topic of another object; (*weather report; construction plan; article about terrorism*); (Rosario and Hearst 2001)
17	MANNER	a way in which an event is performed or takes place; (*hard-working immigrants; enjoy immensely; he died of cancer*); (Blaheta and Charniak 2000)
18	MEANS	the means by which an event is performed or takes place; (*bus service; I go to school by bus.*) (Quirk et al.1985)
19	ACCOMPANIMENT	one/more entities accompanying another entity involved in an event; (*meeting with friends; She came with us*) (Quirk et al.1985)
20	EXPERIENCER	an animated entity experiencing a state/feeling; (*Mary was in a state of panic.*); (Sowa 1994)
21	RECIPIENT	an animated entity for which an event is performed; (*The eggs are for you*) ; includes BENEFICIARY; (Sowa 1994)
22	FREQUENCY	number of occurrences of an event; (*bi-annual meeting; I take the bus every day*); (Sowa 1994)
23	INFLUENCE	an entity/event that affects other entity/event; (*drug-affected families; The war has an impact on the economy.*);
24	ASSOCIATED WITH	an entity/event/state that is in an (undefi ned) relation with another entity/event/state; (*Jazz-associated company;*)
25	MEASURE	an entity expressing quantity of another entity/event; (*cup of sugar; 70-km distance; centennial rite; The jacket cost $60.*)
26	SYNONYMY (NAME)	a word/concept that means the same or nearly the same as another word/concept; (*Marry is called Minnie*); (Sowa 1994)
27	ANTONYMY	a word/concept that is the opposite of another word/concept; (*empty is the opposite of full*); (Sowa 1994)
28	PROBABILITY OF EXISTENCE	the quality/state of being probable; likelihood (*There is little chance of rain tonight*); (Sowa 1994)
29	POSSIBILITY	the state/condition of being possible; (*I might go to Opera tonight*); (Sowa 1994)
30	CERTAINTY	the state/condition of being certain or without doubt; (*He definitely left the house this morning*);
31	THEME	an entity that is changed/involved by the action/event denoted by the predicate; (*music lover; John opened the door.*); (Sowa 1994)
32	RESULT	the inanimate result of the action/event denoted by the predicate; includes EFFECT and PRODUCT. (*combustion gases; I finished the task completely.*); (Sowa 1994)
33	STIMULUS	stimulus of the action or event denoted by the predicate (*We saw [the painting]. I sensed [the eagerness] in him. I can see [that you are feeling great].*) (Baker, Fillmore, and Lowe 1998)
34	EXTENT	the change of status on a scale (by a percentage or by a value) of some entity; (*The price of oil increased [ten percent]. Oil's price increased by [ten percent].*); (Blaheta and Charniak 2000)
35	PREDICATE	expresses the property associated with the subject or the object through the verb; (*He feels [sleepy]. They elected him [treasurer].*) (Blaheta and Charniak 2000)

They observed that most of the time, semantic relations are encoded by lexico-syntactic patterns that are highly ambiguous. One pattern can express a number of semantic relations, its disambiguation being provided by the context or world knowledge. Often, semantic relations are not disjointed or mutually exclusive. For example, the expression "*Texas city*" contains both a LOCATION and a PART-WHOLE relation. In our experiment, we tried to classify words between this two relations but we sometimes reported incorrect results because of this ambiguity. All the relations covered in our experiment are reported in the table above.

Semantic relations may be classified in various ways.

The first major distinction is between a domain-independent relation and a domain specific relation. A domain independent relation is a semantic relation that can be applied across several distinct domains, like, for example, meronymy or location relation that are true within several domains. Domain-related relations, however, apply only to a specific domain. For example the *Organism_Protein()* relation is domain-related because it is restricted to the Biology domain. In our experiment, we used only domain-independent relations. This was due to the type of corpus we had, which was not a domain-related corpus, but a corpus covering different topics.

There are different ways of extracting relations from text. One way to perform this task is the use of either supervised methods and unsupervised methods. Supervised methods use manually labeled training data to achieve better performance, while unsupervised methods implement other techniques to avoid the laborious process of manually labeling the training data. The method we adopted, as described in the next chapter, may be considered partially unsupervised, since the only kind of annotation used for its analysis is POS tagging, and there is just a minimum human intervention.

Some algorithms for extracting lexical relations use web resources such as WordNet, while others do not. WordNet is a large lexical database of English, in which nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The purpose is twofold: to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. Due to its nature, WordNet has been widely used as a corpus for relation extraction.
There are also other experiment which use just text corpus using the required labels in case of supervised approach.

For our work, we decided not to use a web resource as a corpus, preferring plain text instead. Although our corpus was someway web related - since it was a dump of the Italian Wikipedia - it was cleaned of all web-related tags.

Approaches for extracting lexical relations fall into two others categories: pattern-based or cluster-based. Pattern-based approaches are more common; using a pattern-based approach means searching the corpus for specific lexical relations, expressed in very prototypical ways. The majority of published work on relation extraction describe pattern-based approaches. Unfortunately, when looking at patterns, we have to consider their ambiguity. As described previously, a sentence like *Texas city*" contains both a LOCATION and a PART-WHOLE relation. We will take a better look at pattern-based approach later. The clustering approach, which is not as common, uses clustering algorithms to group words according to their meaning in text, labeling the clusters using its members' lexical or syntactic dependencies, and then extracting relations between each cluster member and the cluster label. Caraballo [2] first proposed this approach in 1999, which used conjunction and apposition features to build noun clusters. In 2004 Pantel and Ravichandran [3] extended this approach by making use of all syntactic dependency features for each noun. The advantage of clustering approaches is that they permit algorithms to identify is-a relations that do not explicitly appear in text. However they generally fail to produce coherent clusters from fewer than 100 million words; hence they are unreliable for small corpora.

We decided to implement a pattern-based approach for our algorithm.

Another approach for relation extraction distinguishes between systems that use seeds for extracting relation and systems that do not. Seeds are words, terms representing the relation that should be extracted. There are many algorithms implementing seeds for different purposes and usages: our algorithm implemented seeds as the starting point for relation extraction.

[2] Sharon Caraballo. 1999. *Automatic acquisition of a hypernym-labeled noun hierarchy from text.* In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, June.

[3] Patrick Pantel and Ravichandran Deepak . 2004. *Automatically Labeling Semantic Classes.* In HLT-NAACL 2004: Main Proceedings, May.

2.1.1 Hearst's Automatic Acquisition of Hyponyms

In 1992 Hearst [4] described a method for the automatic acquisition of the hyponymy lexical relation from unrestricted text. Hyponymy (or *is-a*) relation exists when a event, entity, state, is a subclass of another. For example, *daisy-flower* are related by a hyponymy relation, because the daisy is a flower. Two goals motivated Hearst and her approach: avoidance of the need for pre-encoded knowledge and applicability across a wide range of text. At that time, there was much interest in the automatic acquisition of lexical syntax and semantics, with the goal of building up large lexicons for natural language processing. Extracting lexical information from machine readable dictionaries was a great success but was limited- because the set of entries within a dictionary is fixed.

Interpreting unrestricted domain-independent text made it difficult to determine in advance what kind of information would be encountered and how it would be expressed. Instead of interpreting everything in the text in detail, Hearst used a pattern-based approach, that is, she searched for specific lexical relations that were expressed in well-known ways. She discovered that useful information could be found with only a very simple understanding of a text.

Given the sentence:

The bow lute, such as the Bambara ndang, is plucked and has an individual curved neck for each string.

Most fluent readers of English who never encountered the word “Bambara ndang” would nevertheless infer that a “Bambara ndang” is a kind of “bow lute”, even if they had a fuzzy conception of what a “bow lute” was. The sentence above did not deliberately defined the term, as a dictionary would have done, however the semantics of the lexico-syntactic construction indicated by the pattern:

$$NP_0 \text{ such as } \{NP_1, NP_2 .... (and|or)\} NP_n$$

[4] Marti A. Hearst. 1992. *Automatic acquisition of hyponyms from large text corpora* . In:Proceedings of the 14th conference on Computational linguistics

were such that they implied:

$$\text{for all } NP_t, 1 \leq i \leq n, hyponym(NP_i, NP_0)$$

Thus from the sentence it was possible to extract the following semantic relation:

hyponymy("Bambara ndang", "bow lute")

She used the term hyponym as: a concept represented by a lexical item L_1 if native speakers of a certain language accept sentences constructed from the frame *An* L_0 *is a (kind of)* L_1*).* Here L_1 is the hypernym of L_0. The relationship is reflexive and transitive, but not symmetric.
The given example shows a way to discover a hyponymic lexical relationship between two or more noun phrases in a naturally-occurring text.

According to Hearst, there were many ways in which the structure of a language could indicate the meanings of lexical items, but the difficulty lied in finding constructions that frequently and reliably indicate the relation of interest. Lexical relations are often not mutually exclusive, as such, a pattern could be ambiguous and consequentially could be found to contain a certain relation upon further examination, therefore it may not be a reliable representative for that relation. It may seem that- because free text is so varied in form and content, it may not be possible to find such constructions. However, Hearts identified a set of lexico-syntactic patterns that indicates the hyponymy relation and that satisfies the following criteria:

1. They occur frequently and in many text genres.
2. They (almost) always indicate the relation of interest.
3. They can be recognized with little or no pre-encoded knowledge.

Item 1indicated that the pattern would result in the discovery of many instances of the relation, item 2 that the information extracted would not be erroneous, and item 3 that making use of the pattern did not require the tools that it was intended to help build. Finding instances of the hyponymy relation was useful for several purposes like lexicon augmentation, noun phrase semantics and semantic relatedness information.

Hyponymy Relations may be used to augment and verify existing lexicons, including ones built from machine readable dictionaries. Another purpose to which these relations may be applied is the identification of the general meaning of an unfamiliar noun phrase. For example, discovering the predicate

hyponymy("broken bone", "injury")

indicates that the term "broken bone" might be understood at some level as an "injury" without having to determine the correct sense of the component words and how they combine. A term like "broken bone" is not likely to appear in a dictionary or lexicon, although it is a common locution. Having the discovered relations closely related semantically instead of hyponymically is most felicitous when the noun phrases involved are modified and atypical. Consider, for example, the predicate:

hyponymy("detonating explosive", "blasting agent")

This relation might not be a canonical *is-a* relation but the fact that it was found in a text implies that the words meanings are close. Connecting terms whose expressions are quite disparate but whose meanings are similar should be useful for improved synonym expansion in information retrieval and for finding chains of semantically related phrases, as used in the approach for recognizing topic boundaries (Morris and Hirst 1991)[5]. Hearst observed that terms occurring in a list are often related semantically, whether they occur in a hyponymy relation or not.

[5]Jane Morris and Graeme Hirst.1991.*Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text.*In: Computational Linguistics, Volume 17, Number 1,March

(2) such NP as $\{NP,\} * \{(or|and)\}$ NP

... works by such authors as Herrick, Goldsmith, and Shakespeare.

→ hyponym (“author”, “Herrick”),

hyponym(“author”, “Goldsmith”),

hyponym(“author”, “Shakespeare”).

(3) NP $\{, NP\} * \{,\}$ or other NP

Bruises, wounds, broken bones or other injuries . . .

→ hyponym(“bruise”, “injury”),

hyponym (“wound”, “injury”),

hyponym(“broken bone”, “injury”)

(4) NP $\{, NP\} * \{,\}$ and other NP

... temples, treasuries,and other important civic buildings.

→ hyponym(“temple”, “civic building”),

hyponym(“treasury ”, “civic building”)

(5) NP $\{,\}including\{NP,\} * \{or|and\}$ NP

All common-law countries, including Canada and England ...

→ hyponym(“Canada”, “common-law country”),

hyponym (“England”, “common-law country”)

(6) NP $\{,\}especially\{NP,\} * \{or|and\}$ NP

. . . most European countries, especially France, England, and Spain.

→ hyponym(“France”, “European country”),

hyponym(“England”, “European country”),

hyponym(“Spain”, “European country”)

When a relation hyponym(NP_0, NP_1) is discovered, aside from some lemmatizing and removal of unwanted modifiers, the noun phrase is left as an atomic unit, not broken down and analyzed.

If a more detailed interpretation is desired, the results could be passed on to a more intelligent or specialized language analysis component. This kind of discovery procedure could be a partial solution for a problem like noun phrase interpretation, because at least part of the meaning of the phrase is indicated by the hyponymy relation.

However Hearst found some difficulties in her experiment. In example (4) above, the full noun phrase corresponding to the hypernym is "other important civic buildings". This illustrates a difficulty that resulted from using free text as the data source, as opposed to a dictionary - often the form that a noun phrase that occurred in the text was not what was supposed to be recorded. For example, nouns frequently occur in their plural form even if, for a task like this, extracting the singular form would be better. Adjectival quantifiers such as "other" and "some" are usually undesirable and could be eliminated in most cases without making the statement of the hyponym relation erroneous. Comparatives such as "important" and "smaller" are usually best removed, since their meaning is relative and dependent on the context in which they appear.

How much modification is desirable depends on the application to which the lexical relations will be put. For developing a basic, general-domain thesaurus, single-word nouns and very common compounds are most appropriate. For a more specialized domain, more modified terms have their place. For example, noun phrases in the medical domain often have several layers of modification which should be preserved in a taxonomy of medical terms.

How did Hearst's system discover patterns? Initially she discovered patterns (1)-(3) by observation, looking through text and noticing the patterns and the relationships they indicated. In order to find new patterns automatically, she sketched the following procedure:

1. Decide on a lexical relation, R, that is of interest, e.g., "group/member" (in her formulation this was a subset of the hyponymy relation).

2. Gather a list of terms for which this relation was known to hold, e.g., "England-country". This list could be found automatically using the method

described here, bootstrapping from patterns found by hand, or by bootstrapping from an existing lexicon or knowledge base.

3. Find places in the corpus where these expressions occurred syntactically near one another and record the environment.

4. Find the commonalities among these environments and hypothesize that common ones yield patterns that indicated the relation of interest.

5. Once a new pattern had been positively identified, use it to gather more instances of the target relation and go to Step 2.

Hearst tried this procedure by hand using just one pair of terms at time. In the first case she tried the "England-country" example and with just this pair she found new patterns (4), (5) as well as (1) and (3) which were already known. Next she tried "tank-vehicle" and discovered a very productive pattern, pattern (6).
Hearst had tried applying this technique to meronymy (i.e., the part/whole relation), but without great success. The patterns found for this relation did not tend to uniquely identify it, but could be used to express other relations as well. It might be the case that in English the hyponymy relation is especially amenable to this kind of analysis, perhaps due to its "naming" nature.
However, she did not succeeded at the identification of more specific relations, such as patterns that indicated certain types of proper nouns. She had not implemented an automatic version of this algorithm, primarily because Step 4 was undetermined.
Thus, we can say that Hearst's approach used seeds, since she gathered a list of terms for which the relation she was examining was known to hold. She also used a pattern-based approach, as reflected in the described procedures.. Hearst ran the previously illustrated algorithm on *Grolier's American Academic Encyclopedia* (Grolier 1990). Once the algorithm has completed running, she found that, of 8.6 M words of encyclopedia text, there were 7607 sentences that contained the lexemes "such as " contiguously. Out of these, 152 relations fitted the restriction

of the experiment, namely that both the hyponyms and the hypernyms were unmodified. When the restrictions were eased slightly, so that seeds consisting of two nouns or a present/past participle plus a noun were allowed, 330 relations were found. After the algorithm completed the last step and all the found relations were gathered, Hearst looked up for these relations in WordNet. She found that only 180 of the 226 unique words involved in the relations actually existed in the hierarchy, and 61 out of 106 feasible relations (i.e. relations in which both terms were already found in WordNet, were found. The quality of the relations found seems high overall, although there were difficulties. As to be expected, metonymy occurs, as seen in hyponym("king", "institution").

A more common problem was under-specification. For example, one relation was hyponym ("steatornis", "species"), which was problematic because the kind of species needed to be known and most likely this information was mentioned in the previous sentence. Similarly, relations were found between "device" and "plot", "metaphor", and "character", under-specifying the fact that literary devices of some sort were under discussion. Sometimes the relationship expressed was slightly askance of the norm. For example, the algorithm found hyponym("Washington", "nationalist") and hyponym("aircraft", "target") which were somewhat context and point-of-view dependent. According to Hearst this was not a problem, since finding alternative ways of stating similar notions was one of her goals. However, it was important to try to distinguish the more canonical and context-independent relations for entry in a thesaurus.

There were a few relations whose hypernyms were very high-level terms, e.g., "substance" and "form". These were not incorrect; they just might not be as useful as more specific relations.
Overall, Hearst found the results encouraging, although the number of relations found was small compared to the size of the text used. She suggested that this situation might be improved using less stringent restrictions, that would increase the numbers, as the slight loosening shown in the Grolier's experiment indicated. A more savvy grammar for the constituent analyzer should also increase the results.

The next problem was how to automatically insert relations between terms into the hierarchy used, that is WordNet. This involved two main difficulties. First, if both lexical expressions were present in the noun hierarchy but one or both had more than one sense, the algorithm must decide which senses to link together. Hearst had some preliminary ideas on how to work around this problem.
Let's say that the hyponym in question had only one sense, but the hypernym had several. The task was simplified to determining which sense of the hypernym to link the hyponym to and to make use of a lexical disambiguation algorithm, e.g., (Hearst 1991)[6] Furthermore, since it was assumed the hyponym had only one main sense, the following could be done: look through a corpus for occurrences of the hyponym and see if its environment tended to be similar to one of the senses of its hypernym. For example, if the hypernym was "bank" and the hyponym was "First National", every time, within a sample of text, the term "First National" occurred, it was necessary to replace it with "bank", and then run the disambiguation algorithm as usual. If the term could be positively classified as having one sense of bank over the others, then this would provide strong evidence as to which sense of the hypernym to link the hyponym to. The second main problem with inserting new relations raised when one or both terms did not occur in the hierarchy at all. In this case, it was necessary to determine which, if any, existing synset the term belonged in and then did the sense determination mentioned above. Hearst's low-cost approach for automatic acquisition of semantic was meant to provide an incremental step toward the larger goals of natural language processing. Her approach was complementary to statistically based approaches that found semantic relations between terms, but hers required a single specially expressed instance of a relation while the others required a statistically significant number of generally expressed relations. Hearst had shown that her approach was also useful as a evaluating component for existing knowledge bases and lexicons.

[6]Marti A. Hearst. 1991. *Noun Homograph Disambiguation Using Local Context in Large Text Corpora.* In: Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New OED and Text Research, to determine which sense of the hypernym was being used in the sample sentence. October

2.1.2 Berland & Charniak's algorithm for finding parts in a large corpora

In 1999 Berland and Charniak[7] presented a method for extracting parts of objects from the whole (e.g. "speedometer " from "car").
Given a single word denoting some entity that has recognizable parts, the system found and ordered other words that might denote parts of the entity in question, or given Cruse[8] definition, a meronymy relation. Berland and Charniak based their work on Hearst's, using a similar methodology and applying that to the "part-of" relation instead than the "is-a" relation analyzed by Hearst. While she failed in applying her theory to the "part-of" relation, they succeeded by implementing some differences that were really important.

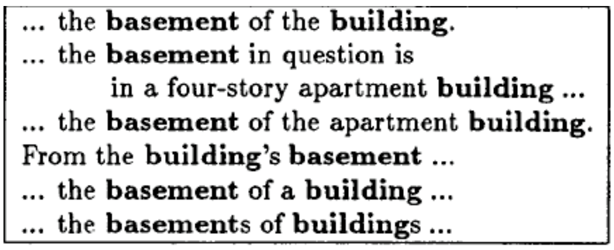
One of the biggest problem they found in their work was the vagueness of the definition of the concept "part". Back then, Webster' s Dictionary defined "part" as "one of the often indefinite or unequal subdivisions into which something is or is regarded as divided and which together constitute the whole". Using such a definition, it was difficult to determine what exactly constituted a part, which translated into some doubts about evaluating the results of any procedure that claimed to find them. The definition did not claim that parts must be physical objects, so "novel" might have "plot" as a part. This problem was handled by asking informants which words in a list were parts of some target word, and then declaring majority opinion to be correct; in other words, human selectors choose the seeds. While the subjects often disagreed, there was fair consensus that what may count as part depends on the nature of the concept expressed by the word: a physical object yields physical parts, an institution its members, a concept its characteristics and processes.

Berland and Charniack' s first goal was to find lexical patterns that tend to indicate part-whole relations. Once again they followed Hearst and found possible patterns

[7]M Berland, E Charniak. 1999. *Finding Parts in very large corpora.* In: Proceedings of the 37th annual meeting of the ACL

[8]D.A. Cruse. 1989. *Lexical Semantics.* Cambridge Textbook in Linguistics; p. 159

by taking two words that were in a part-whole relation and finding sentences in a corpus (the North American News Corpus) that had those words within close proximity. For the couple of words "basement" and "building" the first few sentences were:



... the **basement** of the **building**.
... the **basement** in question is
in a four-story apartment **building** ...
... the **basement** of the apartment **building**.
From the **building's basement** ...
... the **basement of a building** ...
... the **basements** of **buildings** ...

From these examples, they constructed the following patterns:

A. **whole NN[-PL] 's POS part NN[-PL]**

. . . building's basement . . .

B. **part NN[-PL] of PREP the—a DET mods [JJ—NN]* whole NN**

. . . basement of a building...

C. **part NN in PREP the—a DET mods [JJINN]* whole NN**

. . . basement in a building . . .

D. **parts NN-PL of PREP wholes NN-PL**

. . . basements of buildings . . .

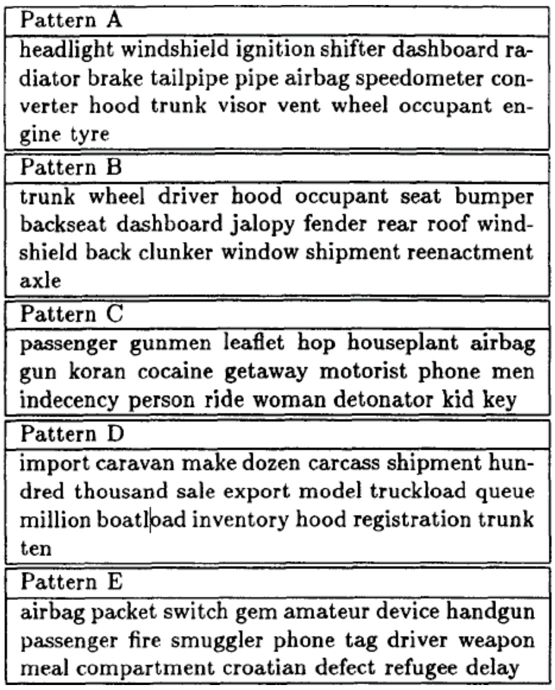
E. **parts NN-PL in PREP wholes NN-PL**

. . . basements in buildings . . .

They assumed here that parts and wholes were represented by individual lexical items (as head nouns of noun-phrases) as opposed to complete noun phrases, or as a sequence of "important" noun modifiers together with the head.

This could cause some problems, e.g., "conditioner" was marked by their informants as not part of "car", whereas "air conditioner" probably would have made it into a part list. Nevertheless, in most cases head nouns had worked quite well on their own. Berland and Charniak evaluated these patterns by observing how they performed in an experiment on a single example.

The following table shows the 20 highest ranked part words (with the seed word "car") for each of the patterns A-E.



Pattern A
headlight windshield ignition shifter dashboard radiator brake tailpipe pipe airbag speedometer converter hood trunk visor vent wheel occupant engine tyre
Pattern B
trunk wheel driver hood occupant seat bumper backseat dashboard jalopy fender rear roof windshield back clunker window shipment reenactment axle
Pattern C
passenger gunmen leaflet hop houseplant airbag gun koran cocaine getaway motorist phone men indecency person ride woman detonator kid key
Pattern D
import caravan make dozen carcass shipment hundred thousand sale export model truckload queue million boatload inventory hood registration trunk ten
Pattern E
airbag packet switch gem amateur device handgun passenger fire smuggler phone tag driver weapon meal compartment croatian defect refugee delay

As shown, patterns A and B outperformed patterns C, D, and E. Although parts occurred in all five patterns, the lists for A and B were predominately parts-oriented. The relatively poor performance of patterns C and E was anticipated, as many things occur "in" cars (or buildings, etc.) other than their parts.

Pattern D was not so obviously bad as it differed from the plural case of pattern B only in the lack of the determiner "the" or "a". However, this difference proved critical in that pattern D tended to pick up "counting" nouns such as "truckload". On the basis of this experiment they decided to continue using only patterns A and B.

As mentioned, Berland and Charliank used the North American News Corpus (NANC), which was a compilation of the wire output of several US newspapers.

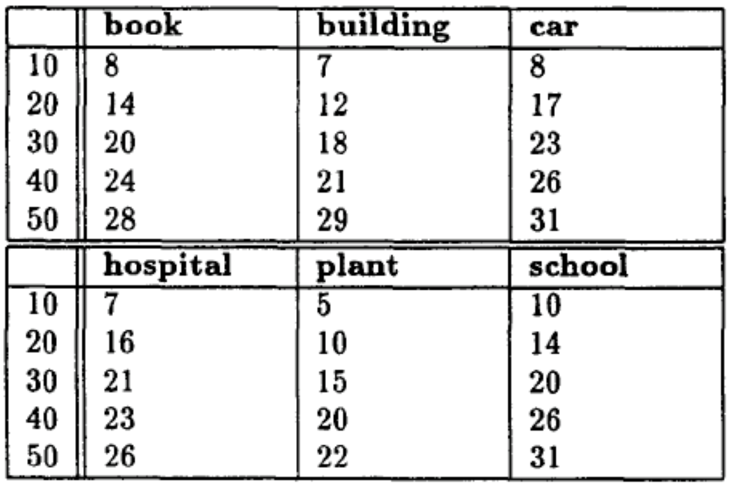
The total corpus was about 100,000,000 words. Running the program on the whole data set took roughly four hours, the bulk of which was spent tagging the corpus.

A drawback of the NANC was the occurrence of repeated articles. Since the corpus consisted of all the articles that came over the wire, some days it contained multiple, updated version of the same stories, containing identical paragraphs or sentences. At first, Berland and Charliank wrote a program to weed out such cases but eventually gave up, due to the fact that "update" article may actually have substantial variations, so there was a continuum between those and articles that were simply on the same topic. Also, the data Berland and Charniak were working on was so sparse that any such repeats were very unlikely to manifest themselves as repeated examples of part-type pattens.

The seeds they implemented were one word and its plural. Words chosen as seeds were words with an high probability of being found in the corpus (such as "building" and "hospital") that they thought would have parts that might also be mentioned therein. They suggested that with enough text, it would be possible to get reasonable results with every noun that met these criteria.

The developed algorithm was composed of three phases.
The first identified and recorded all occurrences of patterns A and B in the corpus.
The second filtered out all words ending with the suffixes "ing", "ness" or "ity", since these suffixes typically occur in words that denote a quality rather than a physical object.
The so extracted words, which possibly represent seeds of the meronymy relation, were ordered by the similarity they reported with the meronymy relation, that is, they were ordered according to how likely they represented the relation, according to some appropriate metric selected. They tested five Berland and Charniak subjects (all of whom were unaware of their goals) for their concept of a "part". They asked the subjects to rate sets of 100 words, 50 of which were in their final results set.
The score of individual words varied greatly but there was relative consensus on most words. Lacking a formal definition of part, and considering the ambiguity

of the meronymy relation, they could only define as correct the words there were marked as part of the concept by the testing subjects, marking the rest as wrong. While the scoring was not perfect, it provided an adequate reference result. The following table summarizes these results



	book	**building**	**car**
10	8	7	8
20	14	12	17
30	20	18	23
40	24	21	26
50	28	29	31

	hospital	**plant**	**school**
10	7	5	10
20	16	10	14
30	21	15	20
40	23	20	26
50	26	22	31

There is shown the number of correct part words in the top 10, 20, 30, 40, and 50 parts for each seed (e.g., for “book”, 8 of the top 10 are parts, and 14 of the top 20).

Overall, about 55% of the top 50 words for each seed were parts, and about 70% of the top 20 for each seed. One ambiguous word, “plant” was chosen, to see what would happen. The program founds parts corresponding to both senses, though given the nature of the text, the industrial use was more common. Testing subjects marked both kinds of parts as correct, but even so, this produced the weakest part list of the six words they tried. As a baseline, the authors also tried using the head nouns that immediately surround their target word as their “pattern” and then applied the same ”strong conditioning, sigdiff” statistical test to rank the candidates.

This performed quite poorly. Of the top 50 candidates for each target, only 8% were parts, as opposed to the 55% for the program. Berland and Charniak’ s program could find parts of objects, given a word denoting the whole object and a large corpus of unmarked text. The program was about 55% accurate for the top 50 proposed parts for each of six examples upon which they tested it. There did not seem to be a single cause for the 45% of the cases that were mistakes.

A few problem were identified. Idiomatic phrases like “a jalopy of a car” or ” the son of a gun” provided problems that were not easily weeded out. Depending on the data, these phrases could be as prevalent as the legitimate parts. In some cases problems arose because of tagger mistakes. For example, “re-enactment” would be found as part of a “car” using pattern B in the phrase “the re-enactment of the car crash” if “crash” was tagged as a verb. The program had some tendency to find qualities of objects. For example, “drivability” is strongly correlated with car. They tried to weed out most of the qualities by removing words with the suffixes “ing”, “ness” and “ity”, The most persistent problem was sparse data, which was the source of most of the noise. More data would almost certainly allow them to produce better lists, both because the statistics would be more accurate, but also because larger numbers would allow to find other reliable indicators. For example, idiomatic phrases might be recognized as such. So we see “jalopy of a car” (two times) but not, of course, “the car’s jalopy”. Words that appeared in only one of the two patterns were suspect, but to use this rule sufficient count on the good words was needed to ensure a representative sample. Hearst tried to find parts in corpora but did not achieve good results. According to the authors their program worked better than Hearst’s because of their very large corpus and the use of more refined statistical measures for ranking the output.

2.1.3 A brief synthesis of Jones and Riloff's bootstrapping algorithm for text learning tasks

In 1999 Riloff and Jones et. al[9] noticed that when applying text learning algorithms to complex tasks, it was tedious and expensive to manually label the large amounts of training data necessary for good performance. Text learning algorithms at the time were reasonably successful when provided with enough labeled or annotated training examples. However, as more complex domains were considered, the requisite size of these training sets got prohibitively large. Creating these training sets became tedious and expensive, since typically they must be labeled by a person.

This led to a search for learning algorithms that do not require such large amounts of labeled data. While labeled data is difficult to obtain, unlabeled data is readily available and plentiful. Castelli and Cover (1995)[10] showed that unlabeled data could be used in some settings to improve classification, although it was exponentially less valuable than labeled data.

However, one cannot learn to perform classification from just unlabeled data alone. By itself, unlabeled data describe the domain of the problem, but not the task over the domain. Thus, unlabeled data must be coupled with at least some information about the target function for the learning task. This target, or seed information can come in many different forms, such as keywords or features which may appear in examples of the target classes, or a small number of examples of the target classes.

Riloff and Jones tried using a bootstrapping framework for text learning tasks that would otherwise require large training sets. The input to the bootstrapping process was a large amount of unlabeled data and a small amount of seed information to inform the algorithm about the specific task at hand. Seed information took the

[9]E. Riloff , R. Jones, A. McCallum and K. Nigam 1999.*Bootstrapping for Text Learning Tasks* . In:JCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications Finite-State Transducers for Semi-Structured Text Mining (AAAI-99) .

[10]V. Castelli and T. M. Cover. 1995.*On the exponential value of labeled samples.* In: Pattern Recognition Letters. pp. 105-111

form of keywords associated with classes. Bootstrapping initialize a learner with the keywords which are then applied by the learner to suggest labels for unlabeled data which in turn builds a new learner, with each iterative process refining on the results.

There were two instantiations of the bootstrapping approach for different text learning tasks. The first case study was performed by Riloff and Jones[11]. Their goal was to automate the construction of both a lexicon and extraction patterns for a semantic category using bootstrapping. The heart of their approach was based on *mutual bootstrapping*, that is, the observation that extraction patterns can generate new examples of a semantic category, which in turn could be used to identify new extraction patterns.

The mutual bootstrapping process began with a text corpus and a handful of predefined seed words for a semantic category. Before bootstrapping began, the text corpus was used to generate a set of candidate extraction patterns. They used AutoSlog (Riloff 1993;1996)[12] to generate extraction patterns for every noun phrase in the corpus. Given a noun phrase to extract, AutoSlog used heuristics to generate a linguistic expression that represented relevant context for extracting the noun phrase. This linguistic expression should be general enough to extract other relevant noun phrases as well. Because of the exhaustive application of AutoSlog, the complete set of extraction patterns produced was capable of extracting every noun phrase in the training corpus. Then, Riloff and Jones applied the extractions pattern to the corpus and recorded their extractions. Using these data, the mutual bootstrapping procedure identified the extraction pattern that was most useful for extracting known category members. This extraction pattern was then used to propose new phrases that belonged in the same lexicon. At each iteration, the algorithm saved the best extraction pattern for the category examined to a list. All extractions were assumed to be category members and were added to the semantic lexicon. Then the next best extraction pattern was identified, based on

[11] E.Riloff and R. Jones. 1999. *Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping.* In: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)

[12] E. Riloff. 1996. *Automatically Generating Extraction Patterns from Untagged Text.* In: {AAAI}/{IAAI}, Vol. 2. pp 1044-1049

both the original seed plus the new words that were just added to the lexicon, and the process repeated. Since the semantic lexicon was constantly growing, the extraction patterns were re-scored after each iteration, using a scoring heuristic based on how many lexicon entries were extracted by a pattern. A pattern that extracted a variety of category members would be scored higher than a pattern that extracted only one or two different category, no matter how often. Scoring was also based on "Head phrase " matching, instead of requiring a exact match. This meant that X matched Y if X was the rightmost substring of Y. For example, "New Zealand " would match any phrase that ends with "New Zealand " , such as "eastern New Zealand ", but would not match "the New Zealand coast". Head phrase matching was important for generality because any noun phrase could be preceded by an arbitrary numbers of modifiers. Riloff and Jones tested this approach by generating dictionaries for locations from corporate web pages used in the WebKB project (Craven et. al, 1998)[13]. Meta bootstrapping identified 191 locations phrases in the web pages. After 50 iterations, 76% of the hypothesized location phrases on the web pages, were true locations.

2.1.4 Snowball: Agitech and Gravano's way of extracting relations from Large Plain text Collections and their inspiration DIPRE, by Brin

In 2000 Agichtein and Gravano[14] noticed that text documents often contain valuable structured data that is hidden in regular English sentences. This data is best exploited if available as a relational table that could be used for answering precise queries or for running data mining tasks. They explored a technique for extracting such tables from document collections that required only a handful of training examples from users.

[13]M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery,*Learning to Extract Symbolic Knowledge from the World Wide Web.* In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)

[14]E. Agichtein, L. Gravano.2000. *Snowball: Extracting Relations from Large Plain-Text Collections.* In: Proceedings of the 5th ACM International Conference on Digital Libraries

These examples were used to generate extraction patterns that in turn resulted in new tuples being extracted from the document collection, in a process called Snowball.
Snowball introduced novel strategies for generating patterns and extracting tuples from plain-text documents. At each iteration of the extraction process, Snowball evaluated the quality of these patterns and tuples without human intervention, and kept only the most reliable ones for the next iteration.Agichtein and Gravano also developed a scalable evaluation methodology and metrics for the task, presented thorough experimental evaluation of Snowball and comparable techniques over a collection of more than 300,000 newspaper documents.

Text documents often hide valuable structured data. For example, a collection of newspaper articles might contain information on the location of the headquarters of a number of organizations.

If we need to find the location of the headquarters of, say, Apple Computers, we could use traditional information-retrieval techniques to find documents that contain the answer to the query. Alternatively, we could answer such a query more precisely if we somehow had available a table listing all the organization-location pairs mentioned in our document collection. A tuple $< o, l >$ in such table would indicate that the headquarters of organization o are in location l, and that this information was present in a document in our collection. Tuple $< Apple, Cupertino >$ in our table would then provide the answer to our query.
The web contains millions of text pages with hidden data that would be best exploited in structured form. If we could build structured tables from the information hidden in unstructured text, then we would be able to run more complex queries and analysis of these tables, and report precise results. Based upon these ideas, Agichtein and Gravano developed the Snowball system for extracting structured data from plain-text documents with minimal human participation. Their techniques built on the ideas and general approach introduced by Brin.

In 1998, Brin[15] introduced the DIPRE method, where DIPRE means Dual Iterative Pattern Expansion to extract a structured *relation* (or *table*) from a collection of HTML documents. DIPRE worked best in an environment like the World-Wide Web, where the table,*tuples* to be extracted tend to appear in uniform contexts repeatedly in the collection documents (i.e., available HTML pages). DIPRE exploited this redundancy and inherent structure in the collection to extract the target relation with minimal training from a user. In fact, DIPRE required the user to just provide a handful of valid tuples of the target relation, with no other training. Hence, in this context DIPRE' s goal was to extract a table with all the organization-location tuples that appeared in a given document collection. Initially, DIPRE was provided with a handful of instances of valid organization-location pairs. For example, we may indicate that $< Apple, Cupertino >$ is a valid pair, meaning that Apple is an organization whose headquarters are located in Cupertino. Similarly, DIPRE was provided with a few other examples. In addition, the user could provide a general regular expression that the entities must match. For example, a potential organization value must match a regular expression

[A-Z0-9][A-Za-z0-9:; :0#!?;&]{4, 45}[A-Za-z0-9]

This regular expression says that an organization must begin either with a capital letter (e.g., Apple), or with a number, (e.g., 3Com), and be followed by four to 45 characters, ending in a letter or a number. This is all the training that DIPRE required from the user. After this initial training phase, DIPRE looked for instances of the example organizations and locations in the text documents.

Then, DIPRE examined the text surrounding the initial tuples. For example, DIPRE inspected the context surrounding Apple and Cupertino in, say, "computer servers at Apple' s headquarters in Cupertino" to construct a pattern "$< STRING1 >$ ' s headquarters in $< STRING2 >$" . Briefly, the algorithm represented an occurrence of a seed tuple as a seven-tuple: $< o, l, order, url, left, middle, right >$, where url is the URL of the source document in which $< o, l >$ was found,

[15]S. Brin.1999. *Extracting patterns and relations from the world wide web.* In: Lecture Notes in Computer Science. pp. 172-183

order is 1 if o appeared before l and 0 otherwise, and left, middle, and right are the parts of the context that surrounds the occurrence of $< o, l >$ in the document. A pattern (represented as a five tuple $< order, urlprefix, left, middle, right >$) was created by grouping together occurrences that all had equal middle string and order, and then setting the url-prefix, left, and right of the pattern to the longest common substrings of all the url, left, and right strings, respectively. The patterns were then filtered by requiring that each pattern was supported by more then one seed tuple, and that the fields urlprefix, left, middle, and right should all be non-empty.

Finally, after generating a number of patterns from the initial seed tuples, DIPRE scanned the available documents in search of segments of text that match the patterns. As a result of this process, DIPRE generated new tuples and used them as new "seeds". DIPRE then started the process all over again by searching for new tuples in the documents to identify new promising patterns.

It can be seen that unlike most machine-learning systems for information extraction, DIPRE required no training other than providing a handful of initial seed tuples and specifying the general pattern that the elements of the extracted tuples must match. By acquiring additional training examples automatically, DIPRE aimed at capturing most of the tuples mentioned in the collection. A key assumption behind this method is that the table to be extracted appeared redundantly in the document collection in question. As a result of this assumption, the patterns that DIPRE generated need not be overly general to capture every instance of an organization-location tuple. Instead, a more critical goal was to discard patterns that were not selective enough, and that may generate invalid tuples. In effect, a system based on the DIPRE method would perform reasonably well even if certain instances of a tuple were missed, as long as the system captured one such instance.

The Snowball systemdeveloped key components of the basic DIPRE method. The Snowball architecture followed the general DIPRE outline except that Snowball introduced key ideas that results in substantially better performance. More specifically, Snowball presented a novel technique to generate patterns and extract tuples from text documents. In addition, Snowball introduced a strategy for evaluating the quality of the patterns and the tuples that are generated in each iteration of the extraction process. Only tuples and patterns that were regarded as being "sufficiently reliable" would be kept by Snowball for the following iterations of the system. These new strategies for generation and filtering of patterns and tuples improved the quality of the extracted tables significantly.

Obviously, a crucial step in the table extraction process is the generation of patterns to be used for finding new tuples in the documents. Ideally, patterns needed both to be selective, so that they did not generate incorrect tuples, and to have high coverage, so that they could identify many new tuples.

Snowball was initially given a handful of example tuples. For every such organization-location tuple ¡o, l¿ Snowball found segments of text in the document collection where o and l occur close to each other, just as DIPRE did, and analyzed the text that "connects" o and l, to generate patterns. A key improvement of Snowball from the basic DIPRE method is that Snowball' s patterns included named-entity tags.

An example of such a pattern is: $< LOCATION > -based < ORGANIZATION >$. This pattern would not match any pair of strings connected by "-based." Instead, $< LOCATION >$ would only match a string identified by a tagger as an entity of type LOCATION. Similarly, $< ORGANIZATION >$ would only match a string identified by a tagger as an entity of type ORGANIZATION. To understand the impact of using named-entity tags in the Snowball patterns, consider the pattern $< STRING2 >$-based $< STRING1 >$. This pattern matched the text surrounding correct organization-location tuples (e.g., "the Armonk-based IBM has introduced..."). Unfortunately, this pattern would also match any strings connected by "-based," like "computer-based learning" or "alcohol-based solvents.". This might result in the inclusion of invalid tuples$< learning, computer >$ and $< solvents, alcohol >$ in the organization-location table. In contrast, by using the version of the same pattern that involved named-entity tags, $< LOCATION >$-based $< ORGANIZATION >$, it would result in a better chance of avoiding this kind of spurious matches. A key step in generating and later matching patterns, was finding where $< ORGANIZATION > and < LOCATION >$ entities occurred in the text. For this, Snowball used a state of the art named-entity tagger, The MITRE Corporation' s Alembic Workbench. In addition to ORGANIZATION and LOCATION entities, Alembic could identify PERSON entities, and could be trained to recognize other kinds of entities. Once the entities in the text documents were tagged, Snowball could ignore unwanted entities (e.g., PERSONS), focusing on occurrences of LOCATION and ORGANIZATION entities, and then analyzed the context that surrounded each pair of such entities to check if they were connected by the right words and hence match our patterns. To define patterns precisely, Snowball followed DIPRE' s approach, and have a pattern consist of a left, a middle, and a right string. An occurrence of an ORGANIZATION and a LOCATION entity would be regarded as a match for a pattern if the text surrounding the entities matches the three strings in the pattern exactly. This approach resulted in somewhat selective patterns (i.e., most of these patterns tended not to generate invalid tuples), yet it suffered from limited coverage (i.e., these patterns might not capture all instances of valid tuples). Hence, Snowball represented the context around the ORGANIZATION and LOCATION entities in the patterns in a more flexible way. As a result, minor variations such as an extra comma or a determiner would not stop the system from matching contexts that were otherwise very close to the patterns. More specifically, Snowball represented the left, middle, and right "contexts" associated with a pattern analogously as how the vector-space model of information retrieval represented documents and queries. Thus, the left, middle, and right contexts were three vectors associating weights (i.e., numbers between 0 and 1) with terms (i.e., arbitrary strings of non-space characters). These weights indicated the importance of each term in the corresponding context. An example of a Snowball pattern was the 5-tuple $< < the, 0.2 >, LOCATION, < -, 0.5 >, < based, 0.5 >, ORGANIZATION, \{\} >$. This pattern would match strings like "the Irving-based Exxon Corporation," where the word "the" (left context) preceded a location (Irving), which was in turn followed by the strings "-" and "based" (middle context) and an organization. What appeared to the right of the organization in the string was unimportant in this case, hence the empty right context in the pattern. Slight variations of the given string would also match the pattern to a smaller extent. For example, a string "...she said. Cupertino-based Apple reportedly..." would tend to match our example pattern, even when the location, Cupertino, was not preceded by any of the terms in the left context (i.e., "the "). This extra flexibility resulted in better coverage of the patterns. To match text portions with our 5-tuple representation of patterns, Snowball also associated a 5-tuple with each document portion that contained two named entities with the correct tag (i.e., LOCATION and ORGANIZATION in our scenario). After identifying two such entities in a string S, Snowball created three weight vectors l_S, r_S, and m_S from S by analyzing the left, right, and middle contexts around the named entities, respectively.l_S had a non-zero weight for each term in the w-term window to the left of the leftmost named entity in S, for some predefined window size w. Similarly, r_S had a non-zero weight for each term in the w-term window to the right of the rightmost named entity in S. Finally, m_S had a non-zero weight for each term in between the two named entities in S. The weight of a term in each vector was a function of the frequency of the term in the corresponding context. These vectors were scaled so their norm was one. Finally, they were multiplied by a scaling factor to indicate each vector's relative importance. Different experiments with English-language documents found that the middle context was the most indicative of the relationship between the elements of the tuple. Hence the system would assign the terms in the middle vector higher weights than the left and right vectors.

After extracting the 5-tuple representation of string S, Snowball matched it against the 5-tuple pattern by taking the inner product of the corresponding left, middle, and right vectors. In order to generate a pattern, Snowball grouped occurrences of known tuples in documents, if the contexts surrounding the tuples were "similar enough." More precisely, Snowball generated a 5-tuple for each string where a seed tuple occurs, and then clustered these 5-tuples using a simple single-pass clustering algorithm, and computed the similarity between the vectors, with minimum similarity threshold Tsim. The left vectors in the 5-tuples of clusters were represented by a centroid ls. Similarly, the middle and right vectors into ms and rs, respectively were collapsed. These three centroids, together with the original tags, formed a Snowball pattern $< l_s, t_1, m_s, t_2, r_s >$, which would be later used to find new tuples in the document collection. After generating patterns, Snowball scanned the collection to discover new tuples. Snowball first identified sentences that included an organization and a location, as determined by the named-entity tagger. For a given text segment, with an associated organization o and location l, Snowball generated the 5-tuple t $=< l_c; t_1; m_c; t_2; r_c >$. A candidate tuple $< o, l >$ was generated if there was a pattern t_p such that Match(t; $t_p$tp) sim, where sim was the clustering similarity threshold. Each candidate tuple would then have a number of patterns that helped generate it, each with an associated degree of match. Snowball used this information, together with information about the selectivity of the patterns, to decide what candidate tuples to actually add to the table that it is constructing. Generating good patterns is challenging. For example, we might generate a pattern $< , ORGANIZATION, < ",", 1 >, LOCATION, >$ from text occurrences like "Intel, Santa Clara, announced..." This pattern would be matched by any string that included an organization followed by a comma, followed by a location. Unfortunately, a sentence "It's a great time to invest in Apple, New York-based analyst Jane Smith said" would then generate a tuple $< Apple, NewYork >$, which would be incorrect because Apple's headquarters is in Cupertino. In summary, the pattern above was not selective, since it might generate incorrect tuples. Snowball would try to identify such patterns and not trust them, and instead focus on other more selective patterns. Under the redundancy assumption that tuples occur in different contexts in the collection, Snowball could afford to not use the less selective pattern above and still be able to extract the tuple $< Intel, SantaClara >$ through a different, more selective pattern. Estimating the selectivity of the patterns, so as to avoid trusting patterns that tend to generate wrong tuples, was one of the problems encountered. Snowball patterns could be based on their selectivity, and trust the tuples that they generate accordingly. Thus, a pattern that was not selective would have a low weight. The tuples generated by such a pattern would be discarded, unless they were supported by selective patterns.

The case for tuples was analogous. It was clear that not all of these tuples were not valid. For example, the tuple $< FruitJellies, Apple >$ was invalid, and was generated because Alembic incorrectly tagged "Apple" as a location and "Fruit Jellies" as an organization. So, if all of these tuples were used as new seeds tuples for the next Snowball iteration, it was possible to generate extraneous patterns that in turn might result in even more incorrect tuples in the next iteration. Different pruning schemes to select the new seed tuples were examined. Only tuples with high confidence were kept. The confidence of the tuple is a function of the selectivity and the number of the patterns that generated it. Intuitively, the confidence of a tuple will be high if it is generated by several highly selective patterns.

The pattern and tuple evaluation was the key part of Snowball, and was responsible for most of the improvement over the DIPRE scheme. As an initial filter, all patterns supported by fewer than Tsup seed tuples were eliminated. Alternative methods for defining Tsup were defined and it was concluded that a simple static value for Tsup worked well. n addition to the filter, the system computed the selectivity of each pattern based on the number of seed tuples that generated the patterns. In that step, the call to function UpdatePatternSelectivity checked each candidate tuple t $=< o, l >$ generated by the pattern in question. If there was a high confidence tuple $t_0 =< o, l_0 >$ generated during an earlier iteration of the system for the same organization o as in t, then this function compares locations l and l0. If the two locations were the same, then the tuple t was considered a positive match for the pattern. Otherwise, the match was negative. Intuitively, the candidate tuple that a pattern generated for the "known" organizations should match the locations of these organizations. Otherwise, the confidence in this pattern would be low. This confidence computation assumed that organization was a key for the relation that was been extracted. The confidence of a pattern P was:

$$Conf(P) = \frac{P.positive}{(P.positive + P.negative)}$$

where P.positive was the number of positive matches for P and P.negative is the number of negative matches.

As an example, consider the pattern P = $< ORGANIZATION >, < LOCATION >$. Assume that this pattern only matches the three lines of text below:

"**Exxon**,**Irving**, said"

"**Intel**, **Santa Clara**, cut prices"

"invest in **Apple**, **New York**-based analyst Jane Smith said"

The first two lines generated candidate tuples $< Exxon, Irving >$ and $< Intel, SantaClara >$, which we already knew from previous iterations of the system. The third line generated tuple$< Apple, NewYork >$. The location in this tuple conflicted with the location in tuple $< Apple, Cupertino >$, hence this last line was considered a negative example. Then, pattern P has confidence

$$Conf(P) = \frac{2}{2+1} = 66\%$$

This definition of confidence of a pattern is just one among many possibilities.

Having scored the patterns, it was possible to evaluate the new candidate tuples. For each tuple the set of patterns that produced it was stored, together with the measure of similarity between the context in which the tuple occurred, and the matching pattern. Consider a candidate tuple T and the set of patterns P =P_i that were used to generate T . Assume that T matched each of the patterns P_i perfectly, with degree of match equal to one. Assume the probability Prob(P_i) with which each pattern P_i generates valid tuples. If these probabilities were independent of each other, then the probability that T was valid, Prob(T), could be calculated as:

$$Prob(T) = 1 - Prob(AllPatternsEliminatedIncorrectly) = 1 - \prod_{j=0}^{|P|}(1 - Prob(Pi))$$

The confidence metric Conf (P_i) was designed to be a rough estimate of Prob(Pi), the probability of pattern P_i generating a valid tuple. It was also accounted for the cases where T has occurred in contexts that did not match our patterns perfectly. For this, each Conf (P_i) term was scaled by the degree of match of the corresponding

$$Conf(T) = 1 - \prod_{j=0}^{|P|}(1 - Conf(P_i) \cdot Match(C_i, P_i))$$

where P = Pi is the set of patterns that generated T and Ci is the context associated with an occurrence of T that matched Pi.

For example, suppose that a tuple ¡Netscape, Mountain View¿ had been generated using the patterns "$< ORGANIZATION >$, $< LOCATION >$" and "$< ORGANIZATION > of < LOCATION >$". These patterns had been found to have confidences of 0.5 and 0.6, which means that individually, these patterns were almost as likely to generate valid tuples as they were to generate invalid tuples. However, the confidence of the tuple that is generated by both of these patterns is:

$$Conf(T_new) = 1 - ((1 - 0.5) \cdot (1 - 0.6)) = 1 - 0.5 \cdot 0.4 = 0.8$$

When describing the calculation of the pattern confidence, any confidence values from previous iterations of Snowball were ignored. To control the learning rate of the system, the new confidence of the pattern was set as:

$$Conf(P) = Conf_{new}(P) \cdot W_{update} + Conf_{old}(P) \cdot (1 - W_{update})$$

The parameter Wupdate could be used to control the speed of learning from new examples. If $W_updat e < 0.5$ then the system in effect trusted new examples less on each iteration, which would lead to more conservative patterns and have a damping effect. For their experiment, Agitech and Gravano set $W_updat e = 0.5$. Similarly, already extracted tuples were often re-extracted. In this case, the new confidence of the tuple could be set as:

$$Conf(T) = Conf_{new}(T) \cdot W_{update} + Conf_{old}(T) \cdot (1 - W_{update})$$

After determining the confidence of the candidate tuples using the definition above, Snowball discarded all tuples with low confidence. These tuples could add noise into the pattern generation process, which would in turn introduce more invalid tuples, degrading the performance of the system. The set of tuples to use as the seed in the next Snowball iteration was then Seed = {T | Conf (T)> T_t}, where T_t was some pre specified threshold.

2.1.5 Pantel and Pennacchiotti's Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations

Espresso [16] is a general-purpose, broad, and accurate corpus harvesting algorithm requiring minimal supervision. The main algorithmic contribution to Espresso was a novel method for exploiting generic patterns which are broad coverage noisy patterns, i.e. patterns with high recall and low precision. Before Espresso, difficulties in using these patterns were a major impediment for minimally supervised algorithms resulting in either very low precision or recall. Pantel and Pennacchotti proposed a method to automatically detect generic patterns and to separate their correct and incorrect instances. The key assumption behind the algorithm was that given a set of reliable (high precision) patterns on a corpus, correct instances of a generic pattern will extract more with reliable patterns on a very large corpus, like the Web, than incorrect ones.

Espresso was based on the framework adopted by Hearst (1992). It was a minimally supervised bootstrapping algorithm that took as input a few seed instances of a particular relation and iteratively learned surface patterns to extract more instances. The key to Espresso lie in its use of generic patterns, that extracted both many correct and incorrect relation instances. For example, for part-of relations, the pattern "X of Y " extracted many correct relation instances like "wheel of the car " but also many incorrect ones like "house of representatives ". Espresso assumes that in very large corpora, like the Web, correct instances generated by a generic pattern would be instantiated by some reliable patterns, where reliable patterns are patterns with high precision but often very low recall (e.g., "X consists of Y " for part-of relations). Espresso iterates between the following three phases: pattern induction, pattern ranking/selection, and instance extraction. The algorithm begins with seed instances of a particular binary relation (e.g., is-a) and then iterates through the phases until it extracts $\tau 1$ patterns or until the average

[16]P. Pantel and M. Pennacchiotti. 2006. *Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations.* In: Proceedings of Conference on Computational Linguistics / Association for Computational Linguistics (COLING/ACL-06). pp. 113-120.

pattern score decreases by more than $\tau 2$ from the previous iteration. In their experiments, Pantel and Pennacchiotti set $\tau 1 = 5$ and $\tau 2 = 50\%$. For tokenization, in order to harvest multiword terms as relation instances, they adopted a slightly modified version of the term definition given by Justeson (1995)[17], one of the most commonly used in NLP literature:

$$((Adj|Noun) + |((Adj|Noun) * (NounPrep)?)(Adj|Noun)*)Noun$$

In the pattern induction phase, Espresso inferred a set of surface patterns P that connected as many of the seed instances as possible in a given corpus. Any pattern learning algorithm could have been used. They chose the state of the art algorithm described by Ravichandran and Hovy (2002)[18] with the following slight modification. For each input instance x, y, they first retrieved all sentences containing the two terms x and y. The sentences were then generalized into a set of new sentences Sx,y by replacing all terminological expressions by a terminological label, TR. For example:

"Because/IN HF/NNP is/VBZ a/DT weak/JJ acid/NN and/CC x is/VBZ a/DT y"

was generalized as:

"Because/IN **TR** is/VBZ a/DT **TR** and/CC **x** is/VBZ a/DT **y**"

Term generalization was useful for small corpora to ease data sparseness. As in the original algorithm, all substrings linking terms x and y were then extracted from Sx,y, and overall frequencies were computed to form P.

In Ravichandran and Hovy (2002), a frequency threshold on the patterns in P was set to select the final patterns. However, low frequency patterns might in fact

[17]J. Justeson. 1995. *Technical terminology: some linguistic properties and an algorithm for identification in text.*In:Natural Language Engineering. pp. 9-27

[18]D. Ravichandran, E. Hovy. 2002. *Learning Surface Patterns for a Question Answering System.* In: Proceedings of the ACL Conference

be very good. Espresso used, instead of frequency, a novel measure of pattern reliability, r_τ.

Espresso ranked all patterns in P according to reliability r_τ.and discarded all but the top-k, where k was set to the number of patterns from the previous iteration plus one. During Instance Extraction, Espresso retrieved from the corpus the set of instances I that matched any of the patterns in P. Next, Espresso filtered incorrect instances and then selected the highest scoring m instances, according to $r_?$, as input for the subsequent iteration. They experimentally set m=200.

In small corpora, the number of extracted instances may be too low to guarantee sufficient statistical evidence for the pattern discovery phase of the next iteration. In such cases, the system entered an expansion phase, where instances were expanded as follow.
Web expansion: New instances of the patterns in P were retrieved from the Web, using the Google search engine. Specifically, for each instance $x, y \in I$, the system created a set of queries, using each pattern in P instantiated with y.
Syntactic expansion: New instances were created from each instance $x, y \in I$ by extracting sub-terminological expressions from x corresponding to the syntactic head of terms. For example, the relation "new record of a criminal conviction part-of FBI report" expands to: "new record part-of FBI report", and "record part-of FBI report". Intuitively, a reliable pattern is one that is both highly precise and one that extracts many instances.

The recall of a pattern p could be approximated by the fraction of input instances that were extracted by p. Patterns should preferably be highly associated with the input instances. Pointwise mutual information (Cover and Thomas 1991)[19] was a commonly used metric for measuring this strength of association between two events x and y:

$$pmi(x, y) = \log \frac{P(x, y)}{P(x)P(y)}$$

[19]T. M. Cover and J. A. Thomas.1991. In: Elements of Information Theory

Pantel and Pennecchiotti defined the reliability of a pattern p, $r_\tau(p)$, as its average strength of association across each input instance i in I, weighted by the reliability of each instance i:

$$r_\tau(p) = \frac{\sum_{i \in I} \left(\frac{pmi(i,p)}{max_{pmi}} * r_i(i) \right)}{|I|}$$

where $r_\tau(i)$ is the reliability of instance i (defined below) and maxpmi is the maximum pointwise mutual information between all patterns and all instances. $r_\tau(p)$ ranged from [0,1]. The reliability of the manually supplied seed instances was $r_\tau(i) = 1$. The pointwise mutual information between instance i = x, y and pattern p was estimated using the following formula:

$$pmi(i,p) = \log \frac{|x,p,y|}{|x,*,y||*,p,*|}$$

where $|x, p, y|$ was the frequency of pattern p instantiated with terms x and y and where the asterisk (*) represented a wildcard. Estimating the reliability of an instance was similar to estimating the reliability of a pattern. Intuitively, a reliable instance was one that was highly associated with as many reliable patterns as possible. Hence, analogous to their pattern reliability measure, they defined the reliability of an instance i, $r_\tau(i)$, as:

$$r_\tau(p) = \frac{\sum_{p \in P^t} \frac{pmi(i,p)}{max_{pmi}} * r_\tau(P)}{|P|}$$

where r(p) was the reliability of pattern p (defined earlier) and maxpmi was as defined above. Note that r(i) and r(p) were recursively defined, where r(i) = 1 for the manually supplied seed instances. Using generic patterns blindly increased system recall while dramatically reducing precision. Minimally supervised algorithms had typically ignored such patterns for this reason. Espresso's recall could be significantly increased by automatically separating correct instances extracted by generic patterns from incorrect ones. The challenge was to harness the expressive power of the generic patterns while remaining minimally supervised. The intuition behind this method was that in a very large corpus, like the Web, correct instances of a generic pattern will be instantiated by many of Espresso's reliable patterns accepted in P. : by definition, Espresso' s reliable patterns extract instances with high precision (yet often low recall). In a very large corpus, like the Web, it was assumed that a correct instance would occur in at least one of Espresso' s reliable pattern even though the pattern's recall was low. Intuitively, the confidence in a correct instance increased when, i) the instance was associated with many reliable patterns; and ii) its association with the reliable patterns was high. At a given Espresso iteration, where PR represented the set of previously selected reliable patterns, this intuition was captured by the following measure of confidence in an instance i = x, y:

$$S(i) = \sum_{p \in P_R} S_p(i) \times \frac{r_\tau(p)}{T}$$

where T is the sum of the reliability scores $r_\tau(p)$ for each pattern $p \in PR$, and

$$S_p(i) = pmi(i,p) = \log \frac{|x,p,y|}{|x,*,y||*,p,*|}$$

where pointwise mutual information between instance i and pattern p was estimated with Google as follows:

$$S_p(i) \approx \frac{|x.p.y|}{|x| \times |y| \times |p|}$$

An instance i was rejected if S(i) was smaller than some threshold τ. Although this filtering might also be applied to reliable patterns, they found this to be true in their experiments since most instances generated by reliable patterns were correct. In Espresso, a pattern was classified as generic when it generated more than 10 times the instances of previously accepted reliable patterns. To evaluate their algorithm, Pantel and Pennacchiotti built two different dataset: TREC consisted of a sample of articles from the Aquaint (TREC-9) newswire text collection: the sample consisted of 5,951,432 words. CHEM was a small dataset of 313,590 words and consisted of a college level textbook of introductory chemistry (Brown et al. 2003). Each corpus was pre-processed using the Alembic Workbench POS Tagger.

Espresso was designed to extract various semantic relations exemplified by a given small set of seed instances. The systems used in the evaluation were the following:

- RH02: The algorithm by Ravichandran and Hovy (2002)
- GI03: The algorithm by Girju et al. (2006) [20]
- PR04: The algorithm by Pantel and Ravichandran (2004) [21]
- ESP-: The Espresso algorithm using the pattern and instance reliability measures, but without using generic patterns.
- ESP+: The full Espresso algorithm exploiting generic patterns. For ESP+, they experimentally set τ to $\tau = 0.4$ for TREC and $\tau = 0.3$ for CHEM

The relations considered were the standard is-a and part-of, as well as the following:

- succession: This relation indicated that a person succeeded another in a position or title. For example, George Bush succeeded Bill Clinton. This relation was evaluated on the TREC-9 corpus.
- reaction: This relation occurred between chemical elements/molecules that could be combined in a chemical reaction. For example, hydrogen gas reacts-with oxygen gas. This relation was evaluated on the CHEM corpus.
- production: This relation occurred when a process or element/object produced a result.

For example, ammonia produced nitric oxide. This relation was also evaluated on the CHEM corpus. For each semantic relation, a small set of seed examples was extracted. The seeds were used for both Espresso as well as RH02.

For each output set, per relation, the precision of the system was evaluated by extracting a random sample of instances (50 for the TREC corpus and 20 for the

[20] R. Girju, A. Badulescu, D. Moldovan. 2006. *Automatic discovery of part-whole relations.* In: Computational Linguistics. pp. 83-135

[21] P. Pantel, D. Ravichandran. 2004. *Automatically labeling semantic classes.* In: Proceedings of HLT/NAACL

CHEM corpus) and evaluating their quality manually using two human judges (a total of 680 instances were annotated per judge). For each instance, judges might assign a score of 1 for correct, 0 for incorrect, and 1/2 for partially correct. Example instances that were judged partially correct include “analyst is-a manager” and “pilot is-a teacher”. The precision for a given set of instances is the sum of the judges’ scores divided by the total instances. Experimental results, for all relations and the two different corpus sizes, showed that ESP- greatly outperformed the other methods on precision. However, without the use of generic patterns, the ESP- system showed lower recall in all but the production relation. ESP+ showed one to two orders of magnitude improvement on recall while losing on average below 10% precision. The succession relation was the only relation where Espresso found no generic pattern. For other relations, Espresso found from one to five generic patterns. In order to better analyze their use of generic patterns, the authors performed the following experiment. For each relation, they randomly sampled 100 instances for each generic pattern and built a gold standard for every generic pattern (by manually tagging each instance as correct or incorrect). They then sorted the 100 instances according to the scoring formula S(i) and computed the average precision, recall, and F-score of each top-K ranked instances for each pattern.They then discovered that recall climbed at a much faster rate than precision decreased, indicating that the scoring function effectively separated correct and incorrect instances.

According to the authors, an interesting avenue of future work would be to automatically determine the proper threshold for each individual generic pattern instead of setting a uniform threshold.

2.1.6 Turney's way of expressing semantic relations without supervision

In 2006 Turney[22] presented an unsupervised learning algorithm for mining large text corpora for patterns that express implicit semantic relations.

Turney' s work is based on Hearst's and Bernland & Charniak's, but he considered the inverse kind of problem, because he could mine a large text corpus for lexico-synctactic patterns that expressed the implicit relation between a couple of word X:Y. In his experiments, he used a corpus of web pages containing about 5×1010 English words. From co-occurrences of the pair ostrich:bird in this corpus, he could generate 516 patterns of the form "X ... Y" and 452 of the form "Y ... X" . Most of these patterns were not very useful for text mining. The main challenge was finding a way of ranking the patterns, so that patterns like "Y such as the X " could be highly ranked. Another challenge was to find a way to empirically evaluate the performance of any such pattern algorithm.

For a given input word pair X:Y with some unspecified semantic relations, he ranked the corresponding output list of patterns$\langle P_i, ..., P_n \rangle$ in order of decreasing pertinence. The pertinence of a pattern P_i for a word pair X:Y was the expected relational similarity between the given pair and typical pairs that fitted Pi. To calculate pertinence, is necessary to measure relational similarity. Turney used Latent Relational Analysis (Turney, 2005)[23] for this task. Given a word pair X:Y, the algorithm should rank the corresponding list of patterns$\langle Pi, ..., Pn \rangle$ according to their value for mining text, in support of semantic network construction and similar tasks. Therefore, his experiments were based on two tasks that provided objective performance measures.

The relational similarity between two pairs of words $X_1 : Y_1$ and $X_2 : Y_2$, is the degree to which their semantic relations are analogous. For example, mason:stone and carpenter:wood have a high degree of relational similarity. Let

[22]P.D. Turney. 2006. *Expressing implicit semantic relations without supervision.* In: Annual Meeting-Association for computational linguistics

[23]P. D. Turney. 2005. *Measuring semantic similarity by latent relational analysis.* In International joint conference on artificial intelligence

$W = X_1 : Y_1, ..., X_n : Y_n$ be a set of word pairs and let $P = P_1, ..., P_m$ be a set of patterns. The pertinence of pattern P_i to a word pair $X_j : Y_j$ is the expected relational similarity between a word pair $X_k : Y_k$, randomly selected from W according to the probability distribution$p(X_k : Y_k|P_i)$, and the word pair $X_j : Y_j$:

$$pertinence(X_j : Y_j, P_i) = \sum p(X_k : Y_k|P_i) \cdot sim_r(X_j : Y_j, X_k : Y_k)$$

The conditional probability $p(X_k : Y_k|P_i)$ can be interpreted as the degree to which the pair $X_k : Y_k$ is representative of pairs that fit the pattern P_i. That is, P_i is pertinent to $X_j : Y_j$ if highly typical word pairs $X_k : Y_k$ for the pattern Pi tend to be relationally similar to $X_j : Y_j$.

Pertinence tends to be highest with patterns that are unambiguous. The maximum value of pertinence $(X_j : Y_j, P_i)$ is attained when the pair $X_j : Y_j$ belongs to a cluster of highly similar pairs and the conditional probability distribution $p(X_k : Y_k|P_i)$ is concentrated on the cluster. An ambiguous pattern, with its probability spread over multiple clusters, will have less pertinence. If a pattern with high pertinence is used for text mining, it will tend to produce word pairs that are very similar to the given word pair; this follows from the definition of pertinence. Turney believed that the previous definition is the first formal measure of quality for text mining patterns. If $f_{k,i}$ is the number of occurrences in a corpus of the word pair $X_k : Y_k$ with the pattern $P_i, p(X_k : Y_k|P_i)$ could be estimated as follows:

$$p(X_k : Y_k|P_i) = f_{k,i} / \sum_{j=1}^{n} f_{j,i}$$

Instead , Turney first estimated $p(P_i|X_k : Y_k)$:

$$p(P_i|X_k : Y_k) = f_{k,i} / \sum_{j=1}^{m} f_{k,j}$$

And then applied Bayes' Theorem:

$$p(X_k : Y_k|P_i) = \frac{p(X_k : Y_k) \cdot p(P_i|X_k : Y_k)}{\sum_{j=1}^{n}(X_j : Y_j) \cdot p(P_i|X_j : Y_j)}$$

Assuming $p(X_j : Y_j) = 1/n$ for all pairs in W:

$$p(X_k : Y_k|P_i) = \frac{p(P_i|X_k : Y_k)}{\sum_{j=1}^{n} p(P_i|X_j : Y_j)}$$

The use of Bayes' Theorem and the assumption that $p(X_j : Y_j) = 1/n$ for all word pairs is a way of smoothing the probability $p(X_k : Y_k|P_i)$, similar to Laplace smoothing.

Turney' s algorithm was the first unsupervised learning algorithm that could find patterns for semantic relations, given only a large corpus (in their experiment $5 \cdot 1010$ words) and a moderately sized set of word pairs (e.g., 600 or more pairs in the experiments), such that the members of each pair appeared together frequently in short phrases in the corpus. The word pairs were not seeds, since the algorithm did not require the pairs to be labeled or grouped; the words were not assumed to be homogeneous.

The needed word pairs could be generated automatically, by searching for word pairs that co-occur frequently in the corpus. However, both Turney' s evaluation methods involved a predetermined list of word pairs. If the algorithm were allowed to generate its own word pairs, the overlap with the predetermined lists would likely be small. This is a limitation of the evaluation methods rather than the algorithm.

Since any two word pairs might have some relations in common and some that are not shared, the algorithm generated a unique list of patterns for each input word pair. For example, mason:stone and carpenter:wood shared the pattern "X carves Y" , but the patterns "X nails Y" and "X bends Y" were unique to carpenter:wood. The ranked list of patterns for a word pair X :Y gave the relations between X and

Y in the corpus, sorted with the most pertinent (i.e., characteristic, distinctive, unambiguous) relations first.

As previously mentioned Turney developed an algorithm called Latent Relational Analysis (LRA) for measuring the relational similarity between two pairs of words. This algorithm can be used to solve multiple choice word analogy questions and to classify noun-modifier pairs (Turney, 2005), but it does not attempt to express the implicit semantic relations.

Turney mapped each pair X :Y to a high-dimensional vector v . The value of each element vi in v was based on the frequency, for the pair X :Y , of a corresponding pattern Pi . The relational similarity between two pairs, $X_1 : Y_1$ and $X_2 : Y_2$, was derived from the cosine of the angle between their two vectors. A limitation of this approach was that the semantic content of the vectors is difficult to interpret; the magnitude of an element vi not being a good indicator of how well the corresponding pattern Pi expresses a relation of X :Y.

Pertinence built on the measure of relational similarity, has the advantage that the semantic content could be interpreted; it could be used to point to specific patterns and confirm the expression of implicit relations. Furthermore, patterns could be used to find other pairs with the same relations.

Hearst (1992) processed her text with a part- of- speech tagger and a unification-based constituent analyzer. This made it possible to use more general patterns. For example, instead of the literal string pattern "Y such as the X" , where X and Y are words, Hearst used the more abstract pattern "NP0 such as NP1 " , where NP_i, represents a noun phrase. For the sake of simplicity, Turney avoided part-of-speech tagging, which would have limited him to literal patterns.

Turney' s algorithm was composed of eleven steps. It took as input a set of word pairs $WX_1 : Y_1, , X_n : Y_n =$ and produced ranked lists of patterns $P_1, , P_m$ for each input pair as output. Every following step was similar to the algorithm of Turney (2005), with several changes to support the calculation of pertinence.

1. **Find phrases:** For each pair X_i : Y_i , it made a list of phrases in the corpus that contained the pair. The Waterloo MultiText System (Clarke et al., 1998)[24] was used to search in a corpus of about 51010 English words (Terra and Clarke, 2003)[25]. The algorithm made one list of phrases that begin with X_i and end with Y_i and a second list for the opposite order. Each phrase must have one to three intervening words between X_i and Y_i . The first and last words in the phrase did not need to exactly match X_i and Y_i . The MultiText query language allowed different suffixes. Veale (2004) observed that it was easier to identify semantic relations between nouns than between other parts of speech. Therefore Turney used WordNet 2.0 (Miller, 1995) to guess whether X_i and Y_i were likely to be nouns. When they were nouns, the only suffixes' variation allowed was pluralization. For all other parts of speech, the algorithm was pretty liberal about suffixes. For example, an adjective such as "inflated" was allowed to match a noun such as "inflation" . With MultiText, the query "inflat*" matched both "inflated" and "inflation".

2. **Generate patterns:** For each list of phrases, the algorithm generated a list of patterns, based on the phrases. It replaced the first word in each phrase with the generic marker "X" and replaced the last word with "Y" . The intervening words in each phrase might be either left as they were, or replaced with the wildcard "*" . For example, the phrase "carpenter nails the wood" yielded the patterns "X nails the Y" , "X nails * Y" , "X * the Y" , and "X * * Y" . The algorithm did not allow duplicate patterns in a list, but noted the number of times a pattern was generated for each word pair X_i : Y_i in each order (X_i first and Y_i last or vice verse). We call this the pattern frequency. It was a local frequency count, analogous to term frequency in information retrieval.

[24]C.L.A. Clarke, G.V Cormack, C.R. Palmer. 1998. *An overview of MultiText.* In: ACM SIGIR Forum

[25]E.Terra, C.L.A. Clarke. 2004. *Scoring missing terms in information retrieval task.* In: Proceedings of the thirteenth ACM international conference

3. **Count pair frequency:** The pair frequency for a pattern was the number of lists from the preceding step that contained the given pattern. It was a global frequency count, analogous to document frequency in information retrieval. A pair $X_i : Y_i$ yielded two lists of phrases and hence two lists of patterns. A given pattern might appear in zero, one, or two of the lists for $X_i : Y_i$.

4. **Map pairs to rows:** In preparation for building a matrix X ,the algorithm needed to create a mapping of word pairs to row numbers. For each pair $X_i : Y_i$, it created a row for $X_i : Y_i$ and another row for $Y_i : X_i$. If W did not already contain $Y_1 : X_1, , Y_n : X_n$, then the number of word pairs was effectively doubled, which increased the sample size for calculating pertinence.

5. **Map patterns to columns:** The algorithm created a mapping of patterns to column numbers. For each unique pattern of the form "X ... Y" from Step 2, it created a column for the original pattern "X ... Y" and another column for the same pattern with X and Y swapped, "Y ... X" . Step 2 could generate millions of distinct patterns. This section resulted in 1.706.845 distinct patterns, yielding 3,413,690 columns., which were too many columns for matrix operations with standard desktop computer at the time. Most of the patterns had a very low pair frequency, 1.371.702 had a pair frequency of one. To keep the matrix X manageable, all patterns with a pair frequency less than ten, were dropped. This left 42,032 patterns, yielding 84,064 columns. Turney (2005) limited the matrix to 8,000 columns, but a larger pool of patterns was better for his purposes, since it increased the likelihood of finding good patterns for expressing the semantic relations of a given word pair.

6. **Build a sparse matrix:** The algorithm built a matrix X in sparse matrix format. The value for the cell in row i and column j was the pattern frequency of the j-th pattern for the the i-th word pair.

7. **Calculate entropy:** The algorithm applied log and entropy transformations to the sparse matrix X (Landauer and Dumais, 1997)[26]. Each cell was replaced with its logarithm, multiplied by a weight based on the negative entropy of the corresponding column vector in the matrix. This gave more weight to patterns that varied substantially in frequency for each pair.

8. **Apply SVD:** After log and entropy transforms, the algorithm applied the Singular Value Decomposition (SVD) to X (Golub and Van Loan, 1996)[27]. SVD decomposed X into a product of three matrices USVT , where U and V were in column orthonormal form (i.e., the columns are orthogonal and have unit length) and S was a diagonal matrix of singular values (hence SVD). If X was of rank r , then S was also of rank r . Let S_k , where $k < r$, be the diagonal matrix formed from the top k singular values, and let U_k and V_k be the matrices produced by selecting the corresponding columns from U and V . The matrix T $U_k S_k V_k$ was the matrix of rank k that best approximated the original matrix X , in the sense that it minimized the approximation errors (Golub and Van Loan, 1996). Following Landauer and Dumais (1997), Turney used k = 300 . This matrix T $U_k S_k V_k$ was a smoothed version of the original matrix. SVD was used to reduce noise and compensate for sparseness (Landauer and Dumais, 1997).

9. **Calculate cosines:**The relational similarity between two pairs, simr $(X_1 : Y_1, X_2 : Y_2)$, was given by the cosine of the angle between their corresponding row vectors in the matrix T $U_k S_k V_k$ (Turney, 2005). The cosine function $\cos x$ is one of the basic functions encountered in trigonometry (the others being the co-secant, cotangent, secant, sine, and tangent). Let θ be an angle measured counterclockwise from the x-axis along the arc of the unit circle. Then $\cos\theta$ is the horizontal coordinate of the arc endpoint. To calculate pertinence, the relational similarity between all possible pairs of pairs would

[26]T.K. Landauer, S.T. Dumais. 1997. *A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge* In: Psychological Review

[27]G.H. Golub, C.F. Van Loan. 1996. Matrix Computations

be needed. All of the cosines could be efficiently derived from the matrix T $U_kS_k(U_kS_k)$ (Landauer and Dumais, 1997).

10. **Calculate conditional probabilities:** Using Bayes' Theorem and the raw frequency data in the matrix X from Step 6, before log and entropy transforms, the algorithm calculated the conditional probability $p(X_i : Y_iP_j)$ for every row (word pair) and every column (pattern).

11. **Calculate pertinence:** With the cosines from Step 9 and the conditional probabilities from Step 10, the algorithm calculated pertinence$(X_i : Y_i, P_j)$ for every row $X_i : Y_i$ and every column P_j for which $p(X_i : Y_iP_j) > 0$. When $p(X_i : Y_iP_j) = 0$, it was possible that pertinence $(X_i : Y_i, P_j) > 0$, but Turney avoided calculating pertinence in these cases for two reasons. Firstly, this speeds up computation, because X was sparse, so $p(X_i : Y_iP_j) = 0$ for most rows and columns. Secondly, $p(X_i : Y_iP_j) = 0$ implied that the pattern P_j did not actually appear with the word pair $X_i : Y_i$ in the corpus; the pattern was just estimated to be appropriate for the word pair, and could not be. Therefore it was preferable to analyze just patterns and word pairs that have actually been observed in the corpus. For each pair $X_i : Y_i$ in W, two separate ranked lists were created, one for patterns of the form "X Y" and another for patterns of the form "Y X" , where the patterns in both lists were sorted in order of decreasing pertinence to $X_i : Y_i$. Ranking served as a kind of normalization. The relative rank of a pattern was more reliable as an indicator of its importance than the absolute pertinence. This was analogous to what happens in information retrieval, where documents are ranked in order of their relevance to a query. The relative rank of a document is more important than its actual numerical score (which is usually hidden from the user of a search engine). Having two separate ranked lists helped to avoid bias. For example, ostrich:bird generates 516 patterns of the form "X ... Y" and 452 patterns of the form "Y ... X" . Since there were more patterns of the form "X ... Y" , there was a slight bias towards these patterns. If the two lists were merged, the "Y ... X" patterns would be at a disadvantage.

Turney's algorithm was inspired by Word Space Model, which is a computational model of word meaning that utilizes the distributional patterns of words collected over large text data to represent semantic similarity between words in terms of spatial proximity.

Pertinence has been evaluated in two different kinds of experiment.

Experiments with word analogy tested pertinence using 374 college-level multiple-choice word analogies taken from the SAT test. For each question, there was a target word pair, called the stem pair, and five choice pairs. The task was to find the choice that was most analogous (i.e. has the highest relational similarity) to the stem. This choice pair was called the solution and the other choices were distractors. Since there were six word pairs per question, there were $374 \times 6 = 2244$ pairs in the input set W. In step 4 of the algorithm the pairs were doubled, but some pairs were also dropped because they did not co-occur in the corpus. To answer a SAT question, the algorithm generated ranked lists of patterns for each of the six word pairs. Each choice was evaluated by taking the intersection of its patterns with the stem' s patterns.
The shared patterns were scored by the average of their rank in the stem' s lists and the choice' s lists.
Since the lists were sorted in order of decreasing pertinence, a low score meant a high pertinence.
The choice with the lowest scoring shared pattern was the selected one. The following table shows three examples, two questions that were answered correctly followed by one that was answered incorrectly. The correct answers were reported in bold font. For the first question, the stem is ostrich:bird and the best choice is (a) lion:cat. The highest ranking pattern that was shared by both of these pairs is "Y such as the X" . The third question illustrates that, even when the answer was incorrect, the best shared pattern ("Y powered * * X") might be plausible.

	Word pair	Best shared pattern	Score
1.	ostrich:bird		
(a)	**lion:cat**	**"*Y* such as the *X*"**	**1.0**
(b)	goose:flock	"*X* * * breeding *Y*"	43.5
(c)	ewe:sheep	"*X* are the only *Y*"	13.5
(d)	cub:bear	"*Y* are called *X*"	29.0
(e)	primate:monkey	"*Y* is the * *X*"	80.0
2.	traffic:street		
(a)	ship:gangplank	"*X* * down the *Y*"	53.0
(b)	crop:harvest	"*X* * adjacent * *Y*"	248.0
(c)	car:garage	"*X* * a residential *Y*"	63.0
(d)	pedestrians:feet	"*Y* * accommodate *X*"	23.0
(e)	**water:riverbed**	**"*Y* that carry *X*"**	**17.0**
3.	locomotive:train		
(a)	horse:saddle	"*X* carrying * *Y*"	82.0
(b)	**tractor:plow**	**"*X* pulled * *Y*"**	**7.0**
(c)	rudder:rowboat	"*Y* * *X*"	319.0
(d)	camel:desert	"*Y* with two *X*"	43.0
(e)	gasoline:automobile	"*Y* powered * * *X*"	5.0

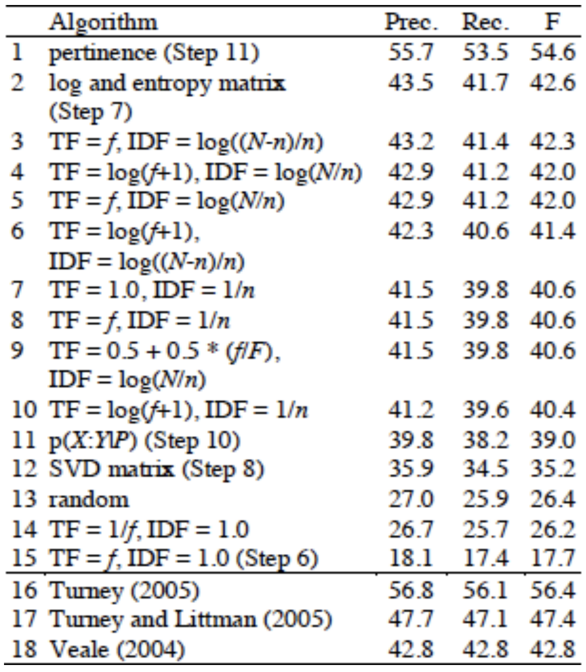
The following table shows the four highest ranking patterns for the stem and solution for the first example. The pattern "X lion Y" was anomalous, but the other patterns seemed reasonable. The shared pattern "Y such as the X" was ranked 1 for both pairs, hence the average score for this pattern was 1.0, as shown in the previous table. Note that the "ostrich is the largest bird" and "lions are large cats" , but the largest cat was the Siberian tiger.

Word pair	"*X* ... *Y*"	"*Y* ... *X*"
ostrich:bird	"*X* is the largest *Y*"	**"*Y* such as the *X*"**
	"*X* is * largest *Y*"	"*Y* such * the *X*"
lion:cat	"*X* lion *Y*"	**"*Y* such as the *X*"**
	"*X* are large *Y*"	"*Y* and mountain *X*"

The following table lists the top five pairs in W that matched the pattern "Y such as the X" . The pairs were sorted by p(X :Y P) . The pattern "Y such as the X" was one of 146 patterns shared by ostrich: bird and lion:cat. Most of these shared patterns were not very informative.

Word pair	Conditional probability
heart:organ	0.49342
dodo:bird	0.08888
elbow:joint	0.06385
ostrich:bird	0.05774
semaphore:signal	0.03741

In the following table, ranking patterns are compared by pertinence to ranking by various other measures, mostly based on variations of tf-idf (term frequency times inverse document frequency, a common way to rank documents in information retrieval). The tf-idf measures were taken from Salton and Buckley (1988)[28].



	Algorithm	Prec.	Rec.	F
1	pertinence (Step 11)	55.7	53.5	54.6
2	log and entropy matrix (Step 7)	43.5	41.7	42.6
3	TF = f, IDF = log((N-n)/n)	43.2	41.4	42.3
4	TF = log(f+1), IDF = log(N/n)	42.9	41.2	42.0
5	TF = f, IDF = log(N/n)	42.9	41.2	42.0
6	TF = log(f+1), IDF = log((N-n)/n)	42.3	40.6	41.4
7	TF = 1.0, IDF = 1/n	41.5	39.8	40.6
8	TF = f, IDF = 1/n	41.5	39.8	40.6
9	TF = 0.5 + 0.5 * (f/F), IDF = log(N/n)	41.5	39.8	40.6
10	TF = log(f+1), IDF = 1/n	41.2	39.6	40.4
11	p(X:Y|P) (Step 10)	39.8	38.2	39.0
12	SVD matrix (Step 8)	35.9	34.5	35.2
13	random	27.0	25.9	26.4
14	TF = 1/f, IDF = 1.0	26.7	25.7	26.2
15	TF = f, IDF = 1.0 (Step 6)	18.1	17.4	17.7
16	Turney (2005)	56.8	56.1	56.4
17	Turney and Littman (2005)	47.7	47.1	47.4
18	Veale (2004)	42.8	42.8	42.8

All of the pattern ranking algorithms were given exactly the same sets of patterns to rank. Any differences in performance were due to the ranking method alone. The algorithms might skip questions when the word pairs did not co-occur in the corpus. All of the ranking algorithms skipped the same set of 15 of the 374 SAT questions. Precision was defined as the percentage of correct answers out of the questions that were answered (not skipped). Recall was the percentage of correct answers out of the maximum possible number correct (374). The F measure was the harmonic mean of precision and recall.

For the tf-idf methods in the previous tables, f was the pattern frequency, n is the pair frequency, F was the maximum f for all patterns for the given word pair, and N was the total number of word pairs. By "TF = f, IDF = 1/ n " , for example (row 8), Turney means that f played a role that was analogous to term frequency and 1/ n played a role that was analogous to inverse document frequency. That

[28]G.Salton, C. Buckley. 1988. *Term-weighting approaches in automatic retrieval.* In: Information and Processing and management

is, in row 8, the patterns were ranked in decreasing order of pattern frequency divided by pair frequency.

The previous table also shows some ranking methods based on intermediate calculations in the algorithm. For example, row 2 gives the results when patterns are ranked in order of decreasing values in the corresponding cells of the matrix X from Step 7.

Row 12 shows the results Turney would get using Latent Relational Analysis (Turney, 2005) to rank patterns. The results in row 12 support the claim that LRA is not suitable for ranking patterns, although it works well for answering the SAT questions. The vectors in LRA yielded a good measure of relational similarity, but the magnitude of the value of a specific element in a vector was not a good indicator of the quality of the corresponding pattern.

The best method for ranking patterns was pertinence (row 1 in previous table). As a point of comparison, the performance of the average senior high school student on the SAT analogies was about 57%. The second best method was to use the values in the matrix X after the log and entropy transformations in Step 7 (row 2). The difference between these two methods is statistically significant with 95% confidence.

Pertinence (row 1) performed slightly below Latent Relational Analysis, but the difference was not significant.

In experiments with noun modifiers, pertinence was evaluated on the task of classifying noun-modifier pairs. The problem was to classify a noun-modifier pair, such as “flu virus” , according to the semantic relation between the head noun (virus) and the modifier (flu). For example, “flu virus” was classified as a causality relation (the flu is caused by a virus). For these experiments, a set of 600 manually labeled noun-modifier pairs was used(Nastase and Szpakowicz, 2003)[29]. There were five general classes of labels with thirty subclasses. Results were obtained with five classes; the results with thirty subclasses followed the same trends (that is, pertinence performed significantly better than the other ranking methods). The five classes were causality (storm cloud), temporality (daily exercise), spatial (desert

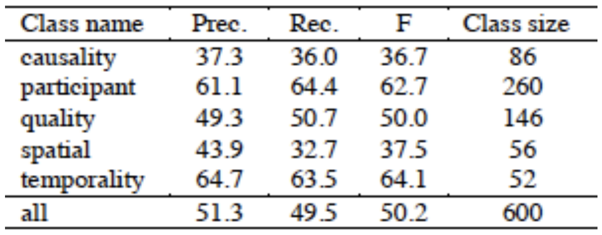
[29]V. Nastase, S. Szpakowicz. 2003. *Exploring noun-modifier sematic relations.* In: 5th International Workshop on Computational Semantics

storm), participant (student protest), and quality (expensive book).

The input set W consisted of the 600 noun-modifier pairs. This set is doubled in Step 4, but some pairs had been dropped because they did not co-occur in the corpus, leaving 1184 rows in the matrix. There were 16,849 distinct patterns with a pair frequency of ten or more, resulting in 33,698 columns. The matrix density was 2.57% .

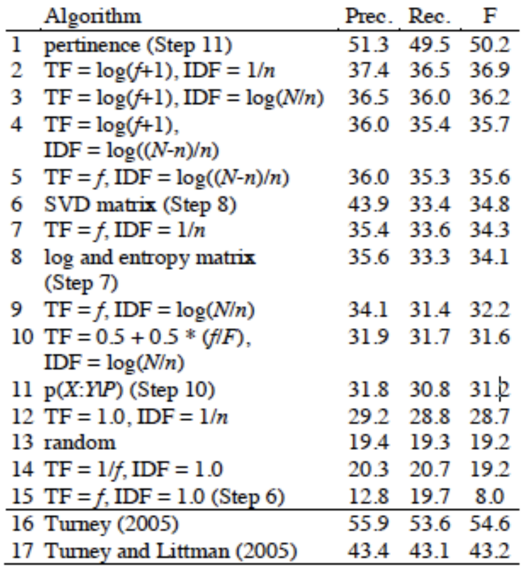
To classify a noun-modifier pair, Turney used a single nearest neighbor algorithm with leave one-out cross-validation. The set was split 600 times.

Each pair got a turn as the single testing example, while the other 599 pairs served as training examples. The testing example was classified according to the label of its nearest neighbor in the training set. The distance between two noun-modifier pairs was measured by the average rank of their best shared pattern. The following table shows the resulting precision, recall, and F, when ranking patterns by pertinence.



Class name	Prec.	Rec.	F	Class size
causality	37.3	36.0	36.7	86
participant	61.1	64.4	62.7	260
quality	49.3	50.7	50.0	146
spatial	43.9	32.7	37.5	56
temporality	64.7	63.5	64.1	52
all	51.3	49.5	50.2	600

The table on the next page shows the performance of pertinence on the noun-modifier problem, compared to various other pattern ranking methods. The bottom two rows were included for comparison; they were not pattern ranking algorithms. The best method for ranking patterns is pertinence. The difference between pertinence and the second best ranking method (row 2) was statistically significant with 95% confidence. Latent Relational Analysis performed slightly better than pertinence (row 1), but the difference was not statistically significant. Row 6 shows the results that was obtained using Latent Relational Analysis to rank patterns. Again, the results supported the claim that LRA was not suitable for ranking patterns. LRA could classify the noun modifiers (as we see in row 16), but it cannot express the implicit semantic relations that showed that an unlabeled noun-modifier in the testing set is similar to its nearest neighbor in the training set.



	Algorithm	Prec.	Rec.	F
1	pertinence (Step 11)	51.3	49.5	50.2
2	TF = log(*f*+1), IDF = 1/*n*	37.4	36.5	36.9
3	TF = log(*f*+1), IDF = log(*N*/*n*)	36.5	36.0	36.2
4	TF = log(*f*+1), IDF = log((*N*-*n*)/*n*)	36.0	35.4	35.7
5	TF = *f*, IDF = log((*N*-*n*)/*n*)	36.0	35.3	35.6
6	SVD matrix (Step 8)	43.9	33.4	34.8
7	TF = *f*, IDF = 1/*n*	35.4	33.6	34.3
8	log and entropy matrix (Step 7)	35.6	33.3	34.1
9	TF = *f*, IDF = log(*N*/*n*)	34.1	31.4	32.2
10	TF = 0.5 + 0.5 * (*f*/*F*), IDF = log(*N*/*n*)	31.9	31.7	31.6
11	p(*X*:*Y*|*P*) (Step 10)	31.8	30.8	31.2
12	TF = 1.0, IDF = 1/*n*	29.2	28.8	28.7
13	random	19.4	19.3	19.2
14	TF = 1/*f*, IDF = 1.0	20.3	20.7	19.2
15	TF = *f*, IDF = 1.0 (Step 6)	12.8	19.7	8.0
16	Turney (2005)	55.9	53.6	54.6
17	Turney and Littman (2005)	43.4	43.1	43.2

2.2 Relation Recognition

Recognizing and classifying entities and relations in text data is a key task in many NLP problems such as information extraction (IE) (Califf and Mooney, 1999; Freitag, 2000; Roth and Yih, 2001), question answering (QA) (Voorhees, 2000) and story comprehension (Hirschman et al., 1999).

In a typical IE application of constructing a jobs database from unstructured text, the system has to extract meaningful entities like title and salary and, ideally, to determine whether the entities are associated with the same position.

In a QA system, many questions ask for specific entities involved in some relations. For example, the question “Where was Poe born?” in TREC-9 asks for the location entity in which Poe was born. The question “Who killed Lee Harvey Oswald?” seeks a person entity that has the relation kill with the person Lee Harvey Oswald.

In all earlier works we know of, the tasks of identifying entities and relations were treated as separate problems. The common procedure is to first identify and classify entities using a named entity recognizer and only then determine the relations between the entities. However, this approach has several problems. First, errors made by the named entity recognizer propagate to the relation classifier and may degrade its performance significantly. For example, if “Boston” is mislabeled as a person, it will never be classified as the location of Poe' s birthplace. Second, relation information is sometimes crucial to resolving ambiguous named entity recognition.

For instance, if the entity “JFK” is identified as the victim of the assassination, the named entity recognizer is unlikely to misclassify it as a location (e.g. JFK airport).

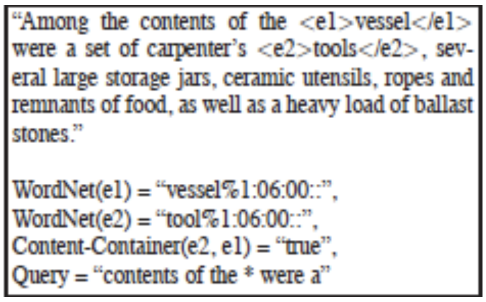
Our algorithm implements a relation recognition system based on a variation of Vector Space Model, as we will explain better in the later chapters. Basically, we would like to develop a system that, given two nominals, is able to correctly classify the relation they share. We use seeds for training our system, and our corpus had been, as we will explain in next chapter, previously POS tagged, but still, we did not encounter the same kind of problem reported here.

2.2.1 SemEval 2007, Task 4. Classification of Semantic Relations between Nominals

The theme of Task 4 of SemEval 2007 (the semantic Evaluation event previously known as SansEval), was the presentation of an evaluation task designed to provide a framework for comparing different approaches to classifying semantic relations between nominals in a sentence. The 14 teams working on this task submitted 15 systems. The classification occurred in the context of sentence in a written English text.

For developing the task, a benchmark data set was created, in order to allow the evaluation of different semantic relation classification algorithms. There was no expectation that a single classification scheme would be proposed, however alluring it would be to try to design a unified standard, because the scheme is likely to have shortcoming. Instead, it was decided to focus on separate semantic relations that many researchers list in their relation sets. An annotated data set was built for seven such relations. Every data set supported a separate binary classification task. The first step for the developing of the project, was the data set creation and annotation guidelines. The data set was created by Natasc and Szpakowicz and had relation labels, part-of-speech and WordNet sense annotations, in order to facilitate classification. Moldovan et al.[30]; and Girju et al.[31] gave the annotators an example of each phrase in a sentence, along with WordNet senses and position of arguments.

The chosen semantic relations were the following: Cause-Effect, Content-Container, Instrument-Agency, Origin-Entity, Part-Whole, Product-Producer and Theme-Tool, with seven detailed definitions, including restrictions and conventions, plus prototypical positive and near-miss negative. For each separate relation, data collection was based on wild-card search patterns allowed by Google. The patterns were built manually, following Hearst (1992). Instances of the relation Content-Container, for example, came up in response to queries such as "* contains *", "* holds *", "the * in the *". Following the model of the Senseval-3 English Lexical Sample Task, the tasks' authors set out to collect 140 training and at least 70 test examples per relation, so they had a number of different patterns to ensure variety. They also aimed to collect a balanced number of positive and negative examples.


"Among the contents of the <e1>vessel</e1> were a set of carpenter's <e2>tools</e2>, several large storage jars, ceramic utensils, ropes and remnants of food, as well as a heavy load of ballast stones."

WordNet(e1) = "vessel%1:06:00::",
WordNet(e2) = "tool%1:06:00::",
Content-Container(e2, e1) = "true",
Query = "contents of the * were a"

This figure illustrates the annotations. The nominals were tagged, so parsing or chunking was not necessary.

For Task 4, the tasks' authors defined a nominal as a noun or base noun phrase, excluding names entities. A base noun phrase, e.g., lawn or lawn mower, is a noun with pre-modifiers. Complex noun phrases were excluded. The procedure was the

[30]D. Moldovan, A. Badulescu, M. Tatu, D. Antohe, R. Girju. 2004. *Models for the semantic classification of noun phrases.* In: HLT/NAACL

[31]R. Girju, D. Moldovan, M. Tatu, D.Antohe. 2005. *On the semantics of noun compounds.* In: Computer Speech & Language

same for each relation. One person gathered the sample sentences (aiming approximately for a similar number of positive and negative examples) and tagged the entities; two other people annotated the sentences with WordNet senses and classified the relations. The detailed relation definitions and the preliminary discussions of positive and negative examples served to maximize the agreement between the annotators. They first classified the data independently, then discussed every disagreement and looked for consensus. Only the agreed-upon examples went into the data sets.

Next, each data set was split into 140 training and no fewer than 70 test examples. The task of classifying semantic relations between nominals attracted the participation of 14 teams who submitted 15 systems. The systems performance information in terms of precision, recall, F-measure and accuracy, macro-averaged over all relations, is shown in the next Table. All the measures were computed as described in Lewis (1991)[32]. The tasks' authors distinguished four categories of systems based on the type of information used - WordNet senses and/or Google queries:

A - WordNet = NO & Query = NO;
B - WordNet = YES & Query = NO;
C - WordNet = NO & Query = YES;
D - WordNet = YES & Query = YES.

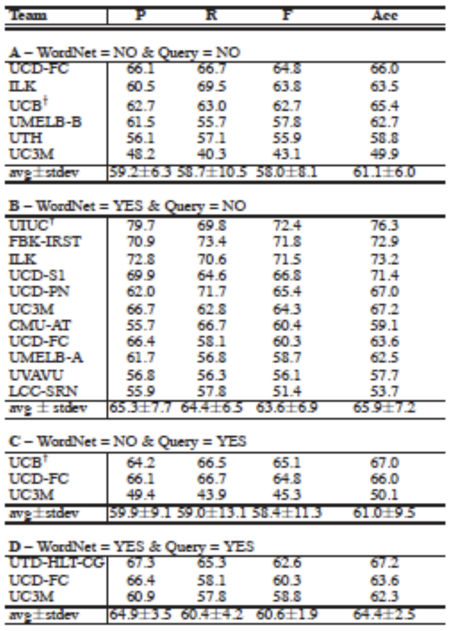
WordNet = "YES" or WordNet = "NO" indicates only whether a system uses the WordNet sense labels in the data sets. A system may use WordNet internally for varied purposes, but ignore their sense labels; such a system would be in category A or C.

Based on the input variation, each submitted system might have up to 4 variations - A,B,C,D. Majority always guesses either "true" or "false", whichever is the majority in the test set (maximizes accuracy). Alltrue always guesses "true"

[32]D.D. Lewis. 1992. *An evaluation of phrasal and clustered representations on a text categorization task.* In: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval

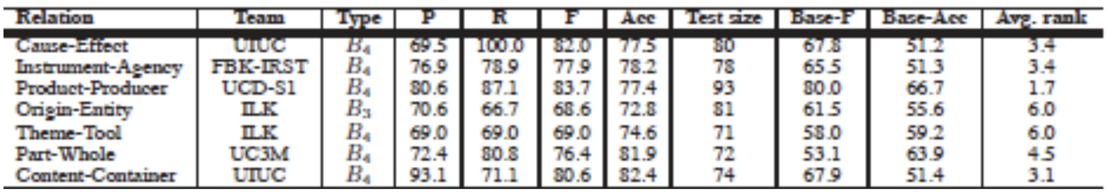
(maximizes recall). Probmatch randomly guesses "true" ("false") with the probability matching the distribution of "true" ("false") in the test dataset (balances precision and recall).

The results in next Table were grouped by category, to facilitate system comparison.



Team	P	R	F	Acc
A – WordNet = NO & Query = NO				
UCD-FC	66.1	66.7	64.8	66.0
ILK	60.5	69.5	63.8	63.5
UCB[†]	62.7	63.0	62.7	65.4
UMELB-B	61.5	55.7	57.8	62.7
UTH	56.1	57.1	55.9	58.8
UC3M	48.2	40.3	43.1	49.9
avg±stdev	59.2±6.3	58.7±10.5	58.0±8.1	61.1±6.0
B – WordNet = YES & Query = NO				
UIUC[†]	79.7	69.8	72.4	76.3
FBK-IRST	70.9	73.4	71.8	72.9
ILK	72.8	70.6	71.5	73.2
UCD-S1	69.9	64.6	66.8	71.4
UCD-PN	62.0	71.7	65.4	67.0
UC3M	66.7	62.8	64.3	67.2
CMU-AT	55.7	66.7	60.4	59.1
UCD-FC	66.4	58.1	60.3	63.6
UMELB-A	61.7	56.8	58.7	62.5
UVAVU	56.8	56.3	56.1	57.7
LCC-SRN	55.9	57.8	51.4	53.7
avg ± stdev	65.3±7.7	64.4±6.5	63.6±6.9	65.9±7.2
C – WordNet = NO & Query = YES				
UCB[†]	64.2	66.5	65.1	67.0
UCD-FC	66.1	66.7	64.8	66.0
UC3M	49.4	43.9	45.3	50.1
avg±stdev	59.9±9.1	59.0±13.1	58.4±11.3	61.0±9.5
D – WordNet = YES & Query = YES				
UTD-HLT-CG	67.3	65.3	62.6	67.2
UCD-FC	66.4	58.1	60.3	63.6
UC3M	60.9	57.8	58.8	62.3
avg±stdev	64.9±3.5	60.4±4.2	60.6±1.9	64.4±2.5

The highest average accuracy on Task 4 was 76.3%. Therefore, the average initial agreement between annotators (70.3%), before revising the definitions, was not an upper bound on the accuracy that could be achieved. That the initial agreement between annotators was not a good indicator of the accuracy that could be achieved, was also supported by the low correlation of 0.15 between the Acc column in the following table and the agreement column in the first table.



Relation	Team	Type	P	R	F	Acc	Test size	Base-F	Base-Acc	Avg. rank
Cause-Effect	UIUC	B_4	69.5	100.0	82.0	77.5	80	67.8	51.2	3.4
Instrument-Agency	FBK-IRST	B_4	76.9	78.9	77.9	78.2	78	65.5	51.3	3.4
Product-Producer	UCD-S1	B_4	80.6	87.1	83.7	77.4	93	80.0	66.7	1.7
Origin-Entity	ILK	B_3	70.6	66.7	68.6	72.8	81	61.5	55.6	6.0
Theme-Tool	ILK	B_4	69.0	69.0	69.0	74.6	71	58.0	59.2	6.0
Part-Whole	UC3M	B_4	72.4	80.8	76.4	81.9	72	53.1	63.9	4.5
Content-Container	UIUC	B_4	93.1	71.1	80.6	82.4	74	67.9	51.4	3.1

Various analysis of the results were performed, which could be summarize here in four questions. Using Xi meant referring to four possible system categories (Ai, Bi, Ci, and Di) with four possible amounts of training data (X1 for training examples 1 to 35, X2 for 1 to 70, X3 for 1 to 105, and X4 for 1 to 140).

Did more training data help?

Overall, the results suggested that more training data improved the performance. There were 17 cases in which they had results for all four possible amounts of training data.

Did WordNet help?

The statistics showed that WordNet was important, although the contribution varied across systems. The results of the UCD-FC system actually went down when WordNet was used.

Did knowing the query help?

Overall, knowing the query did not seem to improve the results. Again, the UCD-FC system differed from the other systems in that the A and C scores were identical, but even averaging over the remaining two systems and 8 cases did not show a statistically significant advantage.

Were some relations harder to classify?

Some relations were more difficult to classify than others. The best F-measure ranged from 83.7 for Product-Producer to 68.6 for Origin-Entity. The difference between the best F-measure and the baseline F-measure ranged from 23.3 for Part-Whole to 3.7 for Product-Producer. The difference between the best accuracy and the baseline accuracy ranged from 31.0 for Content- Container to 10.7 for Product-Producer.
The F-measure showed the best result for each relation, but similar differences among the relations might be observed when all results are pooled. The groups computed the average rank of each relation in the ordered list of relations generated by each system. For example, Product-Producer was often listed as the first or the second easiest relation (with an average rank of 1.7), while Origin-Entity and Theme-Tool were identified as the most difficult. This work made it possible to provide a framework and a benchmark data set to allow for comparisons of methods for classification of semantic relations. The data included different types of information - lexical semantic information, context, query used - meant to facilitate the analysis of useful sources of information for determining the semantic

relation between nominals. The results that have been reported showed successful approaches to this difficult task, and the advantages of using lexical semantic information.

The success of the task showed that the framework and the data are useful resources. The people responsible for this experiment also decided to make this data collection freely accessible, to encourage further research into this domain and the integration of semantic relation algorithms in high-end applications .

2.2.2 Roth and Yih's Probabilistic Reasoning for Entity & Relation Recognition

Roth and Yih's[33] paper developed a novel approach for Relation Recognition - a probabilistic framework for recognizing entities and relations together. In this framework, separate classifiers are first trained for entities and relations. Their output was used to represent a conditional distribution for each entity and relation, given the observed data. This information, along with constraints induced among relations and entities (e.g. the first argument of kill is likely to be a person; the second argument of born in is a location) were used to make global inferences for the most probable assignment for all entities and relations of interest. Their global inference approach accepted as input conditional probabilities which were the outcomes of "local" classifiers. Each of the local classifiers could depend on a large number of features, but these were not viewed as relevant to the inference process and were abstracted away in this process of "inference with classifiers". In this sense, this work extended previous works in this paradigm, such as (Punyakanok and Roth, 2001)[34], in which inference with classifiers was studied when the outcomes of the classifiers were sequentially constrained; here the constraints are more general, which necessitated a different inference approach.

The problem at hand was that of producing a coherent labeling of entities and

[33]D. Roth, W. Yih. 2002. *Probabilistic Reasoning for entity & relation recognition.* In: Proceedings of COLING

[34]V. Punyakanok and D. Roth . 2001. The use of classifiers in sequential inference. In: Arxiv preprint cs.LG/0111003

relations in a given sentence. Conceptually, the entities and relations could be viewed, taking into account the mutual dependencies, as labeled graphs, where the nodes represented entities (e.g. phrases) and the links denoted the binary relations between the entities. Each entity and relation had several properties - denoted as labels of nodes and edges in the graph. Some of the properties, such as words inside the entities, could be read directly from the input; others, like pos tags of words in the context of the sentence, were easy to acquire via learned classifiers. However, properties like semantic types of phrases (i.e., class labels, such as "people", "locations") and relations among them were more difficult to acquire. Identifying the labels of entities and relations was the target of our learning problem. In particular, we learned these target properties as functions of all other "simple to acquire" properties of the sentence. Each nontrivial property of the entities and relations, such as the class label, depended on a very large number of variables. In order to predict the most suitable coherent labels, inferences would have to be made on several variables. However, when modeling the interaction between the target properties, it was crucial to avoid accounting for dependencies among the huge set of variables on which these properties depended. Incorporating these dependencies into their inference was unnecessary and would have make the inference intractable. Instead, these dependencies were abstracted away by learning the probability of each property conditioned upon an observation. The number of features on which this learning problem depended could be huge, and could be of different granularity and based on previous learned predicates (e.g. pos). Inference was then made based on the probabilities.

Although the labels of entities and relations from a sentence mutually depended on each other, two basic classifiers for entities and relations were first learned, in which a multi-class classifier for E(or R) was learned as a function of all other "known" properties of the observation. The classifier for entities was a named entity classifier, in which the boundary of an entity was predefined (Collins and

Singer, 1999)[35]. On the other hand, the relation classifier was given a pair of entities which denoted the two arguments of the target relation. Accurate predictions of these two classifiers seemed to rely on complicated syntax analysis and semantics related information of the whole sentence. However, weak classifiers were derived by treating these two learning tasks as shallow text processing problems. This strategy had been successfully applied on several NLP tasks, such as information extraction and chunking (i.e. shallow paring). It assumed that the class labels could be decided by local properties, such as the information provided by the words around or inside the target. Examples included the spelling of a word, part-of-speech, and semantic related attributes acquired from external resources such as WordNet.

The used propositional learner was SNoW (Roth, 1998; Carleson et al., 1999). SNoW was a multi-class classifier specifically tailored for large scale learning tasks. The learning architecture made use of a network of linear functions, in which the targets (entity classes or relation classes, in this case) were represented as linear functions over a common feature space. Within SNoW, a learning algorithm, a variation of Winnow (Littlestone, 1988)[36] was used. This is a feature efficient algorithm suitable for learning in NLP-like domains where the number of potential features is very large, but only a few are active in each example, and only a small fraction are relevant to the target concept. While SNoW was typically used as a classifier, and predictions were made using a winner-take-all mechanism over the activation value of the target classes, in this case, there was direct reliance on the raw activation value of the output to estimate the posteriors, which was the weighted linear sum of the features. It could be verified that the resulting values were monotonic with the confidence in the prediction, and therefore was a good source of probability estimation. Roth and Yih uses softmax (Bishop, 1995) over the raw activation values as probabilities. Specifically, suppose the number of classes is n, and the raw activation values of class i was acti. The posterior

[35]M.Collins, Y. Singer. 1999. *Unsupervised models for named entity classification.* In: Proceedings of the joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora

[36]N. Littlestone. 1988. *Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm.* In: Machine Learning

estimation for class i was derived by the following equation.

$$p_i = \frac{e^{act_i}}{\sum_{p \leq i \leq n} e^{act_j}}$$

Broadly used in the AI community, belief network was a graphical representation of a probability distribution (Pearl, 1988). It was a directed acyclic graph (DAG), where the nodes were random variables and each node was associated with a conditional probability table which defined the probability given its parents. Roth and Yih constructed a belief network that represented the constraints existing among R's and E's. Then, for each sentence, they used the previously explained classifiers to compute the Prob(Ejobservations) and Prob(Rjobservations), and used the belief network to compute the most probable global predictions of the class labels. The structure of their belief network, which represented the constraints, is a bipartite graph. In particular, the variable E's and R's were the nodes in the network, where the E nodes are in one layer, and the R nodes are in the other.

Since the label of a relation was dependent on the entity classes of its arguments, the links in the network connected the entity nodes, and the relation nodes that have these entities as arguments. For instance, node R_{ij} had two incoming links from nodes E_i and E_j . The conditional probabilities $P(R_{ij}E_i; E_j)$ encoded the constraints. Finding a most probable class assignment for the entities and relations is equivalent to finding the assignment of all the variables in the belief network that maximizes the joint probability. However, this most probable-explanation (MPE) inference problem is intractable (Roth, 1996) if the network contains loops (undirected cycles), which is exactly the case in our network. Therefore, we resorted to the following approximation method instead. Recently, researchers had achieved great success in solving the problem of decoding messages through a noisy channel with the help of belief networks. The network structure used in their problem was similar to the network used here, namely a loopy bipartite DAG. The inference algorithm they used is Pearl' s belief propagation algorithm (Pearl, 1988)[37], which outputs exact posteriors in linear time if the network was singly connected (i.e.

[37]J.Pearl. 1988. *Probabilistic reasoning in intelligent systems.* Morgan Kaufmann Publ.

without loops) but did not guarantee to converge for loopy networks. However, researchers had empirically demonstrate that by iterating the belief propagation algorithm several times, the resulting values often converged to the right posteriors. Due to the existence of loops, Roth and Yih also applied belief propagation algorithm iteratively as their inference procedure.

In order to build different datasets, they first collected sentences from TREC documents, which were mostly daily news such as Wall Street Journal, Associated Press, and San Jose Mercury News. Among the collected sentences, 245 sentences contained relation kill (i.e. two entities that have the murder-victim relation). 179 sentences contained relation born in (i.e. a pair of entities where the second is the birthplace of the first). In addition to the above sentences, they also collected 502 sentences that contain no relations. Entities in these sentences were segmented by the simple rule: consecutive proper nouns and commas were combined and treated as an entity. Predefined entity class labels included other rel, person, and location. Moreover, relations were defined by every pair of entities in a sentence, and the relation class labels defined are other rel, kill, and birthplace. Three datasets were constructed using the collected sentences. Dataset “kill” had all the 245 sentences of relation kill. Dataset “born in” had all the 179 sentences of relation born in. The third dataset “all” mixed all the sentences.
The authors compared three approaches in the experiments: basic, omniscient, and BN.

The first approach, basic, tested their baseline - the performance of the basic classifiers. These classifiers were learned independently using local features and made predictions on entities and relations separately. Without taking global interactions into account, the features extracted were described as follows. For the entity classifier, features from the words around each entity were: words, tags, conjunctions of words and tags, bigram and trigram of words and tags. Features from the entity itself included the number of words it contains, bigrams of words in it, and some attributes of the words inside such as the prefix and suffix. In addition, whether the entity had some strings that matched the names of famous people and places was also used as a feature. For the relation classifier, features were extracted from words around and between the two entity arguments. The types of features included bigrams, trigrams, words, tags, and words related to “kill” and “birth” retrieved from WordNet.

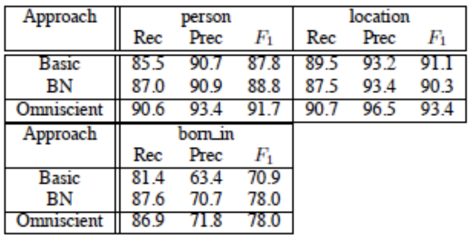
The second approach, omniscient, was similar to basic. The only difference here was the labels of entities were revealed to the R classifier and vice versa. It was certainly impossible to know the true entity and relation labels in advance. However, this experiment might give us some ideas about how much the performance of the entity classifier could be enhanced by knowing whether the target was involved in some relations, and also how much the relation classifier could benefit from knowing the entity labels of its arguments. In addition, it also provided a comparison to see how well the belief network inference model could improve the results. The third approach, BN, tested the ability of making global inferences in the framework. Roth and Yih used the Bayes Net Toolbox for Matlab by Murphy 3 to implement the network and set the maximum number of the iteration of belief propagation algorithm as 20. Given the probabilities estimated by basic classifiers, the network infers the labels of the entities and relations globally in a sentence.

Compared to the first two approaches, where some predictions might violate the constraints, the belief network model incorporated the constraints between entities and relations, thus all the predictions it made would be coherent. All the experiments using these approaches were done in 5-fold validation. In other words, these datasets were randomly separated into 5 disjoint subsets, and experiments were done 5 times by iteratively using 4 of them as training data and the rest as testing. The experimental results in terms of recall, precision,and $F^{-} = 1$ for datasets “kill”, “born in”, and “all” are given in the following tables:

Approach	person			location		
	Rec	Prec	F_1	Rec	Prec	F_1
Basic	96.6	92.3	94.4	76.3	91.9	83.1
BN	89.0	96.1	92.4	78.8	86.3	82.1
Omniscient	96.4	92.6	94.5	75.4	90.2	81.9

Approach	kill		
	Rec	Prec	F_1
Basic	61.8	57.2	58.6
BN	49.8	85.4	62.2
Omniscient	67.7	63.6	64.8

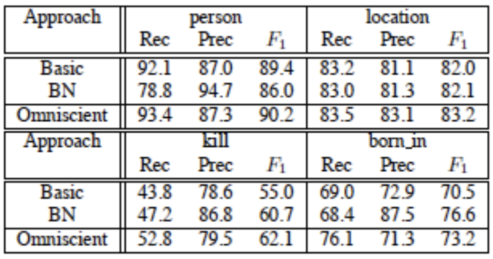
Results for dataset “kill”



Approach	person			location		
	Rec	Prec	F_1	Rec	Prec	F_1
Basic	85.5	90.7	87.8	89.5	93.2	91.1
BN	87.0	90.9	88.8	87.5	93.4	90.3
Omniscient	90.6	93.4	91.7	90.7	96.5	93.4

Approach	born_in		
	Rec	Prec	F_1
Basic	81.4	63.4	70.9
BN	87.6	70.7	78.0
Omniscient	86.9	71.8	78.0

Results for dataset "born in"



Approach	person			location		
	Rec	Prec	F_1	Rec	Prec	F_1
Basic	92.1	87.0	89.4	83.2	81.1	82.0
BN	78.8	94.7	86.0	83.0	81.3	82.1
Omniscient	93.4	87.3	90.2	83.5	83.1	83.2

Approach	kill			born_in		
	Rec	Prec	F_1	Rec	Prec	F_1
Basic	43.8	78.6	55.0	69.0	72.9	70.5
BN	47.2	86.8	60.7	68.4	87.5	76.6
Omniscient	52.8	79.5	62.1	76.1	71.3	73.2

Results for dataset "all"

Two interesting facts were reflected in the results.
First, the belief network approach tends to decrease recall in a small degree but increases precision significantly. This phenomenon was especially clear on the classification of some relations. As a result, the F_1 value of the relation classification results was enhanced to the extent that was near or even higher than the results of the Omniscient approach. This might be explained by the fact that if the label of a relation was predicted as positive (i.e. not other rel), the types of its entity arguments must satisfy the constraints. This inference process reduced the number of false positives, thus enhancing precision. Second, knowing the class labels of relations did not seem to help the entity classifier much. In all three datasets, the difference of Basic and Omniscient approaches was usually less than 3% in terms of F_1, which was not very significant given the size of their datasets. This phenomenon might be due to the fact that only a few of entities in a sentence were involved in some relations. Therefore, it was unlikely that the entity classifier could use the relation information to correct its prediction. The promising results of these preliminary experiments demonstrated the feasibility of this kind of probabilistic framework.

Chapter 3

Preliminary Work

3.1 The algorithm: a brief description

The purpose of our work was to develop a semi-unsupervised system that was able to automatically extract semantical relations between nominals in a text. In addition, we wanted it to correctly classify semantical relations between nominals.

We used a pattern-based approach, selecting seeds instances to infer linguistic patterns. To date, most research on relations harvesting has focused on *is-a* and *part-of* relations. We decided to focus our work on the latter, but we also tried to extract another type of semantical relation, the *location* relation. The location relation indicates that an object, or a person, or a place is situated in a certain other place. For example we could say that a book is *collocated* on a bookshelf, and this would represent a *location* relation for us. Our approach was not web oriented. We did not use WordNet, which is a large lexical database of English in which nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. So, we had no additional information on words meanings.
We did not even use any other resources such as Google counts, which is very useful for accounting words frequencies.
We examined in the previous chapter that those are the most used web-resources

in experiments similar to ours, but we decided not to implement them.
Instead we used a text dump extracted from the Italian Wikipedia in November 2007.

The algorithm we implemented for achieving our purpose was made of succeeding steps, each one covering a particular task. First of all, we had to select the seeds to infer patterns. We manually selected a list of words couple that were known to share the semantic relations we wanted to analyze. Once gathered a list of about 100 couples per relation, we placed them into a text file for further analysis. We wrote a Perl script that took these words as input, together with our corpus, and selected all the sentences that contained the two words, at a given distance the one from the other, meaning that we selected sentences that started with one of the two words of the couple, had another five or six words, and then the second words of the couple. The words composing our seeds were obviously common nouns, since we were interested in extracting semantical relation between nominals. We thought that, having two words sharing a semantical relation, would lead us to find how patterns representing the very same semantical relation would look like. We assumed that we would find patterns occurring in between the two seeds, and we built our script consequentially.

Once a substantial number of patterns was gathered, they were analyzed. Not every pattern found equally represented the relation it was selected for, so we had to choose a association measure in order to effectively compute the degree of representativeness of the pattern with regard to the semantical relation. We decided to use Mutual Information, as used by Pantel and Pennachiotti in Espresso, as our main association measure, but we also used a variation of this measure called Local Mutual Information, which is really useful for reducing noise at low frequencies. Using these measures, we were able to infer which pattern was more representative for every relation, according to the ones having a higher MI or LMI value.

Once this task was completed, having selected the more representative patterns for every relation according to our corpus, we wanted to do something slightly different: we then developed a relation classifier. Basically, given a couple of nominals, the system be able to correctly classify the semantical relation occurring between these nominals. At first, we built a relation classifier bound to discriminate between meronymy and location relation. Our model was a variation of the Vector Space Model commonly used in information retrieval. We built vectors representing the relations we wanted to discriminate between. As a weight measure, we used the previously gathered values of Mutual Information, so that the results of our first experiment were used as a training set for this one. We then built a different array for every word couple we wanted to classify and, using a Perl script, we measured the cosine between the array describing the couple of nominals and the two arrays representing the relations, to evaluate which one between these two was closer to the one representing the nominals. This was obviously the relation it was associated with.

Once this step was completed, we decided to test our algorithm using different kinds of semantical relations. We were inspired by task 4 of SemEval 2007. We tried discriminating between three types of relations, already used in this SemEval task. We chose only three out of the seven relations creators of SemEval selected, since we have already covered the *part-of* or *meronymy* relation in the previous step of our experiment and because we just wanted to test the effectiveness of our algorithm. We decided to try discriminating *cause-effect*, *instrument-agent* and *product-producer* relations. We used the same procedure developed in the previous step i.e. we first selected a list of seeds for every relation, and then we ran our algorithm. We achieved pretty good results, and decided to try another approach, i.e., to use automatically extracted nominals, instead of manually selected ones. To do this, we wrote a Perl script that automatically extracted word couples connected by some patterns, from our algorithm. Then, we ran our algorithm as usual. We discovered that this approach provided even better results than those obtained with manually selected seeds.

This chapter covers all the preliminary work we had to do in order to easily develop our algorithm.

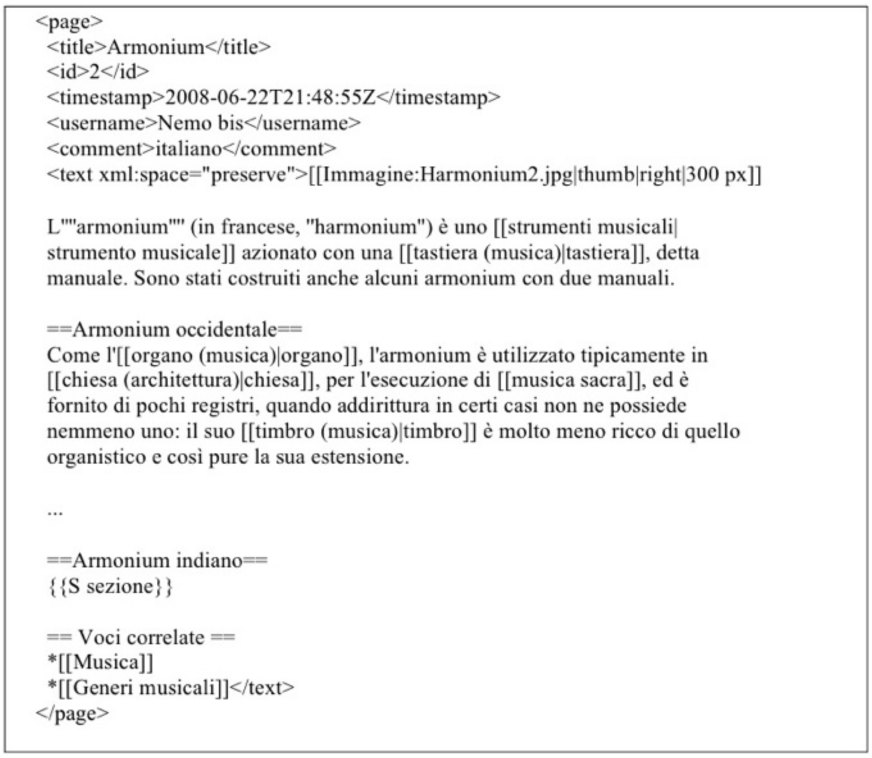
3.2 The corpus

The corpus used for all our experiments was the whole Italian Wikipedia, treated using the modules available in the Tanl (Text Analytics and Natural Language) suite. The corpus was extracted and treated as part of the SemaWiki project, a collaboration between the Computer Science Department and the Linguistics Department of Pisa University and the Institute for Computational Linguistics of CNR, which aims at developing technologies for analyzing text in the Italian language in order to build a Question Answering system based on semantic relations. As part of this activity, a full pipeline of NLP tools has been developed and applied to the Italian Wikipedia, in order to create a large body of text annotated with semantics tags.

When we got our hands on the corpus, the text file had already undergone all the procedures we are going to explain next, and was almost ready to be used.

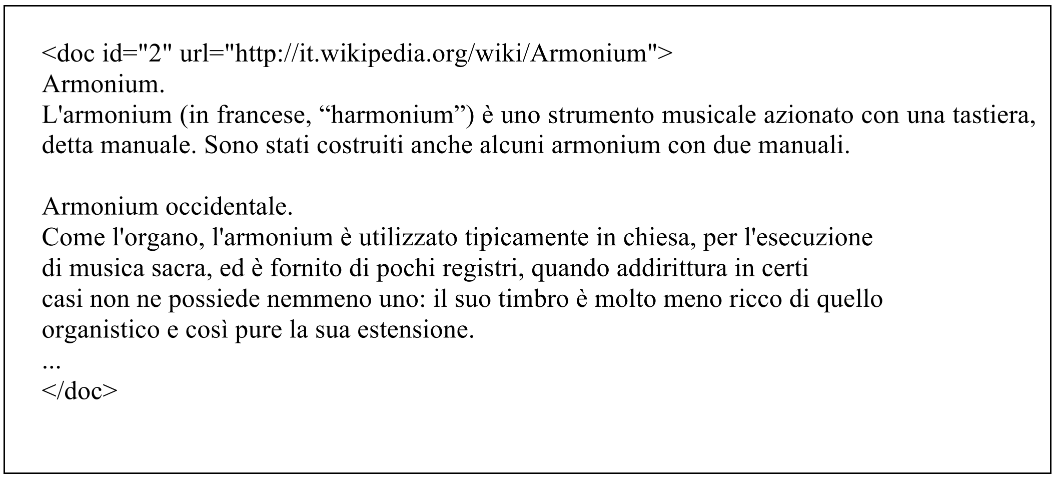
3.2.1 Acquisition

The Wikipedia maintainers provide, each month, an XML dump of all documents in the database consisting of a single XML file containing the whole encyclopedia that can be used for various kinds of analysis, such as statistics, service lists, etc. The Italian Wikipedia dumps are available from the Wikipedia database download. In order to perform any kind of text analysis, it is necessary to extract plain text from the XML dump, removing syntactical decorations. To do this, a particular tool has been used. The Wikipedia extractor tool extracts plain text from a Wikipedia database dump, discarding every other information (such as tables, images or lists). Each document in the dump is represented as a single XML element encoded as follows:



```
<page>
  <title>Armonium</title>
  <id>2</id>
  <timestamp>2008-06-22T21:48:55Z</timestamp>
  <username>Nemo bis</username>
  <comment>italiano</comment>
  <text xml:space="preserve">[[Immagine:Harmonium2.jpg|thumb|right|300 px]]

  L'''armonium''' (in francese, ''harmonium'') è uno [[strumenti musicali|
  strumento musicale]] azionato con una [[tastiera (musica)|tastiera]], detta
  manuale. Sono stati costruiti anche alcuni armonium con due manuali.

  ==Armonium occidentale==
  Come l'[[organo (musica)|organo]], l'armonium è utilizzato tipicamente in
  [[chiesa (architettura)|chiesa]], per l'esecuzione di [[musica sacra]], ed è
  fornito di pochi registri, quando addirittura in certi casi non ne possiede
  nemmeno uno: il suo [[timbro (musica)|timbro]] è molto meno ricco di quello
  organistico e così pure la sua estensione.

  ...

  ==Armonium indiano==
  {{S sezione}}

  == Voci correlate ==
  *[[Musica]]
  *[[Generi musicali]]</text>
</page>
```

For this document the Wikipedia extractor produces the following plain text:



```
<doc id="2" url="http://it.wikipedia.org/wiki/Armonium">
Armonium.
L'armonium (in francese, “harmonium”) è uno strumento musicale azionato con una tastiera,
detta manuale. Sono stati costruiti anche alcuni armonium con due manuali.

Armonium occidentale.
Come l'organo, l'armonium è utilizzato tipicamente in chiesa, per l'esecuzione
di musica sacra, ed è fornito di pochi registri, quando addirittura in certi
casi non ne possiede nemmeno uno: il suo timbro è molto meno ricco di quello
organistico e così pure la sua estensione.
...
</doc>
```

The extraction tool was implemented in Python and it aims to achieve high accuracy in the extraction task.

The standard page format adopted by Wikipedia uses the Wiki syntax, which is a simple and intuitive formalism for specifying meta-information associated to texts (bolds, italics, underlines, images, tables, etc.). Unfortunately this standard is not in use by every author, and some of them prefer to insert HTML markup inside the documents. Wiki and HTML tags are often misused in the text (not closed tags, wrong attributes, etc.) so the extractor deploys several heuristics for maximizing the probability of success.

The Wikipedia dump we used was the one extracted at the end of October 2008, that we obtained by middle November.

3.2.2 Composition

The Italian Wikipedia has various categories into which different topics are grouped.

Main categories are:

- Mathematical, Physical and Natural Sciences
- Art, Literature, Visual and Performatory Arts
- Social and Human Science - Human Activities
- Society, Custom, People
- Technology and Applied Sciences

Each of the main categories have with some sub-categories. Examining the categories we could easily guess that entries regarding different categories would use different types of language. In topics related to scientific categories, the language used would be very technical and domain-related, with specific terms. On the other hand in categories like Society, Custom, People, the language used would be less specific and more related to everyday use, given that the topics would be more common and less specific.

Also, its easy to assume that topics belonging to more specific categories (like Mathematical, Physical and Natural Science, but also like Art, Literature, Visual and Performatory Arts) would be written by people working in those areas or subject matter experts. With the experience and the knowledge of a certain domain, also comes the right language and terminology to speak about it. Instead, topics like Custom and Society includes some sub-categories like gossip or fashion, that often include topics written in a very common and everyday language. Still, even in these categories there are some topics that could be written in very domain-related language. One example could be the category "Bad terms in Italian Language" which could be found under "Custom" → "Common Terms" → "Common Italian Terms" → "Bad terms in Italian Language". Even if what the author is trying to explain is pretty bad, everything is explained in a sort of sociolinguistics fashion, using the proper terminology. In other words, where this kind of topics is concerned, it would be incorrect to say that the subject covered is somewhat influencing the way it is covered.

However, we have to keep in mind that Wikipedia is written by anyone who is willing to contribute. Even if every contribution is put through very careful review, it is still possible to find some pages containing errors, both in content and in form. Lately, on the home page of the Italian Wikipedia, it is possible to find the entry to be translated for each week. The so-called "translation of the week" is a project undertaken by the inter-language co-ordination group, and its aim is to have every topic translated in the majority of languages Wikipedia is available in.

Pages that are developed in this way are somewhat interesting because they do not relate on the type of language chosen by the author, but by a translator who could be totally inexperienced regarding the topic he is translating, just tries to do his best translating it. Still, if the translation is been made from English to Italian, and the English page contained an error that was not caught during revision, the very same error would be put in the Italian version. Analyzing the categories previously reported, it is evident that this kind of corpus covered different type of topics using different kind of language styles. Using a corpus like this raised some problems and doubts, some of which are already covered (like the truthfulness of some affirmations, and consequentially the truth of some extracted relations) while some of them would be covered in next chapters.

3.2.3 How the corpus was elaborated before we got it

Before we obtained the corpus, the text which was already cleaned of any additional information, such as tables, images or lists (as seen in paragraph 2.2.1), has been put through some other instruments in order to have it split into sentences, tokenized and part-of-speech tagged.

For splitting the corpus into sentences, a Python tool called SentenceSplitter.py was used. The tool works on plain text in document format. A document file is defined as containing a series of Wikipedia articles, represented each by an XML doc element:

```
<doc>...</doc>
<doc>...</doc>
...
<doc>...</doc
```

The element doc has the following attributes:

- id, which identifies the document by means of a unique serial number
- url, which provides the URL of the original Wikipedia page.

The content of a doc element consists of pure text, one sentence per line. Here is an example of a doc element:

```
<doc id="2" url="http://it.wikipedia.org/wiki/Harmonium">
Harmonium.
L'harmonium è uno strumento musicale azionato con una tastiera, detta manuale.
…
</doc>
```

For Wikipedia conventions the first line is the title of the article. This kind of text is, basically, what was obtained after the cleaning described in paragraph

2.2.1 . When used as a script, the Sentence Splitter reads the plain text from the standard input and writes the sentences to the standard output. If used as a script from command line, it would look like this: After being split into sentences,

```
> SentenceSplitter.py -t IT-Model.pickle
<doc id="1" url="test">
Prima frase. Seconda frase.
</doc>
<doc id="1" url="test">
Prima frase.
Seconda frase.
</doc>
^D
> _
```

the file was tokenized. Tokenization is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing. A process of tokenization could be used to split the sentence into word tokens. A lexeme, however, is only a string of characters known to be of a certain kind (eg, a string literal, a sequence of letters). In order to construct a token, the lexical analyzer needs a second stage, the evaluator, which goes over the characters of the lexeme to produce a value. The lexeme's type combined with its value is what properly constitutes a token, which can be given to a parser.

Though it is possible and sometimes necessary to write a lexer by hand, lexers are often generated by automated tools. These tools generally accept regular expressions that describe the tokens allowed in the input stream. Each regular expression is associated with a production in the lexical grammar of the programming language that evaluates the lexemes matching the regular expression. These tools may generate source code that can be compiled and executed or may construct a state table for a finite state machine (which is plugged into template code for compilation and execution).

Regular expressions compactly represent patterns that the characters in lexemes might follow. For example, for an English-based language, a NAME token might be any English alphabetical character or an underscore, followed by any number of instances of any ASCII alphanumeric character or an underscore. This could be represented compactly by the string

$$[a - zA - Z_{]}[a - zA - Z\_0 - 9]$$

This means

> any character a-z, A-Z or _, followed by 0 or more of a-z, A-Z, _ or 0-9

. Regular expressions and the finite state machines they generate are not powerful enough to handle recursive patterns, such as:

> n opening parentheses, followed by a statement, followed by n closing parentheses

They are not capable of keeping count, and verifying that n is the same on both sides – unless there are a finite set of permissible values for n. It takes a full-fledged parser to recognize such patterns in their full generality. A parser can push parentheses on a stack and then try to pop them off and see if the stack is empty at the end.

The Lex programming tool and its compiler is designed to generate code for fast lexical analyzers based on a formal description of the lexical syntax. It is not generally considered sufficient for applications with a complicated set of lexical rules and severe performance requirements; for instance, the GNU Compiler Collection uses hand-written lexers.

The tokenizer used here was built using Quex, a lexical analyzer generator. Quex produces a directly coded lexical analyzer engine. Those engines are much faster than the table driven engines of the lex/flex family. For convenience, Quex parses regular expressions in the traditional lex/flex style. This way switching from flex to quex is made very easy. In addition to the fast analyzer engine, quex provides many advanced features, such as “lexer modes” that can be inherited and that provide events for mode transitions. Mode transitions can be allowed and disallowed, one can trigger on indentation events, and many parts of the generated lexical analyzer class can be adorned with its own code.

For easier handling of token sequences a fast token queue is implemented that enables the implementation of lexical analysis directly from sequence diagrams. Also, Quex is based on a dedicated buffer handling strategy that is webbed into the lexical analyzer to provide optimal performance. The established tool for character code conversions 'icon' can also be included into the buffer handling. The Python module is called Tokenizer and exposes the class Tokenizer, from which one can create an Enumerator <Token*> by means of method pipe(). The enumerator exposes methods MoveNext() and Current() as well as the Python iterator interface. Here is an example of its usage:

```
python
>>> import Tokenizer
>>> l = Tokenizer.Tokenizer()
>>> p = l.pipe()
uno due tre
>>> p.MoveNext()
True
>>> c = p.Current()
>>> p.MoveNext()
True
>>> c1 = p.Current()
>>> p.MoveNext()
True
>>> c2 = p.Current()
>>> c2.form
'tre'
>>> c1.form
'due'
>>> c.form
'uno'
>>> p.MoveNext()
True
>>> c3 = p.Current()
>>> c3.form
'\n'
>>> p.MoveNext()
False
```

Lexer is a simple rule based Flex scanner for the Italian language that reads a text stream and extracts tokens to stdout. If no file argument is supplied, lexer reads from standard input. After that, the file is ready to be pos tagged. Part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up the words in a text (or corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context - i.e. relationship with adjacent and related words in a phrase, sentence or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times. This is not rare - in natural languages (as opposed to many artificial languages), a huge percentage of word-forms are ambiguous.

In part-of-speech tagging by computer, it is typical to distinguish from 50 to 150 separate parts of speech for English.

The used part-of-speech tagger is based on the PISA ILC/PAROLE tagset and is conformant to the EAGLES international standard. The tags used could be coarse-grained or fine-grained, with additional feats that contain an unordered set of morph-syntactic features complemented the part- of-speech information. The fine-grained tags are documented in the table below:

Value	Description
A	Adjective
AP	Possessive Adjective
B	Adverb
C	Conjunction
DD	Demostrative Determiner
DE	Exclamative Determiner
DI	Indefinite Determiner
DR	Relative Determiner
DT	Interrogative Determiner
E	Preposition
I	Interjection
N	Cardinal Number
NO	Ordinal Number
PD	Demonstrative Pronoun
PI	Indefinite Pronoun
PP	Possessive Pronoun
PQ	Personal Pronoun
PR	Relative Pronoun
PT	Interrogative Pronoun

PU	Punctuation
RD	Determinative Article
RI	Indeterminative Article
S	Common Noun
SA	Abbreviation
SP	Proper Noun
SW	Foreign Noun
V	Verb
X	Residual Class

The field contains an unordered set of morph-syntactic features complementing the part-of-speech information. The corpus we used, had this additional information but, as we will explain in the next paragraph, we removed it because we did not need it for our purpose.

3.2.4 Consistency

The corpus described here was made of about 150 million tokens. The text file containing it was about 6 GB big.

3.3 How we modified the corpus

The file we obtained after the POS tagging, had the following structure:

con con E

un un RIMS

vero vero A-MS

e e C

proprio proprio A-MS

plebiscito plebiscito S-MS

```
per per E
il il RDMS
sindaco sindaco S-MS
```

There was one word per line, and every word had its part of speech tag. This kind of part of speech tag is composed of two parts, the POSTAG and the FEATS (optional).

As previously mentioned, for our purpose, we did not really need the additional information provided by the FEATS field. There was no value in knowing if a name was masculine or feminine, because all that we needed was knowing the position of the name in the sentence, knowing if the name appeared in the position we wanted it to be. So we could easily get rid of this information without losing what was needed for analysis. Also, eliminating this un-needed information from our corpus made the text file smaller. This was useful since we needed the text file as small as possible,so that it would be easier and faster to our machine to process every file. For the very same reason, we decided that having one word per line made no sense, and that it would be easier to process a file that was linear, like a normal text file in which every word had the associated part of speech tag.

But, first of all, we had to replace every space with a character (like "/") that would made it possible to consider every line as if it was a single word composed by the word itself, the root it came from and the pos tag.

To implement the modifications needed, we used a Perl script that took as input the text file we had, looking like the one we already showed, and made an output file that had what we wanted. We used Perl for every script we had to make for these experiments, because it is probably the best programming language for handling text files, that is the kind of data we had.
Basically we did something like this:

```
FUNCTION pulituraCodice (Corpus) {
Takes the whole corpus as Input and returns the cleaned version of it as Output

FOR EACH line IN Corpus DO
       Remove everything after "-"
       Substitute every space with "\"
       Eliminate carriage return
END
}
```

The actual Perl script used could be found in Appendix 1.

The output was what we expected, a text file containing all the words, each of them with its pos tag and one word next to the other like it happens in a "normal text".

con/con/E un/un/RIMS vero/vero/A e/e/C proprio/proprio/A plebiscito/plebiscito/S per/per/E il/il/RDMS sindaco/sindaco/S

This format of the text file was exactly what we used for analysis. The output file text containing the corpus was divided in 36 smaller files. This was absolutely necessary to speed up the process of analyzing the text, otherwise it would have taken too much time, or worse, it would have frozen our computer.

To divide our file in smaller files, we just used some commands of the Macintosh shell. In fact, for developing our algorithm, we used a 2008 Macbook with a 2.4 GHz Intel Core 2 Duo processor and 2 GB 667 MHz SDRAM.

We used a bash shell, which was the default Mac OS shell since Mac OS X.3 Panther.

All this was done because, even though the machine was pretty new, it was not that fast during the execution of our algorithms and froze frequently. We actually tried executing one of the Perl scripts described in the next chapter, using as input the whole text, and ended up with a frozen computer which would not execute anything. We then had to think about some ways of simplifying the execution and reducing the files to smaller ones seemed the simplest way of dealing with the problem. Once we had obtained these 20 smaller text file, which text was exactly in the format we planned it to be, we were ready for start our experiments.

Chapter 4

Pattern Extraction and Evaluation

In the previous chapter, we described the various steps prior to composing our algorithm for the extraction and the classification of sematic relations. In this chapter we are going to explain how we developed the first step of this algorithm, that is, the relation extractor. As we have already briefly explained in chapter 2, we decided to develop a semi-unsupervised learning algorithm for mining text corpora for patterns expressing implicit semantic relations.
We selected a list of input word pair X:Y, with a very well known semantic relation between them, which we used as seeds for extracting the corresponding list of output patterns $\langle Pi, ..., Pn \rangle$. The patterns were supposed to be sentences commonly used to express the given relation. For example, if we analyze the cause-effect relation between two words (X and Y), we would say that given relation is true if X **is the cause of** Y or if Y **is caused by** X. The words in bold are considered to be patterns expressing the cause-effect relation. The obtained output patterns were then ranked according to how well they described the given relation. For ranking the output we used Mutual Information (MI), a dimensionless quantity which can be thought of as the reduction in uncertainty about one random variable given knowledge of another.
High mutual information indicates a large reduction in uncertainty; low mutual

information indicates a small reduction.
We started our experiment by trying to extract patterns which better described meronymy-holonymy relation and then we repeated the same procedure with location relation. A meronymy denotes a constituent part of, or a member of something. That is,

X is a meronym of Y if Xs are parts of Y(s), or

X is a meronym of Y if Xs are members of Y(s).

For example, "finger" is a meronym of "hand" because a finger is part of a hand. Similarly "wheel" is a meronym of "automobile". We then tried to apply the developed procedure for extracting another kind of patterns: the ones expressing a location relation. A location denotes a constituent is placed somewhere. That is,

X is a location of Y if X is collocated in Y.

We then wrote a Perl script that took every seed couple (w1 and w2) and our corpus as an input, all the sentences in which w1 and w2 occurred simultaneously and extracted everything in between with a given distance of 5 or 6 words from one seed to the other. These extracted sentences would later become our patterns. Every one of this patterns represented, to a certain degree, the semantic relation that we were trying to extract. We then had to sort our patterns to see which one better represented the relation taken in exam. That is where Mutual Information came in handy. We calculated the Mutual Information running between any seed couple and every pattern they appeared with, and obtained as a result which pattern better described the given relation for the given couple. The second step was to consider every couple as the same MERO or LOCA string, in order to find the more representative patterns for the given relation. Patterns that were high in our classification table would better represent the semantic relation, while patterns that were low would be less representative.

4.1 Selecting the seeds

The first step of our algorithm was selecting a list of word pair connected and related by the semantic relation we were trying to describe, to be used as seeds for our experiments. This is a somewhat common procedure when trying to extract relations from texts. Hearst used this criterion (but she applied it to the is-a relation only), while Berland and Charniak actually found lexical patterns that tended to indicate part-whole relations. They followed Hearst and took two words that were in a part-whole relation, finding sentences in a corpus that had those words within close proximity. Pantel and Pennacchiotti's Espresso was based on Hearst's framework, so their algorithm took as input a few seed instances of a particular relation and iteratively learned surface patterns to extract more instances.

Even Turney used some word pair for training his algorithm. What we did, anyway, was slightly different since the word pair he used as input were not seeds, because they were not labeled nor grouped, while our were grouped according to their semantic relation. What we did could be considered pretty similar to what had been done with the Espresso algorithm. Still, Espresso iteratively learned to extract more instances, while our algorithm just used the seed to learn patterns intercourring between them. But what we did could be easily associated with Berland and Charniak's work, too, because we tried to use our seed to extract patterns that indicated a given relation, while finding sentences in our corpus containing the words of every seed couples within a given number of words. First, we tried to extract patterns describing meronymy relation, so we selected a list of 110 couple known to be connected by meronymy relation. These words were selected by a human, whose only criterion for choosing them was his judgment. Basically, if the selector thought a couple was representative for a given relation, said couple could be added to the list.

The list has also been reviewed by another human judge who added some other couples and deleted the ones that were not suitable, according to his own judgment. Having Wikipedia as our corpus meant we had to choose our words carefully. First

of all, we should say all the words used as seeds were POS tagged as “S ” since they were all nominals, because we focused our work in trying to extract relations between nominals.
Since Wikipedia is an online encyclopedia whose content could be provided by almost anyone willing to do it, there are many pages about very generic topics and just as many pages about very specific and domain-related topics, as we already explained in paragraph 2.3.2. We had to account this situation while selecting our seeds and take care of both categories. We seemed to favour generic words, which were easier to think about, but there were some very specific and somewhat uncommon words that had been used because they were accounted in our corpus, too and were good in expressing the given semantic relation. Our list for the meronymy relation is found in Appendix A.

The list was made of domain-free words like, for instance, “noce-gheriglio” (“walnut-kernel”), “edificio-facciata” (“building-faade”) or “libro-pagina” (“book-page”). These words were easy to think about and were accounted in our corpus, so they were added to the list.
In our list are also accounted words that are not really “common” in Italian, but were indeed included in our corpus and consequently in our list, like for instance “OCS:Chip” which is clearly connected to computer science domain, or “N2:azoto” (“N2:nitrogen”), clearly belonging to chemical domain, or “Cervello-Talamo” (“brain-thalamus”) belonging to the biology or medical field. We could observe that there are many chemical terms because there were many Wikipedia pages regarding chemical domain and biology as well. As we could easily predict, since Wikipedia is written by volunteers and not by experts, there were a lot of mistakes, and also some “untypical” words, not really used in everyday language; old terms that are no longer used and sound wrong even if they are not. An example could be represented, in our list, by the couple “elaboratore:memoria” (“computer:memory”). “Elaboratore” is an Italian word that means computer, but is no longer used in everyday language, and has been replaced by the English term. However we decided to keep the word in our list because it was accounted in our corpus, so it should lead to some results.

Selecting pertinent words was a very important step in the procedure. If the words selected as seeds were not present in our corpus, the script could not extract patterns for these words. If they were not sufficiently representative of our relation, they could lead to the extraction of just as much non-representative patterns and compromise the results of our analysis, because as discussed, the extracted patterns were supposed to be the words commonly used to express a given relationship. We then had to make a second list for seeds representing the location relation. Also in this case, we had to selected words very carefully.
As already done for the meronymy relation, the list was selected by human judges using their judgment as the only criterion for adding seeds to the list. First, we selected a list made of words that were thought to be appropriate to represent a location relation. We then noticed that between them, there were a lot of proper nouns. This was expected, because is easy to think that a certain place is located in a certain geographical area. In this case, both the name place and the geographical area would be indicated with their proper noun. It's also easy to think that a certain company is located in a certain place, and even in this case, both the company and the place would be indicated by their proper nouns. Two example mentioned earlier, are the seeds "Apple:Cupertino" or "Riddlesden:Yorkshire", included in the first version of our list.

At first we included proper nouns in our list of seeds, but then we removed them. We decided to try extracting our patterns without using proper nouns as seeds. We also verified that they were not very well accounted in our corpus, especially relating to location relation. Our first list, without the proper nouns, did not produce a lot of patterns, that's why we had to add other seeds to said list. In the end we obtained a list, made of 92 seed couples.

Only two of the couples contained a proper noun: "Valle:Piemonte" which means "Valley:Piemonte" and "Piramide:Egitto" which means "Pyramid:Egypt". In this case the couples were not made of two proper nouns as mentioned above, but just one. The fact that they have been used was partially justified, since using just one proper noun was less specific, less binding and led to more results when extracting the patterns that using two of them, which would have led to a really specific search throughout our corpus, and we would probably obtained just a few results. The list is made up of very common words, not really domain related, except for a very small group. We listed words like "Scoglio:Spiaggia" ("Rock-Beach") or "Statua:Scalinata" and "Statua:Cittadina" ("Statue:Flight of stairs" or "Statue:Town"). We chose the term "scalinata" ("flight of stairs") over "scala" ("stair") and "cittadina" ("town") over "città" ("city") which are obviously more common, because of the occurrences of these words with the word "statua" ("statue") in our corpus. Selecting words that occurred together more frequently was necessary to find more patterns, as such this led us to chose uncommon words versus more common ones.

Once we had both lists, we had to slightly change the words. In our algorithm, we would have to account for occurrences of every word in both their singular and plural forms. We wanted the script to take care of every combination of the two because we wanted to maximize the number of patterns extracted. So, let's say the script was analyzing the seeds "Tavolo:Cucina" ("Table:Kitchen"), it needed to account for the following four effective word pair:

Tavolo:Cucina ("Table:Kitchen")
Tavoli:Cucina ("Tables:Kitchen")
Tavolo:Cucine ("Table:Kitchens")
Tavoli:Cucine ("Tables:Kitchens")

and extracting patterns between them. To do this, we wrote the words as if they were implemented in a Regular Expression. In computing, Regular Expressions provide a concise and flexible means for identifying strings of text of interest, such as particular characters, words, or patterns of characters. Regular expressions are written in a formal language that can be interpreted by a regular expression processor, a program that either serves as a parser generator or examines text and identifies parts that match the specification provided.

Given that the words would be implemented in a Regular Expression in the Perl script (as we will see in the next paragraph), we wrote our seeds as a Regular

Expression that found both the singular and the plural version of the same word, for instance:

“Tavolo:Cucina” became “Tavol[oi]:Cucin[ae]”

In a Regex, the two char between the squared brackets mean that you can choose one char or the other. In this way our purpose was completely satisfied. We would have done something similar for capital letters at the beginning of the word, but it was unnecessary since there is a special instruction that could be implemented in the regex, which does that automatically. We implemented this instruction in the following step.
Our list was saved as MacOS Roman format, since we were working on a Macbook. The corpus instead was saved in UTF-8 format. Selecting the seeds was the first step of our algorithm: after compiling the two list of seed couples we were ready to develop the algorithm for pattern extraction.

4.2 Extracting the patterns

As has been discussed, our patterns were sentences which expressed the given semantic relation we were trying to describe. These patterns were extracted from our corpus using a Perl script that took as input our list of seeds and our corpus. For every couple, the script searched all the sentences in the corpus in which the seed words occurred simultaneously and then extracted everything in between the two words, with a given distance of 5 or 6 words between them.
The script's first task was to take every couple in the list and put it into a hash. In Perl, hashes are a sort of associative arrays. Their structure is similar to an array structure, but instead of using an index to access or write an element, hashes use keys. A Perl hash would look like this:

$$
\begin{gathered}
\%item = (\\
'code' \Rightarrow 22, \\
'name' \Rightarrow' Cobra', \\
'description' \Rightarrow' CollectionKnife', \\
'price' \Rightarrow 110000 \\
\qquad\qquad\qquad\qquad);
\end{gathered}
$$

The arrow could be exchanged with a comma without causing any problems for the script. This kind of structure was exactly what we needed because this way, our list was implemented in a hash which allowed every meronym to be the key for its holonym and every object to be the key for its location. Basically, it maintained the relation between every word in the first column of our list and every word in the second column. The first word, the one before the colon (":"), should become the key of the hash, while the second word should become the value. We initialized an empty hash (my %hash) and used a "while" cycle to read the entire input text file, since we had to do this for every single word pair contained in our list. We then needed to take each text line one by one, since every couple was contained in a single line. To do this we used the chomp instruction, which also removed any newline character from the end of a string. We then needed to build

the actual hash putting the words into the key or the value. To do this we divided the words using the colon as a discriminant and put the word to the left of the colon into a variable called $word1, and the word to the right of the column into a variable called $word2. The hash was then filled up using every $word1 as a key and every $word2 as the corresponding value.
Basically, we did something like this:

```
FUNCTION listCreation (List, Hash) {

Takes the list containing our seed couples as Input and create a Perl Hash with
them which is the output

CREATE new Hash
FOR EACH line IN List DO          (every couple was contained in a single line)
          Divide the couples using ";" as discriminant
          Put the words left to the ";" as a key of the hash
          Put the words right to the ";" as the corresponding value of the hash
END
}
```

This was the preliminary work needed in order to correctly import our list in the script.

4.2.1 The Script

Next we imported the text file that containing our corpus. As previously mentioned, we divided the corpus into 36 smaller text files because the one containing our corpus was too big to be processed easily with our computer.
We opened a text file, containing part of the corpus we were going to process and another text file which was the output for that selected part. When the analysis of every single part was completed, we put all our output together in a single file. What we did at this point was to implement a loop that examined our text, and for every couple $key - $value of our hash, the script searched for a string which begun with our $key, followed by six words, and a punctuation sign except for the point (i.e. the period) or for brackets, and then again our $value.

Since our corpus was POS tagged we could tell the Regular Expression which part of speech we wanted every single word to be. We allowed words that are adjectives, numerals, pronouns, conjunctions, prepositions, verbs, selecting the correct POS for every category and place it in our Regular Expression. We decided not to include periods or brackets for a reason. If a word (let's say the first seed of a couple) is before a point and the other one is after, they are probably not in the same sentence and as such are not useful for our purpose, since there would be no relations between them. Similarly, if one of the two words is inside a bracket, and the other is not, they probably are not sharing any kind of semantic relation. It was necessary for the Regular Expression to ignore words with uppercase, for instance if the first word refers to the beginning of a sentence. As such we added the char i at the end of the regex, which means that the search should be case insensitive.

We placed every result in a string that was then printed inside a text file, so we could have a text file with every pattern extracted by our regex. The pseudo-code for that script is the following:

```
FUNCTION patternSearch (Corpus, Hash) {

Takes the Corpus as Input and searches for patterns expressing the
semantic relation, returning them as output

FOR EACH text IN Corpus DO
        Search sentences starting with "$Key"
        Make sure said sentences end with "$value"
        Make sure said sentences have six words top between "$key" and "$value"
        and said words  are adjectives, numeral, pronouns, conjunctions,
        prepositions, verbs but not other nominals. Punctuation is allow except
        for brackets and point, since words following brackets and point are
        usually referring to a different sentence. This search has to include both
        capital and not capital letters

END}
```

This was the preliminary work, needed in order to correctly import our list in the script.

All this was implemented in a Perl script that produced the results that we expected.
We also had to consider sentences that started with the words after the colon in our list and ended with the words before the colon, so basically sentences that reported the given seeds, but in reverse order compared to what we specified. To do this we had to exchange the variables $key and $value in our regex. We could have done this in the same Perl script used before implementing two parallel (obviously not two indented) loops, but we noticed that computing times were really long, and decided to write another script which simply exchanged the position of the two variables inside the Regular Expression. The script we made is listed in Appendix 2.

We obtained 530 total sentences for the meronymy relation and 575 total sentences for the location relation. As stated, these patterns were obtained using every possible order of the words in our list (we considered w1 as our first word and w2 as our last word and w2 as our first word and w1 as our last one). The extracted sentences were really generalized, like in this format:

w1 .* pattern .* w2

This meant that a sentence was made of one word in our list, some other words that did not express any relation (identified with the structure .*, which means every character occurring once or more than once), the effective pattern, some other words that did not express any relation, and the word in our list that was the one corresponding to the first one, as set up in our list. We then had to make a list containing only the patterns. The patterns extracted from the sentences were obviously duplicated and repeated, but we put every pattern just once in said list, getting rid of possible duplications. Obviously this was not enough to understand which patterns better classified the given semantic relation. To discover this, we needed to use some association measure, which allowed us to understand how every pattern related itself to every word pair it co-occurred with and how every pattern described the semantic relation it was extracted for. Basically, we wanted to know the strength of the association between the extracted patterns and the input instances.

4.3 Evaluating the patterns

What we needed to do at this point was to discover which kind of patterns better described the given semantical relations. We extracted a lot of patterns, but not every pattern had the same degree of connection with the relation it was associated to. We had some idea on what we expected to be more representative, but we had to verify and confirm our hypothesis.
We extracted different kinds of patterns. Some were made of short proposition (two words top, generally a infinitive verb and a preposition or a present participle verb and a preposition), others were made of just one word, which could be a preposition or a conjunction, rarely a verb or an adjective. According to Zipf's law and its application to linguistic tasks, we thought that probably the short propositions were more representative than the single words. This was because Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, which occurs twice as often as the fourth most frequent word, etc. For example, in the Brown Corpus "the" is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million). True to Zipf's Law, the second-place word "of" accounts for slightly over 3.5% of words (36,411 occurrences), followed by "and" (28,852). Only 135 vocabulary items are needed to account for half the Brown Corpus. It also says that in a corpus there will be a lot of words thinly representative and just a few words really representative. This is logical since words like conjunctions and prepositions occur pretty much in every natural language sentence and are obviously not much domain-related. Since our single words were prepositions and conjunction, we thought that probably they would not be so representative.

To prove our point, we chose to use an association measure, called Mutual Information.

4.3.1 Mutual Information

As discussed earlier, in probability theory and information theory, the mutual information (sometimes known by the archaic term transinformation) of two random variables is a quantity that measures the mutual dependence of the two variables. This kind of association measure had been already used by Pantel and Pennacchiotti in Espresso, with exactly the same purpose, since they required their extracted patterns to be highly associated with every input instance. Mutual Information suited our task perfectly, since we wanted to know the degree of relatedness between a certain extracted pattern and a certain input the pattern was extracted with.

Also, we wanted to discover how good every pattern was in describing the relation it was supposed to be representing. Formally, the mutual information of two discrete random variables X and Y can be defined as:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in Y} p(x,y) log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right)$$

where p(x,y) is the joint probability distribution function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of X and Y respectively. First, we wanted to know which pattern better described the given relation for every different word pair it had been extracted with. The needed formula was the following:

$$MI = log_2 \frac{p(P,M)}{(p(P) * p(M))}$$

where p(P,M) was the probability of finding a word pair with a certain pattern, p(P) was the probability of finding a certain pattern and p(M) was the probability of finding a word pair.

p(P,M), or the probability of PM, could be computed as the ratio between the frequency of PM, that is the number of times that the given pattern and the given

couple occurred together in our corpus, and the total occurrences of the pattern in the whole corpus.
p(P), or the probability of finding a given pattern in our corpus, could be computed as the ratio between the frequency of P, that is the total number of times the given pattern was found in our corpus, and the total occurrences of the pattern in the whole corpus.
p(M) or the probability of finding a given word pair in our corpus, could be computed as the ratio between the frequency of M, that is the total number of times the given word pair were found together in our corpus, and the total occurrences of the pattern in the whole corpus.
Together with the Mutual Information, we used another measure, called LMI, Local Mutual Information. LMI is a more sophisticated version of Mutual Information, which tends to remove the noise that could be found with low frequencies. LMI could be computed using the following formula:

$$LMI = MI * freq(PM)$$

where freq(PM) is the number of times the given pattern P occurs with the given word pair.

4.3.2 Our way of proceeding

We used a perl script to count the occurrences of the pattern with every word pair contained in our list. The output we had in mind was something like this:

$$(Pianeta - Spazio) \Rightarrow 2$$
$$(Opera - Museo) \Rightarrow 1$$
$$(Libro - Biblioteca) \Rightarrow 1$$

The previous list is a little part of the output we obtained for the location relation, regarding the pattern “A”. We made a script that gave us an output file for every pattern, this way our data were already organized as needed for further analysis.

The script we implemented took two files as input: the text file containing our word pair and the text file containing the list of patterns previously extracted. As already done before, we had to put our word pair inside a hash. That was not a problem, since we used the very same structure implemented in the script that extracted the pattern. Once this was done, we created an output file which contained the results. We had an output file for every different pattern because this made it easy to implement the spreadsheet that we used for our calculations. We then made a loop that scanned the file containing the patterns, and for every word contained in our list, the script checked if there was a string containing the first word of the couple, the pattern and then the second word of our couple. We then had to count, for every couple, how many times it was found with the pattern. Every time this happened, a counting variable was incremented, so that we have the effective number of the co-occurrences. Obviously, we had to increase the counting variable for each separate word pair.

Once this was completed, we printed the output, redirecting it into the text file we made just for this purpose.

Similar to the procedure for extracting the patterns, we had to consider sentences that started with the words after the colon in our list and ended with the words before the colon. To do this we simply exchanged the variables $key and $value in our regex, as already done before. The reason why we did this instead of implementing both the processes in the same script, was because of the overhead the computing load, that would have taken a lot of time and, worse, would have frozen our machine. However, once we extracted both, we summed up the results and placed it in a single file.
The script we implemented could be found in Appendix B, while the pseudo-code for it is the following:

```
FUNCTION patternCount (coupleList, patternList) {

Takes the patternList as Input and counts the occurrences of every pattern, returning
it as output

VAR patternSearch (the pattern we have to count occurrences of)
VAR count

FOR EACH line IN patternList DO
        IF line is made of "$key" followed by a space, a sequence made of ".*",
         patternSearch, another sequence made of ".*" and "$value", increments
         count variable for that pattern with said word pair

END
}
```

With the script we counted all the patterns that co-occurred with every word pair in our pattern list. The Regex we implemented here was meant to count every sentence starting with the $key, followed by a space, a sequence of two char .* meaning every possible word(s) could be there, the actual pattern, another sequence of char identical to the one already mentioned, and the $value. This was the very same format of the patterns obtained with our extraction script.

When the pattern was made up of only a word it was pretty easy to detect that pattern. When there were more than one words, we had to think about something that would have made possible to extract exactly the pattern we were looking for. One example of it could be the pattern “considerare in” (Italian for: “to consider in”). This would have meant having to retrieve (and count) every possible declination of the verb “considerare” and of the preposition “in”. Since every word was POS tagged, and in the form word/lemma/tag, the only part that was possibly going to change and consequentially leading us to problems, was the word. We initially thought of not putting the word, and instead place a .* sequence (meaning every character occurring one or more times) between the lemmas. To tell the truth, while dealing with counting the patterns in the patterns file, this would have worked, since there were no risks of finding some other words in between the two (or more) words composing a pattern.
Still, when searching for the appearance of the very same pattern in all the corpus,

it was possible to find some words in between them and getting our results all messed up. To avoid this, we thought of using the following solution: any words in the sequence word/lemma/pos, was replaced by the string $[A-Za-z\backslash u00C0-\backslash u017F\backslash u1e00-\backslash u1ef9]1$, which meant every character (included accented ones), in sequence of at least 1. Following every POS and before every word we placed a $\backslash s0$, which meant that all that was meant to be found between a word composing a pattern and the next one, was at least a space. This was pretty accurate and worked fine for us.

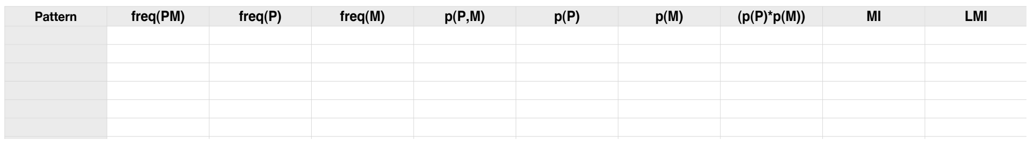
We ended up with one text file for every pattern, each one containing all the couples co-occurring with the given pattern and the number of times it happened. This was not enough for the kind of analysis we had in mind, since we also needed to count all the times every pattern occurred in the corpus with every word pair. To do this we had to implement another Perl script which was similar to this one, but slightly simplified, since what we needed was simply to scan all the corpus file for occurrences of the given pattern and add one to a variable every time the pattern was found. Obviously a loop was necessary to do this kind of counting.

Even in this script, we used the very same regex implemented below to the occurrences of the patterns found. Still, instead of using it with $key and $value, since we were looking for every appearance of every extracted pattern with everything, we exchanged our $key and $value with some wild card, which made it possible to find every time every given pattern that was found in the corpus. At this point we had everything we needed and we were ready to calculate our MI and LMI to see what pattern was more representative for every relation.

```
FUNCTION patternCountCorpus (Corpus, patternList) {

Takes the Corpus as Input and counts the occurrences of every pattern, returning
it as output

VAR patternSearch (the pattern we have to count occurrences of)
VAR count

FOR EACH line IN Corpus DO
        IF line contains patternSearch, it increments count variable for that
        pattern.
        A pattern is made of one of more word/lemma/pos sequences which were
        represented as a sequence of at least a char, followed by a"/" followed
        by another sequence of at least a char, followed again by  a"/" followed by
        another sequence of at least a char. After the POS, it should find a space.


END
}
```

4.3.3 Computing Association Measure and Evaluating the results

We putted all our data in a spreadsheet.
Our empty table looked like this:



Pattern	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI

The first column contained the patterns; the second, third and fourth columns contained all the required frequencies. The fifth, sixth and seventh contained all the probabilities, as described on p. 96. The eighth column contained the product of two probabilities: the probability of finding the pattern and the probability of finding every single word pair. We implemented this column because this product was necessary to calculate both MI and LMI, given the fact that the formula we used to compute this two variables was the following:

$$MI = log_2 \frac{p(P, M)}{(p(P) * p(M))}$$

The ninth column contained all the computed MI values and the tenth contained all the LMI values. Our first spreadsheet, related MI and LMI to every word pair representing the relation we were trying to analyze. This means that, for every word pair, we made a little table containing all the patterns found with that word pair, and organized it for decreasing values of MI or LMI, so that patterns with a high correlation with the couple were high in the table, while patterns with low correlation with the couple were low in the table. The tables containing all the words describing the meronymy relation were ordered according to their MI values. However, the tables describing the location relation were ordered according to their LMI values, because the location data were more sparse and there were more data having very low frequencies, so we found that LMI represented better the relation.

As we predicted, the patterns with the highest rate of MI or LMI were the ones composed of more that one word, which were not very common words as propositions or conjunctions. In fact we could assume that for the meronymy relation, the most representative patterns related to our seeds were short expressions like "composto di" (Italian for: "made of"), "dotato di" (Italian for "provided with") and even "privato di", which is the Italian for "deprived of". As stated according to Zipf's law, words like prepositions or conjunctions, were very low in our table, for every word pair, since they were not really representative of our relation. The short expressions, instead, were representative of the meronymy relation, because they expressed concepts evoking that kind of relation. As a matter of fact, an Italian speaker thinking about the meronymy relation, would probably use some of these short sentences to express it.

Still, this table was not completely clear in defining which pattern better expressed the meronymy relation. To have a table which better expressed the meronymy relation, we had to consider all the words as if they were the same one, to say

MERO. This way, every pattern was related to every word pair meant to be representative of the given relation. Computing MI and LMI between the pattern and this new number indicating all the word pair, basically meant evaluating the strength of a possible connection between the patterns and what we thought was highly representative of the relation. We had to sum every f(M) value, or better the frequency of every single word pair in our table. At this point it was possible to compute the value of MI and LMI that related every pattern to every word pair we thought were more representatives of our relation.

The table representing the patterns expressing meronymy relation showed interesting things. As mentioned the patterns that better expressed the meronymy relations were not prepositions or conjunctions, really low on our table, but instead were verbs or adjectives. According to our calculation, the most representative pattern for the meronymy relation in our corpus was "distinguere tra" (Italian for "distinguish between". This expression was mostly used in our corpus to distinguish different parts composing an object, like for example, in this sentence: "La linguistica distingue tra fonetica e fonologia" which means "Linguistics distinguish between phonetics and phonology". In this sentence, we (meaning Italian speakers) are actually capable of gathering the following conclusion:

Phonetics and phonology are two branch composing Linguistics.

and hence we could say that the previously mentioned pattern is actually good defining the meronymy location.
Observing the table (Appendix A) we could also say that we right in deciding to order this table according to the MI values instead of the LMI, since there were some conjunctions with a high value of LMI but a pretty low value of MI, which is more correct, because a conjunction like "E" (Italian for "and") would not have a really strong connection with the meronymy relation. When examining the location relation, words like these appear even in defining other relations, because obviously there are a lot of such words in a corpus and there would be a lot of them occurring with every kind of seeds we used for our experiments. We felt that the patterns discovered were not bad, and really connected to the meronymy relation,

according to a human judge evaluating them like we did for the “distinguere tra” pattern.

We performed the same procedure for the location relation, trying to determine patterns that were representative of it. We first produced a table containing data which showed the correlation between every single word pair and the pattern it was found co-occurring with. Then we produced a table that expressed just the correlation between the patterns and the relation. The procedure we followed was the same as described for the meronymy location but instead of organizing the tables for descending values of MI, we decided to organized them for descending values of LMI, because this time, there were a lot of sparse data, that we did not have in the previous case.

Even in this case we could see that words like conjunctions or prepositions were not much representative of the relation, while obviously verbs or short sentences represented really well the relation we had to examine, or at least were representative of the relation which connected every word pair. For example, “trasferire a” (Italian for: “transfer to”), “essere custodito in” (Italian for “being kept in”) or even “a nord di” (Italian for “north to”) were really representative words for the location relation, according to the definition of *location* we gave. That is, the location relation occurs when a nominal *is collocated* somewhere (and this somewhere is represented by another nominal). In this case, all the patterns we extracted were somewhat expressing this type of connection, because we could say that, if something got transferred from one place to another, it then was collocated in the place it has been transferred to. Similarly, if something is being kept somewhere, it has to be collocated there. In the same way, if something is north to another thing, it means it is actually collocated north to it.
Once all the patterns were obtained, we had to compute the degree of association between every pattern and the location relation. As with the meronymy relation, we had to sum the frequency of all the different word pair. This way we had the frequency of every location couple, or better, of every location relation we extracted from our corpus using our seeds. Using MI and LMI, it was possible to determine which pattern (or whose patterns) better represented the relation.

We found that between the most highly ranked patterns, it was possible to find “ad oriente di” (Italian for “east to”), “localizzare su” (Italian for “localize on ”) and “venire collocati ” (Italian for “being collocated”), that is, short sentences highly representative of our relation. Unfortunately, there were some patterns that were highly ranked, but were not really representative of the location relation, such as “risucchiare”, (“suck in”). This kind of problem (even if, fortunately, in our experiment misplaced patterns were really rare) was a consequence of using seeds for harvesting corpora for semantic relation. Probably the pattern occurred really frequently with a couple or more of our seeds, that is why it has been selected as a representative pattern for the location relation.

Our technique for extracting patterns representing a given relation was based on different steps which could be summed up as follows. First, we have to create a list of word pair (seeds) known to be connected by the relation we want to examine. These words were selected by human judges, and the selection was based only on their judgment, without any other condition. We then searched for every occurrence of every couple in our corpus and extracted the words in between, to see which kind of connection they have. These words were our patterns. But obviously, not every pattern had the same degree of pertinence with our relations. There were patterns that really represented the given relations and patterns that tended to occur with every possible relations, just because they were made of conjunctions or prepositions. Conjunctions and prepositions are words known to be frequently occurring in every kind of text, and so, even in our corpus, they would occur frequently without having a very representative meaning. This meant that probably, words like these would be found frequently with our seeds, would be found frequently as patterns but would be not very representative of our relations. We were pretty sure of this, especially because we had a first proof of this in Zipf’s law. However, we had to find which were the most representative patterns. To do this, we used a quantity that measures the mutual dependence of two variable. We wanted to understand which pattern was more related to every seed pair, and then what pattern was better related to the relation being examined. To do this, we had to compute the Mutual Information (which was the chosen measure) between

every pattern and the seeds it was found with.
Every obtained data was put into a table which was ordered according to descending values of MI and LMI. This way we had the more representative patterns on top of our table. After this, we computed the association between the patterns and the given relations. To do this, we had to consider every seed as part of the relation, and sum up the frequency of every seed pair. This way we determined the frequency of the relation in the corpus, because we are sure that between given seeds we have a certain type of relation, the one we are trying to express, and this is the only way of counting how frequently the relation occurs in the corpus. So it is possible to compute the association between the relation and every pattern, using the same association measure seen before, the Mutual Information. At this point, we have a table with the more determining patterns on top and the less determining patterns on bottom. Not only what we did made it possible to determine what patterns better expressed the relation we wanted, but we also used this as the first step for another algorithm which was made for relation classification.

Chapter 5

A relation classification model implementing Vector Space Model

In the previous step of our algorithm, we developed a system capable of determining which pattern better described any relations we examined. But we also wanted to do something different. In the previous experiment we selected some word seeds and a given relation known to occur between the words, and we extracted some patterns and then determined how they were representative of the relation. We wanted to see if it was possible to develop another step of the algorithm that, given a list of words and some relations, was capable of effectively determining specifically which relation exists between them.

As we described in the second chapter, the classification of semantical relation has become an important and very much explored task in NLP. In this work, we tried to develop our own algorithm for this task.

We decided to use a model very frequently used in Information Retrieval, Vector Space Model, and to adapt it to our purpose, given the fact that we did not have to work on document and terms, but on terms and relations. So, we changed the Vector Space Model a little, adapting it for our purpose.

5.1 Vector Space Model

Vector space model (or *term vector model*) is an algebraic model for representing text documents (and any object, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings. Its first use was in the SMART Information Retrieval System.

A document is represented as a vector. Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. One of the best known schemes is tf-idf weighting. The definition of term depends on the application. Typically terms are single words, keywords, or longer phrases. If the words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus).

Relevancy rankings of documents in a keyword search can be calculated, using the assumptions of document similarities theory, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as the same kind of vector as the documents.

In practice, it is easier to calculate the cosine of the angle between the vectors instead of the angle, so cosine similarity is used.
Cosine similarity is a measure of similarity between two vectors of n dimensions, computed by finding the cosine of the angle between them, often used to compare documents in text mining. Given two vectors of attributes, A and B, the cosine similarity, θ, is represented using a dot product and magnitude as:

$$cos\theta = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating independence because a 0 value means that the

query and document vector are orthogonal and have no match, probably because the query term is non-existent in the corpus being considered, and in-between values indicating intermediate similarity or dissimilarity.
Obviously, like every other model, the vector space model has its limitations. First of all, long documents are poorly represented by this model, because they have poor similarity values. Long documents produce a small scalar product and consequentially a large dimensionality. The second problem is that the search keywords must precisely match document terms; if this does not happen, word substrings might result in a "false positive match", which obviously gives inaccurate answers to every queries. Also, the vector space model suffers semantic sensitivity. Basically, this means that documents with similar context but different term vocabulary will not be associated, resulting in a "false negative match". Lastly, the order in which the terms appear in the document is lost in the vector space representation.

The vector space model procedure can be divided into three stages. The first stage is document indexing where content bearing terms are extracted from the document text. The second stage is the weighting of the indexed terms to enhance the retrieval of documents relevant to the user. The last stage ranks the documents with respect to the query according to a similarity measure. It is obvious that many of the words in a document do not describe the content, for example words like the, is (we already covered this topic in the previous chapter explaining why this happens, and explaining Zipf's Law). By using automatic document indexing those non significant words (function words) are removed from the document vector, so the document will only be represented by content bearing words. This indexing can be based on term frequency, where terms that have both high and low frequency within a document are considered to be function words. In practice, term frequency has been difficult to implement in automatic indexing, a stop list is more commonly used instead. A stop list holds common words to remove high frequency words (stop words), which makes the indexing method language dependent. In general, 40-50% of the total number of words in a document are removed with the help of a stop list. Non linguistic methods for indexing have also been implemented.

Probabilistic indexing is based on the assumption that there is some statistical difference in the distribution of content bearing words, and function words.

Recently, an automatic indexing method which uses serial clustering of words in text has been introduced. The value of such clustering is an indicator if the word is content-bearing. Term weighting has been explained by controlling the exhaustivity and specificity of the search, where the exhaustivity is related to recall and specificity to precision. The term weighting for the vector space model has entirely been based on single term statistics. There are three main factors for term weighting: term frequency factor, collection frequency factor and length normalization factor. These three factors are multiplied together to make the resulting term weight.

A common weighting scheme for terms within a document is to use the frequency of occurrence as stated by Luhn in his 1958 article "The automatic creation of literature abstract"[1]. The term frequency is somewhat content descriptive for the documents and is generally used as the basis of a weighted document vector. It is also possible to use binary document vector, but the results have not been as good compared to term frequency when using the vector space model.
There are various weighting schemes used to discriminate one document from the other. Most of these, e.g. the inverse document frequency, assume that the importance of a term is proportional with the number of document the term appears in. Experimentally it has been shown that these document discrimination factors lead to a more effective retrieval, so they lead to an improvement in precision and recall.

The third possible weighting factor is a document length normalization factor. Long documents have usually a much larger term set than short documents, which makes long documents more likely to be retrieved than short documents. Different weight schemes have been investigated and the best results, both in recall and precision, are obtained by using term frequency with inverse document frequency and length normalization.

[1]H.P: Luhn. 1958. *The automatic creation of Literature Abstracts* In:Advances in automatic text summarization. 1999. pp. 15-21

The similarity in vector space models is determined by using associative coefficients based on the inner product of the document vector and query vector, where words overlap indicates similarity. The inner product is usually normalized. As already explained, the most popular similarity measure is the cosine coefficient, which measures the angle between the document vector and the query vector.

5.2 Our Idea

To test our hypothesis, we decided to start from the relations we had already described in our previous experiment. We knew exactly how our algorithm described meronymy and location relation and which patterns better described them according to our algorithm.

Starting from there, we tried to understand if two nominals constituting a seed pair, were connected by a meronymy relation or a location one. We used the results of the previous experiment as a training set, to train our variation of the Vector Space Model and to build up vectors. In Vector Space Model, a document is represented as a vector, each dimension corresponding to a separate term. If a term occurs in the document, its value in the vector is non-zero. We would like to do the same, but instead of considering documents, we had to consider relations and to build a vector representing them.

What we had in mind was building a n-dimensional vector for every relation, using the data obtained by the previous experiment. In other words, for every relation we had a certain number of patterns, each of them connected to the relation by a value of Mutual Information or Local Mutual Information, describing the degree of association between the given association and the pattern. Basically, each dimension of this kind of vector, was a previously-extracted pattern, both for the meronymy and the location relation. In this case, the weight value associated to every pattern could be the MI or the LMI value that connected the same pattern to every relation.
We placed all the extracted patterns in every vector, not repeating the ones in

common. For every relation we created two vectors, one built using MI as weighting scheme, associated to every pattern (the pattern was considered some sort of “label” for the corresponding vector cell) and one built using LMI values. Since every vector had a cell for every possible pattern found, we would have a “non-zero” value if the pattern was found for the examined relation, and a “zero” value if no pattern was found for that relation.

We already had our training set completed, since the algorithm was trained on how an vector representing a certain relation should look like. We could now build a test set. To do that we selected some couple of nominals, knowing to be associated to one or the other relation to be used as seeds, and then built an vector for each of them. Then we measured the cosine between the previously built vector and the currently obtained vector, to measure their similarity. This should tell us how similar every couple is to every relation, and consequently which kind of relation was the one connecting the two words in the couple. This way we modified Vector Space Model and adapt it, to make it work with the kind of data we had.

5.3 Preparing the training set

We built our training set using the previously selected relations (meronymy and location). In Vector Space Model, vectors are usually built using words describing the document that should be analyzed as the vector dimensions. This made no sense in our case and we decided to do something different. We used the extracted patterns as the dimensions of our vectors. So, our relations were described by n-dimensional vectors, where n was the total numbers of patterns extracted for both the relations.

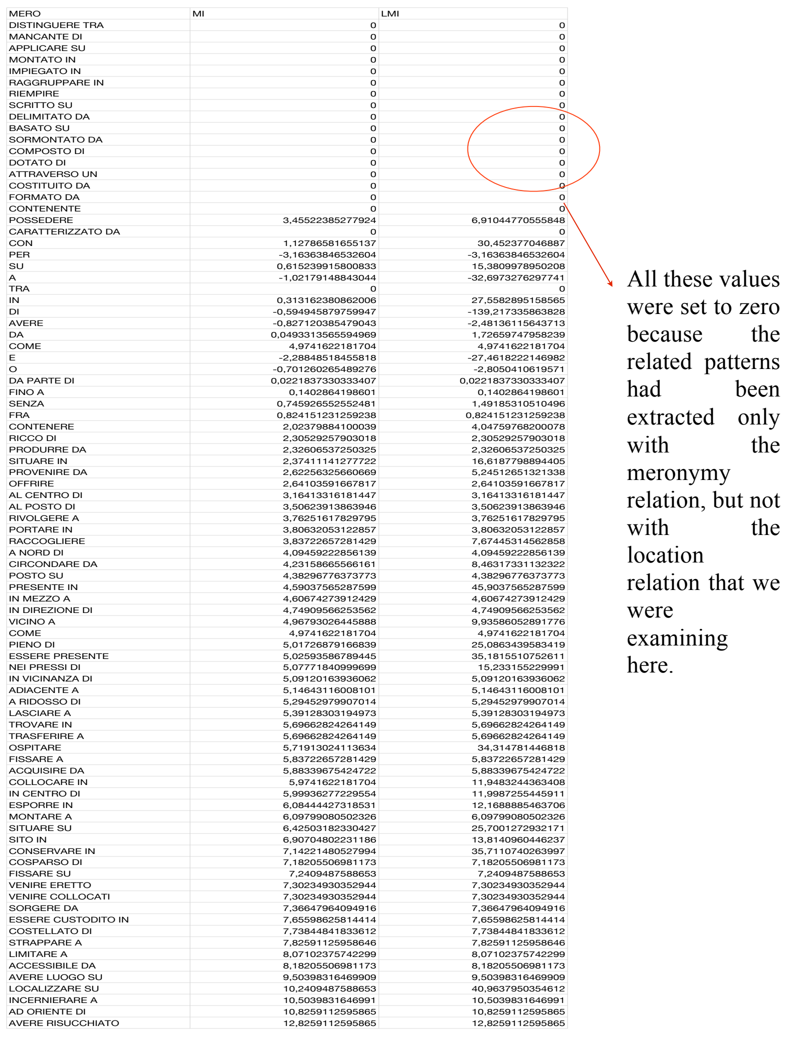
The patterns found to be representative both of the meronymy and the location relation, were not repeated, that is, they were placed in every array just once.
To build our vector this way, we had to take the tables containing the extracted patterns and examine which patterns occurred with both the relation, to place it

just once inside every vector. In fact, to build our vectors we used a technique that was probably not commonly recommended, but it worked for us.

We built a table containing all the patterns extracted from the previous analysis, basically copying and pasting the two tables obtained and shown in chapter two, and merging them into one.

We examined all the common pattern values, in order to do not repeat the shared ones in our vectors. Once we did this, we had a table that contained all the patterns we extracted in our previous analysis, every one of them appearing in each vector just once. We use the skeleton of this table as the base to build our vectors. We erased all the non-needed fields, maintaining just the field containing the MI value and the one containing the LMN value, with the label field. At this point, we placed the previously obtained values in their field. We had to make two tables. One for the meronymy relation with both the MI and the LMI values and one for the location relation and the MI and LMI values.

We then had two tables. Obviously, tables representing the meronymy relation would have all zero-values for the patterns representative only of the location relation, and conversely, tables representing the location relation would have all zero-values for patterns representative only of the meronymy relation. Basically, a table containing our patterns, would look like the following:



MERO	MI	LMI
DISTINGUERE TRA	0	0
MANCANTE DI	0	0
APPLICARE SU	0	0
MONTATO IN	0	0
IMPIEGATO IN	0	0
RAGGRUPPARE IN	0	0
RIEMPIRE	0	0
SCRITTO SU	0	0
DELIMITATO DA	0	0
BASATO SU	0	0
SORMONTATO DA	0	0
COMPOSTO DI	0	0
DOTATO DI	0	0
ATTRAVERSO UN	0	0
COSTITUITO DA	0	0
FORMATO DA	0	0
CONTENENTE	0	0
POSSEDERE	3,45522385277924	6,91044770555848
CARATTERIZZATO DA	0	0
CON	1,12786581655137	30,452377046887
PER	-3,16363846532604	-3,16363846532604
SU	0,615239915800833	15,3809978950208
A	-1,02179148843044	-32,6973276297741
TRA	0	0
IN	0,313162380862006	27,5582895158565
DI	-0,594945879759947	-139,217335863828
AVERE	-0,827120385479043	-2,48136115643713
DA	0,0493313565594969	1,72659747958239
COME	4,9741622181704	4,9741622181704
E	-2,28848518455818	-27,4618222146982
O	-0,701260265489276	-2,8050410619571
DA PARTE DI	0,0221837330333407	0,0221837330333407
FINO A	0,1402864198601	0,1402864198601
SENZA	0,745926552552481	1,49185310510496
FRA	0,824151231259238	0,824151231259238
CONTENERE	2,02379884100039	4,04759768200078
RICCO DI	2,30529257903018	2,30529257903018
PRODURRE DA	2,32606537250325	2,32606537250325
SITUARE IN	2,37411141277722	16,6187798894405
PROVENIRE DA	2,62256325660669	5,24512651321338
OFFRIRE	2,64103591667817	2,64103591667817
AL CENTRO DI	3,16413316181447	3,16413316181447
AL POSTO DI	3,50623913863946	3,50623913863946
RIVOLGERE A	3,76251617829795	3,76251617829795
PORTARE IN	3,80632053122857	3,80632053122857
RACCOGLIERE	3,83722657281429	7,67445314562858
A NORD DI	4,09459222856139	4,09459222856139
CIRCONDARE DA	4,23158665566161	8,46317331132322
POSTO SU	4,38296776373773	4,38296776373773
PRESENTE IN	4,59037565287599	45,9037565287599
IN MEZZO A	4,60674273912429	4,60674273912429
IN DIREZIONE DI	4,74909566253562	4,74909566253562
VICINO A	4,96793026445888	9,93586052891776
COME	4,9741622181704	4,9741622181704
PIENO DI	5,01726879166839	25,0863439583419
ESSERE PRESENTE	5,02593586789445	35,1815510752611
NEI PRESSI DI	5,07771840999699	15,233155229991
IN VICINANZA DI	5,09120163936062	5,09120163936062
ADIACENTE A	5,14643116008101	5,14643116008101
A RIDOSSO DI	5,29452979907014	5,29452979907014
LASCIARE A	5,39128303194973	5,39128303194973
TROVARE IN	5,69662824264149	5,69662824264149
TRASFERIRE A	5,69662824264149	5,69662824264149
OSPITARE	5,71913024113634	34,314781446818
FISSARE A	5,83722657281429	5,83722657281429
ACQUISIRE DA	5,88339675424722	5,88339675424722
COLLOCARE IN	5,9741622181704	11,9483244363408
IN CENTRO DI	5,99936277229554	11,9987255445911
ESPORRE IN	6,08444427318531	12,1688885463706
MONTARE A	6,09799080502326	6,09799080502326
SITUARE SU	6,42503182330427	25,7001272932171
SITO IN	6,90704802231186	13,8140960446237
CONSERVARE IN	7,14221480527994	35,7110740263997
COSPARSO DI	7,18205506981173	7,18205506981173
FISSARE SU	7,2409487588653	7,2409487588653
VENIRE ERETTO	7,30234930352944	7,30234930352944
VENIRE COLLOCATI	7,30234930352944	7,30234930352944
SORGERE DA	7,36647964094916	7,36647964094916
ESSERE CUSTODITO IN	7,65598625814414	7,65598625814414
COSTELLATO DI	7,73844841833612	7,73844841833612
STRAPPARE A	7,82591125958646	7,82591125958646
LIMITARE A	8,07102375742299	8,07102375742299
ACCESSIBILE DA	8,18205506981173	8,18205506981173
AVERE LUOGO SU	9,50398316469909	9,50398316469909
LOCALIZZARE SU	10,2409487588653	40,9637950354612
INCERNIERARE A	10,5039831646991	10,5039831646991
AD ORIENTE DI	10,8259112595865	10,8259112595865
AVERE RISUCCHIATO	12,8259112595865	12,8259112595865

All these values were set to zero because the related patterns had been extracted only with the meronymy relation, but not with the location relation that we were examining here.

FIGURE 5.1: Values of the location vector

What we did at this point was to "transform" these tables into text values that could be easily adapted to become Perl vectors. We then eliminated the column containing all the pattern labels, and one between the MI or the LMI one. This table, made of just one column containing MI or the LMI value, would then be exported using the "Export as CSV" function natively included in Apple Numbers. CSV means "Comma Separated Values" and its output could be treated as a text file, with all values separated by a comma, which we could easily transfer into the vector we needed. All we had to do was just operate a simple "find and replace" exchanging all the commas with semicolons to match the Perl syntax required for building vectors. After exporting the column containing the, say, MI values, we just had to undo the previous column-deleting operation.

Using this technique was useful because we could easily maintain the element order, to build vectors that had the same object in the very same position. In fact, this technique made it possible for us to build a table that worked as a skeleton for the position every element should have in the table. It was pretty easy to re-use it for every word pair taken in analysis, all we had to do was change all the values in the table and then, again, delete the "label" column and one between the MI and LMI column alternatively. The received output were placed into two different folders, according to the value that was examined (one folder for the MI and another one for LMI).

In the end we had four vectors that could be considered our "Training Set", built from the results of the previous experiment. These vectors could be considered our training set because they were the "reference" vectors, the ones that determined how a given relation should look like and the ones which the others vectors (the ones representing different word pair) should be compared to in order to discover which relation was shared by each given couple.

5.4 Preparing the test set

What we needed to test now was the accuracy of our algorithm in determining if a given word pair shared a meronymy relation or a location relation.

To do this, we needed a list of seed nominals to analyze, or in other words, a test set. To choose the words, we used the same criterion already used in the previous experiment. The words were selected by a human judge,based on her personal judgement and corresponding to the frequency the words occurred in the corpus. This means that the selected seeds were known to be at least accounted for in our corpus. We had to select words known to be accounted for the meronymy and for the location relation, just like we did in the previous experiment.

There were no automated process for selecting the words yet, but as described in next chapter it is possible to implement an algorithm that automatically selects automatically words thought to be related to one specific relation. For now, we needed to focus on the current research problem, and decided to take the easiest approach to it. We had to be sure the words were contained in our corpus, or there would have be no pattern extraction related to it and, consequently, no possibility of comparing the words with the referring vectors.
We selected a list of about 60 words, half of which were related to the meronymy relation and the other half to the location relation. Some of them were inserted purposely inside this list because they were ambiguous, that is, they could be easily misinterpreted and easily associated to the wrong class. For example, a word that was classified as member of the meronymy class could be easily misinterpreted and associated to the location relation. As described in the first chapter, lexical relations could easily be ambiguous, and here we have some examples of this characteristic.
A specific example could be the couple: “Circoscrizione:Comune” (italian for “area:municipality”) which was placed as a location relation, but could also be interpreted as a meronymy relation since the area is a part of the municipality. Ambiguous entries like this one were good to effectively evaluate the capability of our algorithm, because classifying non-ambiguous couples was much easier than classifying ambiguous ones, and we were interested in seeing how this algorithm was going to handle every kind of words we were feeding it.

Once we selected the nominals constituting our seed pair, we extracted all the patterns concerning every couple, just like we did previously. We used the very same script for extracting them, since it proved to be effective enough for our purpose.

We extracted a consistent number of patterns, enough for what we meant to do with them. For every of the patterns we had to compute the relative MI and LMI values, that not only gave us the degree of association between the two words for every couple, but was also necessary for building the vectors used for evaluating the similarity between every couple and the relations. The extracted patterns were put in tables with their values of MI and LMI, in order to have them arranged and kept in store for further re-use, that was the best choice since we would have to re-use these data to build our vectors.

Once we gathered all these data, we built a vector for every word pair. As previously explained, we used a table, to place all our values. If a word pair was not found with a specified pattern, the field in the table that corresponded to that pattern, would report a "0" value, if the pattern was found to occur with the word pair, the field in the table would contain the MI (or LMI, according to which set of arrays we were building) value found regarding that couple and the pattern.

Again, every table obtained for every single word pair, was then transformed into two vectors, one containing all MI values, and one containing LMI values. These vectors were compared with the reference vectors using a VSM-like algorithm, which could tell us the closeness between them and, consequently, could recognize the relation shared between the words.

5.5 The modified VSM : the Perl Script

The next step was to develop the Perl script that would enable us to actually compare the reference vectors (the ones obtained by the algorithm against the training set) with every vector obtained in the corpus against each word couple.

To do this, we implemented the vectors inside a Perl Script. However we also have to normalize them and to enable comparisons to be made. This was done using a Perl Package, a set of Perl instructions grouped and organized together.
In this Script we implemented the PDL module. PDL (which stands for Perl Data Language) is used to give Perl the ability to compactly store and speedily manipulate large N-dimensional vectors, according to the official web site.
PDL turns Perl in to a free, vector-oriented, numerical language similar to such commercial packages as IDL and MatLab. One can write simple perl expressions to manipulate entire numerical vectors all at once. For example, using PDL the

Perl variable $a can hold a 1024x1024 floating point image: it only takes 4MB of memory to store this data and expressions like $a=sqrt($a)+2 can manipulate the whole image in a few milliseconds. The module is freely downloadable both on Sourceforge and CPAN, which is the larger online repository of Perl Modules.

We implemented this module to be able to work with big n-dimensional vectors quickly, since each one of our vectors was about 100-dimensional.
What we wanted our script to do was take as input three vectors, two being the referring vectors (one representing the meronymy relation and the other representing the location relation) and the last being the one that represented the seeds. We wanted the latter to be compared to the former two, measuring the cosine in between the normalized vectors and selecting as the chosen relation the one represented by the vector which was closer to the one representing the seeds. We made two different Perl scripts, one implementing the vectors obtained using MI values as weighting scheme, and the other one obtained using LMI values.

Both scripts should obviously take as input the reference vectors and the one representing the seeds and then measure the cosine between the last and the other two. After that, each script should report the two values found by comparing the vectors, so that we could use them to see which relation, every seed couple was closer to, and consequentially which the relation the two words shared. The pseudo-code for this script is the following, while the actual script we made is found in Appendix B.

```
FUNCTION cosineComputing (meronymyArray, locationArray, seedArray) {

Takes as Input three arrays (two representing semantical relation, one representing
a couple of seed, and returns as output the cosine value of the angle between
every relation array and the seed array

VAR cos1
VAR cos2

cos1= Cosine of the angle between meronymyArray and seedArray
cos2=Cosine of the angle between locationArray and seedArray
}
```

The script performed as expected. It took the vectors as input, even if what we used here is not exactly an vector, but instead is a piddle. The problem with using Perl vectors here is that they would not scale. Perl vectors consume a lot of memory, and there are no native functions for comparing vectors to one another. We would have to loop through our vectors, which is slow.

A pidlle is a structure, introduced by the PDL module, that basically is a numerical vector stored in column major order (meaning that the fastest varying dimension represent the columns following computational convention rather than the rows as mathematicians prefer). Even though, piddles look like Perl vectors, they are not. Unlike Perl vectors, piddles are stored in consecutive memory locations.

Piddles are referenced with a leading $. We created them by writing the command piddle, followed by the piddle, which is probably the easiest approach. Now, that we had all our vectors set up, we just needed to compute the closeness between the reference vectors (meronymy and location) and the vector representing the seeds we were testing. There are many ways to do this. One of the simplest (and most intuitive) is the cosine measure. Taking the cosine of that angle gives us a value from zero to one, which is handy. Vectors with no relations shared will have a cosine of zero; vectors that are identical will have a cosine of one. Partial matches will have an intermediate value - the closer that value is to one, the more similar the vector representing the seeds is to that relation. The formula for calculating the cosine is this:

$$cos = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$$

where V2 and V2 are our vectors, the vertical bars indicate the 2-norm, and the $\cdot$ indicates the inner product. We then normalized the three vectors using the norm command. Norm normalizes a vector to Euclidean length. Normalizing in Vectorial Space basically means that, given a vector, that vector is taken to have a unitary norm. Normalizing a vector is useful for simplifying the calculations needed to be done. We can normalize the vectors to unit length using the norm function, because we were not interested in their absolute magnitude, only the angle between them.

Once done with computing the inner product between the two reference vectors and the one representing the seeds, we had to re-convert the PDL object obtained into a Perl scalar. These two values were the cosine between the meronymy vector and the seeds vector and the cosine between the location vector and the seeds vector. We made the Perl program print these values into a file (which was named after the seeds) for further examination. We decided to have one different file for every seed pair. This was done because we would have to put our results into a spreadsheet for evaluation, and having a separate file for every result was a way of keeping everything ordered. The script computed the cosine of the angle between the vectors very fast.
Although we referred to the script that used LMI vectors, there was also a duplicate script using MI vectors.
Once all the results were gathered, we examined them to see if this modified version of the VSM could be a effective way of classifying relations.

5.6 Results Evaluation

In order to evaluate our script, we had to see how it worked with our vectors. This was done by observing our results to see how the vectors built for every seed pair was connected to every relation, and which relation a seed pair was closer to. This would determine which relation every seed pair was associated with, because the vector it was closer to, would be the one indicating the relation that the two seeds shared, according to the algorithm.
As previously explained, the cosine measure of the angle in between two vectors, could easily indicate their closeness. Vectors with no relations at all will have a cosine measure of 0, while two completely identical vectors will have a cosine

measure of 1. This means that, the more similar the two vectors are, the cosine measure of the angle between them would be closer to 1, while, if two different vectors had very little in common, they would have a cosine value close to 0. Knowing this enable us to recognize if our algorithm worked correctly, i.e., if it recognized correctly the relation, or not.

This evaluation was made by a human judge who classified the seeds according to the relation they shared, and then evaluated the results obtained with the Perl script, using his own judgement. This way, it was possible to measure the error percentage as well as to evaluate the precision, recall and f-measure of the algorithm. In particular, the human judge evaluated the performance of the algorithm in terms of its preciseness in classifying the semantic relation of patterns extracted in association with our seed pair.
We evaluated both the results obtained using and implementing MI as the weight measure, and the results obtained using LMI, to see which measure worked better with our system and produced better results.

5.6.1 The measures: Precision, Recall and F-measure

Precision and Recall are two widely used statistical classifications. Precision can be seen as a measure of exactness or fidelity, whereas Recall is a measure of completeness.

In a statistical classification task, somewhat like ours, the Precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the class) divided by the total number of elements labeled as belonging to the class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to that class but should have been).

Precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a Recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C).

Often, there is an inverse relationship between Precision and Recall, where it is possible to increase one at the expense of reducing the other. For example, an information retrieval system (such as a search engine) can often increase its Recall by retrieving more documents, at the cost of increasing the number of irrelevant documents retrieved (decreasing Precision). Similarly, a classification system for deciding whether or not, say, a fruit is an orange, can achieve high Precision by only classifying fruits with the exact right shape and color as oranges, but at the cost of low Recall due to the number of false negatives from oranges that did not quite match the specification.

Usually, Precision and Recall scores are not discussed in isolation. Instead, either values for one measure are compared to a fixed level of the other measure (e.g. precision at a recall level of 0.75) or both are combined into a single measure, such as the F-measure, which is the weighted harmonic mean of precision and recall. In the context of classification tasks, the terms true positives, true negatives, false positives and false negatives are used to compare the given classification of an item (the class label assigned to the item by a classifier) with the desired correct classification (the class the item actually belongs to). True positive actually means that a value, meant to be positive, was recognized and found to be positive by our algorithm. The same happens with true negative. A value which is recognized to be negative by our algorithm, actually was meant to be negative. False positive and false negative are a little more complicated. Type I error, also known as an "error of the first kind" , or a "false positive" is defined as the error of rejecting a null hypothesis when it is actually true. Plainly speaking, it occurs when we are observing a difference when in truth there is none. Type I error can be viewed as the error of excessive credulity.

Type II error, also known as an "error of the second kind", or a "false negative" is defined as the error of failing to reject a null hypothesis when it is in fact not true. In other words, this is the error of failing to observe a difference when in truth there is one. Type II error can be viewed as the error of excessive skepticism. To be more clear, imagine that a pregnancy test has produced a "positive" result (indicating that the woman taking the test is pregnant); if the woman is actually not pregnant though, then we say the test produced a "false positive". Instead a type II error occurs if a pregnancy test reports "negative" when the woman is, in fact, pregnant. We implemented a table like this one:

		Correct Results/Classification	
		E1	E2
Obtained Results/Classification	E1	True Positive	False Positive
	E2	False Negative	True Negative

Reporting all the values we found to be true positive, true negative, false positive and false negative for every single relation. Precision could be computed as :

$$Precision = \frac{tp}{tp + fp}$$

the numbers of true positive is divided by the sum of true positive and false positive. Recall could be computed as:

$$Recall = \frac{tp}{tp + fn}$$

the numbers of true positive divided by the sum of true positive and false negative. As we already said before, a popular measure that combines Precision and Recall is their harmonic mean, also know as F-measure, which is computed as follows:

$$F - Measure = 2 \cdot (precision \cdot recall)/(precision + recall)$$

5.6.2 Our Evaluation

We gathered all the information we needed and built two tables, the first one used to analyze all the values we discovered using the MI as our weight measure and the second one meant to analyze the values discovered using the LMI as our weight measure.

We put every value into a table, with 5 columns. The first contained all the different seed pair. The second contained the relation that was expected to be found, related to every seeds, according to our human judge. The third and fourth columns reported the cosine value that was found, for every seed pair in relation to the meronymy relation and the location relation respectively. The fifth column told us if the results provided by our algorithm were corrected or not, according to the values reported in the previous columns. If a seed pair was classified by our human judge as a member of the meronymy relation, we would expect to find a higher value in the meronymy column and a lower one in the location column.

If this happens, we would say that the relation has been correctly classified by our algorithm, and we would place a “si”́ (yes) value in the last column for that particular seed pair. Instead, if the seed pair was classified as member of the meronymy relation, but we found an higher value in the location column, it means that our algorithm has mistakenly classified the relation, and we would place a “no” value in the last column for that particular seed pair. This last column was meant to be used for computing the percentage of correct answer, but it was also useful for helping us find true and false positive and true and false negative. Using the MI we obtained pretty good results (the table reporting all the results can be seen in Appendix A), but, to be more accurate, we computed the following table meant to highlight the percentage of correct answers versus the wrong ones:

			Percentage
Correct Answer	27	37	72.972972972973
Wrong Answer	10	37	27.027027027027

As we can see here, classifying 37 seed pair we obtained 27 right classification versus 10 wrong classification, with a percentage of correct answer of almost 73%.

This result is not bad, given that our algorithm had a 50% probability of classifying our relation the wrong way.

Still, some of the seeds that were wrongly classified were somewhat ambiguous even for the human judge. For example, the seeds canzone-album (song-record) were classified as members of the meronymy relation, because usually a record is made of some songs, but still, one could think of a song as located or placed into a record. The same can be said about all the misplaced location relation (the ones that were recognized as meronymy), for example paese-casale (village-farmhouse). In fact is true that a farmhouse is placed or collocated into a village, but is also true that a village is made up of some farmhouses.

So we could say that the error percentage reported by our algorithm is influenced by the ambiguity of some of the chosen seeds, and that, probably, using less ambiguous seeds would have lead to more accurate results. Still, it is interesting to see how the algorithm was capable of dealing with more “difficult” words.

After computing this percentage, we also decided to compute precision, recall and f-measure for both the meronymy and the location relation. This way we could evaluate which relation was better handled by our classifier. We built a table as the one seen in the previous paragraph, with all the counting for true and false positive and true and false negative. We counted as true positive all the times that a seed was associated correctly to a relation, and in the same way we counted as true negative all the times that a seed pair was not associated with a relation which actually was not the one shared between the said seeds. We counted as false positive, all the times the seeds were associated to a relation which was not the right one, while we classified as false negative all the times a seed pair was not classified as not having the relation that was actually shared between them. The results for the meronymy relation are the following:

Meronymy

		Correct Answer	
		P	**N**
Gathered Result	**P**	14	8
	N	2	13

Precision	0.636363636
Recall	0.875
F-Measure	0.736842105

while for the location relation we had:

Location

		Correct Answer	
		P	**N**
Gathered Result	**P**	13	2
	N	8	14

Precision	0.8666666667
Recall	0.619047619
F-Measure	0.7222222222

We can see here that the algorithm found less difficult classifying meronymy relation, while it was a little less accurate in classifying the location one. Anyway, the difference is subtle and, as we already said before, could also depend on the ambiguity of the chosen seeds for every relation.

Our work did not end here, because we still had to evaluate the results obtained using the LMI as weight measure implemented for building the vectors. These results could be found in Appendix A, while the percentage of good versus bad classification is the following:

			Percentage
Correct Answer	29	37	78,378378378378
Wrong Answer	8	37	21,621621621622

From these results we can see that using LMI instead that MI improved the way our algorithm worked, since it increased the correct percentage of a good 6%, given that it was 73% using MI and almost 79% using LMI. The difference in our results may be due to the fact that LMI tends to remove the noise that could be found with low frequencies, as discussed in the previous chapter. Some of the non-recognized words are the same, but some are different, and LMI seems to work better than MI with location relation, which instead, provided more wrong results for the location relation.

If we examine precision, recall and f-measure obtained using LMI, the fact that LMI worked better for the location relation is even clearer. These are the results for the meronymy relation:

Meronymy

		Correct Answer	
		P	**N**
Gathered Result	**P**	13	4
	N	4	16

Precision	0,764705882
Recall	0,764705882
F-Measure	0,764705882

And these are the results for the location relation:

Location

		Correct Answer	
		P	**N**
Gathered Result	**P**	16	4
	N	4	13

Precision	0,8
Recall	0,8
F-Measure	0,8

Contrary to what reported using MI, using LMI our algorithm found less difficult classifying location relation, while it was a little less accurate in classifying the meronymy one. Also, the f-measure values reported are higher, meaning that LMI was more appropriate for evaluating these two relations, obviously relating to the data-set we had, and especially related to the the training set we built and the data sparseness reported.

Chapter 6

Testing our model to recognize three different relation: two trials

After the results obtained with our relation recognizer, we decided to test it on different relations, to see how well it performs.

We took inspiration from the SemEval 2007 experiments and chose three out of the seven relations they selected for their experiments. The chosen semantic relations for the SemEval task were the following: Cause-Effect, Content-Container, Instrument-Agency, Origin-Entity, Part-Whole, Product-Producer and Theme-Tool, with seven detailed denitions, including restrictions and conventions, plus prototypical positive and near-miss negative.
Since we already covered the part-whole relation in the previous step of our algorithm, we decided to focus on Cause-Effect, Instrument-Agency and Product-Producer.

We defined Cause-Effect as the relation intercourring between two nominals (X and Y) in a sentence , when, according to common sense, the situation described in the entails that X *is the cause of* Y.
Basically, we could say that cause-effect relation is a relation between cause-concept and effect-concept. For example if we say, *"Death" from "inhalation" of petroleum distillates is well recognized in misuses of volatile substances."*, we

could deduce that *death* is caused by *inhalation*, and so that *death* and *inhalation* are connected by a cause-effect relation

The Instrument-Agency relation was defined as occurring between two words X and Y when, according to common sense, the situation described in a sentence entails the fact that X *is the instrument (tool)* of Y or, equivalently, Y *uses* X. For example, the sentence *"The dentist got the drill ready to begin work"* implies that the dentist (Agency) will likely use the drill (Tool), and so that dentist and drill are connected by a Instrument-Agency relation.

The Product-Producer relation already had some definitions.
Girju et al. (2005)[1] named this relation "MAKE/PRODUCE" and defined it as "an animated entity creates or manufactures another entity". They had two versions of it, depending on the directionality. For example in "honey bee" the modifier (honey) is the product and the head (bee) is the producer. In "GM car" the direction is reversed the modifier (GM) is the producer and the head (car) is the product.
Levi (1979)[2] also had two versions of her "MAKE" relation, which differed by their directionality: "MAKE1 was product-producer", while MAKE2 was "producer-product". Some examples of hers MAKE1 are: "honeybee", "silkworm", "musical clock", and "sebaceous glands". Examples of MAKE2: "daisy chains", "snowball", "consonantal patterns", and "molecular chains".

According to our judgment, Product-Producer (X, Y) was true for a sentence S that mentions entities X and Y, if and only if, according to common sense, the situation described in S entails the fact that X *is a product of* Y, or Y *produces* X.
An example of this relation could be found in the sentence: *"The reactor will aim to turn sea water into fuel by mimicking the way the sun produces energy"*. In this case we could infer that the sun is the producer and the *energy* is the product.

[1]R. Girju, D. Moldovan, M. Tatu, D. Antohe. 2005. *On the semantics of noun compounds.* In: Computer Speech & Language

[2]J.N. Levi, 1979. *The syntax and semantics of complex nominals.* New York: Academic Press

We then used the very same procedure already developed in the previous step of our algorithm, in order to classify couples of nominals and assign them the right semantic relation by choosing between the three mentioned. First of all we had to build a training set, selecting the appropriate seeds for every relation and then building vectors defining each one of them, using MI and LMI as a weight measure for the patterns composing every vector.
Then, we built the training set, with another list of words to be classified between the three possible relations using our modified version of Vector Space Model.

Once this task was completed, we developed an unsupervised version of this step of the algorithm. That is, we implemented a script that made it possible for the system to automatically select the seeds to be used and use them for building the training set. Also, the system could automatically select the words to be used for testing the algorithm.
We obtained pretty good results, as we will further explain.

6.1 First Approach: Manually Selected Words

We implemented two different approaches for testing our relation classifier. The first one was similar to the one we described in the previous chapter, since it used human-selected seeds for harvesting the corpus and for training the algorithm.

6.1.1 Preparing the training set

The first step was preparing a training set that could "teach" the system how the three examined relations should look like.
First of all, we had to build three different list of seeds, each one representative of one relation.
These lists were human-selected and, in order to build a functional classifier, we had to account for words that were pretty frequent in our corpus, and obviously, in order to have functional lists, we had to select nominal pairs that suited the definition we gave for every relation. Similar to the other experiments, we used only common nouns as seeds, since we were interested in classifying relations between nominals.
We selected three lists of 35 seed pair per relation, adding to total of 105 seed pair.
As already explained the seeds we selected were all common nouns. We tried selecting words that were pretty common and not domain-related, since this way, we should find more occurrences of the words in the corpus, and consequentially be able to extract more patterns from it.

For the Instrument-Agency relation we came up with about 35 trades and the relative instruments typically used for each. So we have seeds like *"dentista":"trapano"* (which means "dentist":"drill") or *"bibliotecario":"catalogo"* (Italian for "librarian ":"catalogue"). The "drill" is a typical instrument used by a "dentist", just like a "catalogue" is a typical instrument used by "librarians". All the other seeds we chose were pretty much selected using the same criterion, and were similar to these two.

For the Cause-Effect relation we strictly followed the definition we gave, and selected seeds that were considered to be one the cause of the other. For example, we selected seeds like *"virus":"infezione"* (Italian for "virus":"infection") or *"terremoto":"tsunami"* ("earthquake":"tsunami") because, according to our judgement "viruses" are considered to be causing "infections" and, in the same way, "earthquake" are considered to be causing "tsunamis". All the seeds in our list were selected according to this criterion.

For the Product-Producer relation, some of the seeds selected came from names of trades and the products that are created from these trades, like for example *"sarto": "abito"* (Italian for "tailor ": "dress"), given that we assume "tailors " to make or produce "dresses ". We also selected some other seeds that did not follow this criterion, but were still connected by the product-producer relations, according to our judgment. For example, we selected seeds like: *"gallina ":"uovo "*, which means "hen":"egg ", or *"albero ":"frutto "* which means "tree ":"fruit " because we thought the hen to be the producer of eggs, and, in the same manner, the tree to be the producer of fruit.

Once the lists were gathered the mentioned lists, we extracted patterns representing each one of these relations from our corpus, using the same script developed in the previous step.We extracted patterns representing the cause-effect relation, the instrument-agency relation and the product-producer relation, applying the same procedure already used for extracting patterns representing the meronymy and the location relations.

Once all the patterns were extracted, we evaluated which were the most representative for every relations. We used the same association measures chosen for the other experiments, which were MI and LMI, computing the association degree with every pattern and the relation it was extracted for.
Since we needed to see how every pattern related to every relation, there was no need to compute the association measure between the patterns and every single seed pair as with the previous experiments. We just computed the association

between the patterns and every relations, considering all of the seeds as if they were the same, since they were all representative of the same relation.

We discovered that the most representative patterns extracted for the Instrument-Agency relation were expressions like: *"coniato da "* which means "coined by ", *"inventato da"* which means "invented by " and *"usato "* which means "used ".
The most representative patterns extracted for the Cause-Effect relation were expressions like *"arrecare da "* which means "brought by" , *"avvenire per"* which means "to happen for".
The most representative patterns extracted for the Product-Producer relation were *"modellare "*, which means "to model ", *"secernere "*, which means "to secrete ", and *"scolpito da "* which means "carved by".

We can see that the patterns were pretty accurate and described well the given relations. There were some patterns not particularly accurate in describing the relations (in addition to the prepositions and the conjunctions, that, as we already said in the previous chapters, were not relevant for any relations), like for example *"sotto forma di "*, in the form of "to secrete " which was retrieved analyzing the Cause-Effect relation. But this was something that we anticipated even before retrieving the results, because if the words composing the pattern occurred frequently in the same sentence as the analyzed seeds, the pattern appeared as highly ranked.

Once we had all our patterns and all the weights assigned to each of the patterns based on the relations they were found with, we could finally build our vectors. We had to build three vectors, each representative of one relation. We built a vector representative of the Cause-Effect relation (for which every pattern found with that relation, had the value of MI or LMI greater than 0, and for every pattern that was not found in association with that relation had a 0 value), one representative of the Instrument-Agency relation, and one for the Product-Producer relation.

When we finished building the vectors, our training set was complete. We finally had the referring vectors for every relations, to be compared with vectors representing word pair we wanted to classify.

6.1.2 Preparing the test set

To prepare the test set, we used a procedure similar to the one just described. First of all we selected a list of words, to be classified using the three relations. We selected three lists of 15 couple of nominals, one for each relation (i.e., 45 total couple of nominals to discriminate between).
The words were selected according to the previously given definitions, but we also placed inside our lists some "ambiguous " words that could be misclassified, similar to what we did when we tried to classify words between meronymy and location relation. Since ambiguity is one peculiarity of semantic classification, it was not difficult to find ambiguous words.
For example, we classified the seed pair *"uragano":"danno"* (which means "hurricane":"damage") as members of the Cause-Effect class, because it is known that hurricanes cause damages, but still, hurricanes could also produce damages.
We did this because we were interested in evaluating how good our system could be when faced with challenging problems, by giving it ambiguous entries to analyze.
We then extracted all the patterns and evaluated them. For the training set, we did not have to evaluate the degree of association between every pattern and the seed pair they were extracted with, because we just needed to know how every pattern would describe the relation it was extracted with, and consequentially, the weight that it should have in the vector representing the relation. Instead in this case, we have to build two vectors (one with MI weighting and one with LMI) representing each seed pair. Basically, for every selected seed pair, we would have two vectors representing it, each reporting the MI or LMI value obtained for a pattern relative to that seed pair, if the pattern was found to occur with that

couple, reporting a zero otherwise. So we would have a total of 75 vectors made of MI values and a total of 75 vector made of LMI values.

Each one of these vectors was then compared with the referring vectors built previously. Vectors built using MI as their weight value, were compared with referring vectors built using the same association measure, and the same was done for vectors built using LMI as the chosen measure).

The script we implemented in order to achieve the classification of nominals was slightly modified, since in the previous experiment the script was written to select to select between two possible referring vectors, while this time it had to choose between three vectors. We modified the referring vectors, replacing the old ones with the ones gathered in the first step of this experiment. Once completed, we could easily confront them with every vector describing the word pair we needed to classify.

These scripts measured the cosine of the angle between each referring vector and the vector representing the couple that we wanted to classify. It then printed three values, one for each cosine value. The higher value, or better, the value that was closer to 1, was the one that represented the right association. In fact, if we found that the vector representing a certain seed pair and the Cause-Effect vector reported a cosine of 0,8 of the angle between them, while the same seeds' vector had a cosine of 0,3 for the angle between it and the Product-Producer relation, that word couple would be classified as representative of the Cause-Effect relation, since this has the closest value.

We ran our script for every word couple, to obtain the values of association each couple reported with every relation, according to our modified VSM.
This step was repeated twice, since we had to analyze both the vectors that implemented MI as the chosen weighting measure, and the vectors that implemented LMI.

Once all the cosines values were gathered, we were able to evaluate the results.

6.1.3 Evaluating the results

Since this experiment used a semi-unsupervised approach, all our words had already been classified by a human expert, who had selected chose every word couple as representative of an already determined category before running the classification script. So, when we obtained the results of the classified word pair, we were able to compare it to our previsions, to see how accurate the algorithm was.

As mentioned earlier, we placed some “ambiguous ” words between the couple of nominals our vector had to classify, just to see how our classifier would operate in such conditions.
In addition, some of the patterns found in the vectors built to represent the relations, were not really representative of the relations according to human judgment, but they were included because they probably were occurring frequently in the corpus with the seeds used for training the classifier. This could be a problem, since it was not obvious that the same pattern would also occur frequently with the words we wanted to classify (even though according to human judgment they were representative of the same relation). As such there could be some mis-classification due to these two factors.

Yet despite the above, our algorithm performed quite well.

First of all, we noticed that we did not found all the couples we chose in our corpus. Some of the couples did not occur, so we could not classify them. Although there were not many results, the ones obtained were quite good.

Using MI as our association measure, we reported the following results:

			Percentage
Correct Answer	11	20	55
Wrong Answer	9	20	45

We could see that we got a percentage of 55% correct answers and 45% of incorrect answers.

But this data are not completely explicative, we should take a look at precision, recall and f-measure data to analyze which kind of relation were the harder to classify.

Agency/Instrument

		Correct Answer	
		P	**N**
Gathered Result	**P**	3	0
	N	3	14

Precision	1
Recall	0,5
F-Measure	0,666666667

Cause/Effect

		Correct Answer	
		P	**N**
Gathered Result	**P**	4	2
	N	2	12

Precision	0,6666666667
Recall	0,6666666667
F-Measure	0,6666666667

Product/Producer

		Correct Answer	
		P	**N**
Gathered Result	**P**	4	4
	N	4	8

Precision	0,25
Recall	0,5
F-Measure	0,3333333333

These three table indicate the degree of precision obtained in classifying the couples of nominals between the three relations.

It can be seen that the relation that has been mostly mis-classified was the product-producer relation. The mis-classification occurred both because some seed pair that were humanly classified as members of product-producer class, were not classified that way by the script, and because some of the words humanly classified as

not being members of this class, were instead classified as such.
For example, the seed pair *"zanzara":"malaria"* (Italian for "mosquito": "malaria") was humanly classified as being member of the cause-effect relation, but the algorithm classified it as product-producer. This is not completely wrong, since it is true that mosquitoes cause malaria, but from another perspective, they even produce it.
As previously explained, the mis-classification could depend on the ambiguity of the relation, but it could also depend on the patterns that happened to be found frequently with the seeds.

The vectors built using the LMI as weighting measure provided more accurate classifications, as can be seen from the following table:

			Percentage
Correct Answer	16	20	80
Wrong Answer	4	20	20

Using LMI as the weighting measure, we improved the precision of our classifier from 55% of right answers to 80% of right answers. This could mean that we had a lot of sparse data in our corpus and between the ones we chose to analyze, since it is known that LMI performs much better in case of sparse data.

Analyzing the performance of the script with every relation, we could see that there are other differences with how the algorithm performed having LMI as a weighting measure:

Agency/Instrument

		Correct Answer	
		P	**N**
Gathered Result	**P**	3	0
	N	3	14

Precision	1
Recall	0,5
F-Measure	0,666666667

Cause/Effect

		Correct Answer	
		P	**N**
Gathered Result	**P**	6	1
	N	1	12

Precision	0,8571428571
Recall	0,8571428571
F-Measure	0,8571428571

Product/Producer

		Correct Answer	
		P	**N**
Gathered Result	**P**	7	3
	N	0	10

Precision	0,7
Recall	1
F-Measure	0,8235294118

In this case the script performed better with Cause-Effect relation and Product-Producer (which instead was the one MI had the worst results with), and reported some errors in classifying the Instrument-Agency relation. Apart from that, the results were rather good.

The misclassified couples were:

- *"musicista":"strumento"* (Italian for "musician": "instrument") which was human classified as an Instrument-Agency relation, while the script classified it as Product-Producer relation;
- *"programmatore":"computer"* (Italian for "programmer": "computer") which was human classified as Instrument-Agency, while the script classified it as Cause-Effect;
- *"zanzara":"malaria"* (Italian for "mosquito": "malaria") which was human classified as Cause-Effect, while the script classified it as Product-Producer;
- *"amanuense":"manoscritto"* (Italian for "programmer": "manuscript") which was human classified as Product-Producer, but was actually not classified by our script, since it gave all 0 results, for every relation it was associated with. When two vectors have a cosine 0 value of the angle between them, it means that the vectors are completely independent the one from the other. So, probably, in this case, our script was incapable in choosing a relation

between the proposed three, because according to the built vectors, there were no connections between any of them.

The other misclassified values could be due to the ambiguity of the relation, or maybe to an incorrectly built vector. We discussed this earlier in relation to the couple “mosquito”: “malaria”, because it happened to be misclassified even when using MI values.
Regarding the couple “musician”: “instrument”, the misclassification was probably due to the ambiguity of the words. It is true that musicians use a instrument to do their job, and so it actually is a Instrument-Agency relation as it was classified from the human judge. But a instrument could also be produced by a musician, since it is know that musicians like to customize their instruments. Perhaps, there were some topics in the corpus describing this particular fact, i.e. resulting in a vector describing these words, composed by patterns related to the Product-Producer relation, instead that the Instrument-Agency.
There also was the misclassified couple made of the nominals “programmer”:“computer”. This couple was classified by the human judge as Instrument-Agency, because programmers use computers as the main instrument in their jobs. The script instead classified it as Cause-Effect relation, maybe because there was some ambiguities or some errors in the way we built the vector representing that couple.
The other couples were correctly classified, meaning that the algorithm could, in most cases, resolve the ambiguities and that our vectors were mostly well built using patterns representative of every relations according to human judgement, reflecting how the human judge perceived each relation.

Once we finished classifying these couples, we tried to develop another version of this step of the algorithm which did not used human selected seeds, but instead implemented a script that extracted automatically the seeds both for the training and the test sets.

6.2 Second Approach: Automatically Extracted seeds

Here we were interested in seeing what would happen if we implemented a script that would automatically extract the nominals to be used for training the algorithm, and even selecting the words to evaluate.

We were also interested in seeing if there would be any improvement in the classification performance.

6.2.1 Automatically Extracting the seeds

We built a Perl script that could automatically extract the seeds.
To do this, we noted the definitions given for every relations, made a Perl script that that took as input our corpus and produced a list of seeds.
Then, we implemented a loop that searched for occurrence, in the corpus for the most prototypical patterns for each relation according to the judgment of the developer of the Perl code. The judgement of the developer was heavy influenced by the definitions we gave for every relation. That means that, in order to extract seeds for the Cause-Effect relations, we would have searched for nominals occurring with patterns like "causes" or "caused by" in between them, while in order to extract seeds representing the Product-Producer relations, we would have searched for nominals occurring with patterns like "produces" or "produced by". We operated the very same way while extracting the Instrument-Agency relation, only this time we used prototypical patterns like "uses" or "used by".
Basically we put a Regular Expression in our loop that searched for every word couple occurring with patterns like the ones mentioned above, and extracted them, putting the results in a text file, one for each relation, as can be seen in the pseudocode below:

```
FUNCTION seedSearch (Corpus) {

Takes the Corpus as Input and searches for patterns expressing the
semantic relation, returning the nominals that patterns were extracted
with as output

FOR EACH text IN Corpus DO
         Search sentences containing prototypical pattern for each relation.
         Extract the sentences they were contained in
         Isolate the two nominals around the pattern and print them in a output
         file

END}
```

Since this was only a trial, we decided to select the word couples that had been found more frequently, and put them as the seeds that our script should use to extract patterns and, finally, build the vectors describing each one relation.
We ran the script for each different relation, changing the regular expression three times to suit the relation we were working each time. The results were three lists of 34 words each.
Among all the results obtained, there were some strange and curious couples that a human judge would probably never have selected, but these word couples were pretty good expressing the relation they were extracted for.

6.2.2 Preparing the Training Set

Once the couples of nominals to be used as seeds were gathered, we were ready to use our script to extract patterns related to every seed.
But first, we took a look at the automatically extracted seeds. As mentioned, there were some "atypical" words in the lists, or at least words that a human judge would never have selected as representative of these relations, even though they were not wrong according to the classification of the relation we have given the script.
For the Agency-Instrument relation, the relations looked like the following:

- *"dato":"industria"* (Italian for "data": "industry"). This should look a pretty strange relation but it make sense if we consider that every industry (the agency in this couple) actually uses data (the instrument) for almost anything.

- *"padre":"educazione"* (Italian for "father": "education"). Actually, if we consider being a father as a job (and it is, most of the times) we could say that education is an instrument every father should use with his children, and that is why our script extracted this couple with this relation.

- *"sistema":"gentoo"* (Italian for "system": "gentoo") and *"applicazione":"Ajax"* (Italian for "application": "Ajax"). Here systems and applications are considered the agency using gentoo (a Linux distribution) and Ajax (a group of interrelated web development techniques used to create interactive web applications or rich Internet applications) to perform their jobs. It is actually true, since a lot of systems implement and use Gentoo, while a lot of web applications are built using Ajax. Also, this couple met the criterion what we decided when we were selecting the words to be used as seeds, since it selected two couples with proper nouns in them, while we chose not to account them.

Actually, all the extracted seeds were a little "atypical", and the mentioned couple is just an example, but none of them was misplaced, since they all expressed, to some degree, the relation they were bound to represent.

Even for the Cause-Effect relation we obtained some really strange couples.

- *"sifilide":"demenza"* (Italian for "syphilis": "dementia"). This seemed really strange at first, but after a quick research, we discover that syphilis could actually cause dementia, since mental illness caused by late-stage syphilis was once one of the more common forms of dementia, and spirochete could cause a lot of neurological symptoms, including changing in personality (often classified as some sort of dementia)

- *"pelle":"dermatite"* (Italian for "skin": "dermatitis"). This couple was pretty funny, because it means that having a skin could cause dermatitis. This is obvious, since it is not possible to catch a dermatitis without having a skin. But still, having a skin is not the cause of dermatitis in itself.
- *"folk":"rock"*. It is known that rock music is derived form folk music, so maybe that is why our algorithm classified folk as the cause of rock.

The above were the most "strange" and "uncommon" seed pair that the algorithm extracted for the Cause–Effect relation.

Even for the Product-Producer relation there were atypical seeds, some of which even went against some rules we stated while we were manually selecting the seeds. For example, we assumed that we would not want to have proper nouns as seeds. But the script extracted some couples in which one of the seeds was actually a proper noun, as in the following:

- *"autovettura ":"Opel"* (Italian for "car": "Opel") where Opel is a German world renown car industry.
- *"fucile":"Benelli"* (Italian for "rifle": "Benelli") where Benelli is an Italian industry that produces weapons, so it also produces rifles.
- *"veicolo":"Piaggio"* (Italian for "vehicle": "Piaggio"), Piaggio being the most important moto-scooter italian industry.
- *"sceneggiato":"Rai"* (Italian for "television serial": "Rai"). Since Rai is the Italian public television, it is obvious that it would sometimes produce some television serials.

The other couples extracted not include any proper nouns.

- *"artigiano ":"arte"* (Italian for "craftsman": "arts") which is true because a craftsman actually produces arts.

- *"treno ":"rumore"* (Italian for "train": "noise"). Even if trains do not produce noise in the literal sense, they are really noisy, so in a sense they do produce noise.
- *"autore ":"personaggio"* (Italian for "author": "character"). A novel author, invent various characters, so in a certain sense, authors produce them, while writing their book.

These couples, even if uncommon, were probably found by our script because they happened to occur frequently with the prototypical patterns we selected.
We did not just put only one pattern for every relation, but added more, in order to find more couples and fewer duplicates.
Once we had our three lists, we were ready to proceed by feeding them to the script meant to extract new patterns, and after gathering these, we could finally build the vectors meant to be representative of each relation.

To extract the patterns, we used the same script already used for every pattern extraction but we gave it as input the automatically extracted seeds.
We were able to extract a conspicuous number of patterns for every relation, and then use them, as long as their association measures with the relation, to build the vectors.
In this step of the script we built vectors representative of every relations: we computed the association measure between each pattern and the relation itself, to see how every pattern was representative of the relation and how much they should weigh in the vector.
We then built our "reference" vectors, putting the previously obtained weight measure inside them, for every pattern that was actually found to occur with every relation, and a 0 value for every pattern that was not found to occur with the relation we were building the vector for.
We built vectors using both the MI and the LMI as weighting measure. Our training set was ready; the next step was to prepare the Test set,in order to verify how well this automatic version would actually perform.

6.2.3 Preparing the Test Set

In order to develop our Test Set we had to select new seed pair. We decided to use the same script we used before and automatically extracted the seed pair to be used for the test.
We decided to test our algorithm with a limited number of seeds, so we composed three lists of 15 seed pair each, obtaining a total of 45 word couple to examine.
As with the Training Set, we extracted some "strange" and "curios" words for every relation, words that actually shared somewhat the relation they were extracted with, but that would never have been selected by a human judge.

For the Agency-Instrument relation, we found some strange couples:

- *"spada ":"scozzese"* (Italian for "sword": "Scottish"). This probably would be found because Scottish people and swords appear frequently in the same sentence, along with some patterns like "use " or "used by ". Sentences like these are probably referring to the ancient Scottish population.
- *"intercettazione ":"caso"* (Italian for "telephone tapping": "case"), where case is used like a agency while telephone tapping is considered an instrument. In this couple the relation is not the prototypical Agency-Instrument relation, since there is not a proper Agency, given that no-one is executing a proper action, but still, during cases, telephone tapping are often used as proof, as such, the Agency-Instrument relation exists for how we defined it.

The other seed pair were, more or less, prototypical words that could be expected to be found when considering the Agency-Instrument relation.

For the Cause-Effect relation we probably found the more "curious" and somewhat interesting couples.

- *"clamore ":"morte"* (Italian for "clamour ": "death"). Even if this is not a prototypical Cause-Effect relation, it is true that death, especially when it occurs to public persona, can cause clamour.

- *“moglie ”:“ansia”* (Italian for “wife ”: “anxiety”). Most husbands would probably agree with this, but, according to our script, wives cause lot of anxiety. This is probably the funnier and most curious couple extracted by our script.

While extracting couples for the Product-Producer relation, we did not find atypical results. This time, we did not even find any proper nouns included in our words, unlike the previous experience with the Training set.

Once all the needed words were gathered, we extracted patterns relative to each couple, and built the vectors meant to be representative of every word couple.
To do this, we used the same criterion implemented and tested in the other experiments.

6.2.4 Evaluating the results

The results obtained were interesting: using automatically extracted patterns improved significantly the performance of our script. Using MI as our weighting measure we had the following results:

			Percentage
Correct Answer	24	26	92,307692307692
Wrong Answer	2	26	7,6923076923077

As can be seen from the table below, we obtained a good 92.3% of correct answers, versus a 7.7% of misclassification. This was a very good performance, especially considering that using human selected seeds led to a percentage of just 55% correct anwers.
As before, we looked at precision, recall and f-measure, in order to understand which kind of relation our script had more difficulties in classifying.

We can see here that the three relations have been easily classified into the right category, although the one that was a little less precise was the Agency-Instrument

Agency/Instrument

		Correct Answer	
		P	N
Gathered Result	P	7	0
	N	1	18

Precision	1
Recall	0,875
F-Measure	0,933333333

Cause/Effect

		Correct Answer	
		P	N
Gathered Result	P	10	2
	N	0	14

Precision	0,8333333333
Recall	1
F-Measure	0,9090909091

Product/Producer

		Correct Answer	
		P	N
Gathered Result	P	7	0
	N	1	18

Precision	1
Recall	0,875
F-Measure	0,9333333333

relation.

In fact, our script classified as Cause-Effect the couple *"velivolo ":"artiglieria"* (Italian for "aircraft": "artillery") which actually was extracted as a couple related to the Agency-Instrument relation.

This misclassification was probably due to the way we built our vectors.

Something similar happened with the couple *"guerra ":"debito"* (Italian for "war": "debt")which was extracted as representative of the Producer-Product relation, and was here classified from our script as a Cause-Effect relation. Actually, in this case, we can not say that the script got it wrong, since, any Italian native speaker would probably classify this couple as being representative of the Cause-Effect relation, reasoning that a war causes debt. Still, we could infer that a war produces debt, so our extracting script was not incorrect.

This error is probably a good example of the ambiguity typical of semantic relations, since this couple could have been labeled as Cause-Effect and Producer-Product and both the classifications would have been correct in some way.

Using LMI as our weighting measure, we obtained the following results:

			Percentage
Correct Answer	23	26	88,461538461539
Wrong Answer	3	26	11,538461538462

As we can see, they were still pretty good, achieving a percentage of right classification of 88.5% versus a wrong classification percentage of 11.5%. Even though these values were slightly lower than the values obtained using MI as weighting measure, we obtained slightly better performance that the one obtained while using manually selected seeds, which obtained a 80% right classification and a 20% misclassifications.

Examining the precision, recall and f-measure we got the following results:

Agency/Instrument

		Correct Answer	
		P	N
Gathered Result	P	6	0
	N	2	18

Precision	1
Recall	0,75
F-Measure	0,857142857

Cause/Effect

		Correct Answer	
		P	N
Gathered Result	P	10	3
	N	0	13

Precision	0,7692307692
Recall	1
F-Measure	0,8695652174

Product/Producer

		Correct Answer	
		P	N
Gathered Result	P	7	0
	N	1	18

Precision	1
Recall	0,875
F-Measure	0,9333333333

As with MI, we found that the Agency-Instrument relation was the worst classified, even though all achieved relatively good results, was the Agency-Instrument

relation.
In fact, our script misclassified the couple *"merce ":"ferrovia "* (Italian for "goods": "railway"), since it classified it as Cause-Effect relation. To tell the truth, the couple is not even a proper Agency-Instrument relation, because there is not agency, even if the railway could be considered an instrument. Anyway, the script failed in recognizing the category this seed was extracted for.
The same happened for the seed pair: *"velivolo ":"artiglieria"* (Italian for "aircraft": "artillery"). Even in this case, the couple was extracted as representative of the Product-Producer relation, while it was classified as Cause-Effect.
Since this couple was misclassified even when we used MI as our weighting measure, the reason we gave before for justifying the misclassification was correct, meaning that this particular misclassification is probably due to the way we built our vectors.
Another error occurred with the couple *"guerra ":"debito"* (Italian for "war": "debt") which was extracted as representative of the Producer-Product relation, and was here classified from our script as a Cause-Effect relation. In other words, this should not even be considered an error.
Like the previous one, this couple was misclassified even using MI as a weighting measure.The error occurred because, as previously discussed, this couple could be classified both as Cause-Effect and Product-Producer.
So, basically, this error is due to the ambiguity of semantical relations, which is one of the biggest problem facing relation classification.

Chapter 7

Conclusions

Our experiments focused on two important tasks of natural language processing: relation extraction and relation classification.

We executed all our experiments using a dump of the Italian Wikipedia, extracted in November 2008.

Our first experiment was about relation extraction. We decided to try extracting *meronymy* relation and *location* relation, using a pattern based approach and manually selected seeds for building the extracting algorithm.
We searched for every occurrences of seed pairs pre classified with a given relation, in the corpus and extracted everything in between them that matched some specified criterion we decided. Everything we extracted became a pattern, meant to be representative of the relation it was extracted for. We then had to classify the obtained patterns, based on their relevance according to the relation they were meant to describe. To do this, we used two association measures: mutual information (MI) and local mutual information (LMI). This task lead to some interesting conclusions.
We were assuming that patterns made of just preposition or conjunction would have be frequently found while other, more complex pattern, would have been less frequently found. We would also assume that the more frequently found patterns would have been the less representative of the relation they were occurring with,

while the less frequently found patterns would have been more representative. This was easily inferable, since preposition and conjunction occur with pretty much every part of speech in every possible context, and so are not very indicative of anything. On the other hand, our calculations showed that the pattern that were less frequently found, were really indicative of the relation they occurred with. As a matter of fact, they obtained high values of mutual information and local mutual information with the relation itself, meaning that they were higly associated and higly representative of the relations.

The patterns extracted for both relations showed interesting things. Firstly, we should say that the higly ranked patterns according to the association measures were short sentences, verbs and adjectives.

In both cases (*meronymy* and *location* relation) we have some data occurring really frequently, and some data occurring sparsely, just once or twice. After a manual inspection of all the results we got, we discovered that mutual information worked better in weighting up patterns for the *meronymy* relation, since there were some conjunctions extracted as representative of the *meronymy* relation, that reported mildly high values of local mutual information. This may be brought back to the fact that, for the *location* relation, there were just a few sparse data, and mutual information works better for non-sparse data.
The two patterns that better described the *meronymy* relation, according to our calculus were “distinguere tra” (Italian for “distinguish between”) and “mancante di” (Italian for “missing”). In both cases, we found the patterns coherent with the definition of *meronymy* we gave. In fact, “distinguere tra” was mostly used in our corpus, in distinguishing different parts composing an object, like, for example, in this sentence: “La linguistica distingue tra fonetica e fonologia” which means “Linguistics distinguish between phonetics and phonology”. From this sentence an Italian speaker is able to infer that:

Phonetics and phonology are two branches of Linguistics which somehow are composing Linguistics.

and hence, that phonetics and linguistics are connected by *meronymy* relation.
The pattern "mancante" was typically found in sentences like: "La ruota mancante della bicicletta, which means "The *missing* wheel of the bike". From a sentence like this one, an Italian speaker would easily infer that:

A bike could have a wheel (since a wheel could be missing from a bike)

and hence the wheel is part-of the bike. We could say that our algorithm provided good results in extracting patterns for the *meronymy* relation.

Contrary to what assumed for the *meronymy* relation, patterns expressing *location* relation achieved a better classification while using local mutual information as the association measure. This may be due to the fact that there we selected less seeds to be used for the *location* relation, and between them, there were more seeds occurring just once or twice with our patterns. Still, we were able to extract a higher number of patterns for the *location* relation, meaning that the seeds we selected for this task were probably better than the ones selected for extracting the *meronymy* relation. We have found a confirmation of the fact that selecting good and appropriate seeds is necessary to have good results, when using seeds-based approach like we did.

Between the higher classied patterns, we have found some interesting results. For example, the one that scored the second best results, was "ad oriente di" (Italian for "east of"), which is clearly expressing a *location* relation since being east to something means being actually being *collocated* east to something.
We could find also "localizzato su"(Italian for "localized on") or "avere luogo in"(Italian for "take place in"). "Localizzato su" was usually found in sentences like: "Crateri sono localizzati sulla luna", which means "Craters are localized on the moon". From a sentence like this, is really easy to infer that:

craters are *collocated* on the moon

since in this case *localized* and *collocated* are synonym.
Even the pattern "avere luogo in" is pretty good at expressing the *location* relation.

Such an expression usually occurred in sentences like:“il mercato ha luogo nel tendone”, which means “market takes place in the big top”, from which a native speaker could easily deduct that:

if the markets *takes place in* the big top, it is actually *collocated in* it

So, we could assume that our way of extracting patterns using seeds was pretty good, and the scripts and regular expressions we developed for the purpose were good too, since we achieved pertinent results.
We found that one limitation of this kind approach reside in the criterion used for selecting the seeds, that is, if the selected seeds are not frequently occurring in our corpus, or they are not higly representative of the relation they are selected for, they might lead to incorrect results, or to a very meagre number of extracted pattern.
So, we found out that one way for further improving the algorithm can be selecting a new set of seeds, trying to make them even more related to the corpus than the one we used here. One approach could be manually selecting new lists of seeds. But it also possible to implement an algorithm that automatically extracts seeds representative of a certain sematic relation from the corpus.

Our second experiment was about relation classification.
We tried discriminating between *meronymy* and *location* relation, so we could re-use the data we obtained in the previous experiment.
To perform our classification, we decided to implement a variation of vector space model (VSM), in which we built vectors representing the two semantical relations at hand and one vector for each word pair selected for our test set. In fact, we used the data gathered in the previous experiment for building vectors representing the training set, while we chose 60 new seed pairs to build our test set; 30 pairs were referring to the *meronymy* relation, while the other 30 were referring to the *location* relation. All the realized vectors were n-dimensional, every dimension was a pattern found for the two relation. We built two sets of vectors, one using mutual information as weighting scheme, while the other used local mutual information, that is, for every relation and even every seed pair in our test set, we built two

vectors, one implementing mutual information values, and one implementing local mutual information, since we could not be sure which measure was going to work better and we needed to test them both.
We then measured the cosine of the angle between the vector representing the seeds and the two vectors representing the relations, in order to measure the closeness between them, to infer which relation the seeds were more similar to.
We first compared all the vectors built using MI as weighting scheme, then we compared all the vectors built using LMI.

We obtained good results with this method. For the vectors built using mutual information as weighting measure, we obtained a percentage of 72,9% right classification versus a percentage of 27,1% wrong classification.
Some of the misclassificated couples, were actually pretty ambiguous, even for a human judge. For example, the couple "canzone-album" (Italian for "song-record") was classified by the human judged as member of the *meronymy* relation, because it is easy to think of a record as *made of* songs, but still the songs could also be thought about as *collocated inside* a record. So we could say that the classification was wrong according to our classification, but it was not so wrong all considered. The same can be told about the misplaced *location* relations (the ones that were recognized as *meronymy*), for example the couple "paese-casale" (Italian for "village-farmhouse"). In fact it is true that a farmhouse is placed or collocated into a village, but is also true that a village is made of some farmhouse.
We then had confirmations of how ambiguous semantical relations are, and how this is a really big problem when trying to classify between them.

We then used precision, recall and f-measure to see which of the two relation was better classified. We found that, for the *meronymy* relation, we had a f-measure value of 0,74 while for the *location* relation we had a f-measure of 0,68. So we found out that our algorithm found less difficult classifying *meronymy* relation, at least while using mutual information as the weighting measure.

Examining vectors built using local mutual information, we got 80,5% correct answers versus 19,5% wrong ones.

From these results we can see that using LMI instead that MI improved the performance of our algorithm, since the percentage of correct results got a 7% better, given that it was almost 73% using MI and it was almost 81% using LMI. The difference between these results could be cause by the sparseness of our data since we know that Local Mutual Information tends to remove the noise that could be found with low frequencies.

If we examine precision, recall and f-measure obtained using LMI, the fact that LMI worked better for the *location* relation is even more clear.
The f-measure is the same for both the relation(in both cases we reported a value of 0,78%), meaning thatthe algorithm reported the same grade of error in evaluating both the relations, which is better than what we obtained using MI. Even the values reported are higher, meaning that this measure was more appropriate for evaluating these two relations, obviously relating to the data-set we had, and especially related to the the training set we built.

Since we obtained these results, we decided to go on with this experiment. In fact we decide to test how good our algorithm would perform in discriminating between three different semantical relations. We took inspiration for this experiment from the SemEval 2007 task 4, choosing three out of the seven relation selected by the staff of SemEval for their experiment. We selected *cause-effect* relation, *product-producer* relation and *agency-instrument* relation. While discriminating between this three relations, and using vectors built using mutual information as weighting measure, we obtained a result that was slightly worse than the one we achieved discriminating between two relations. We reported a 55% of correct classification and a 45% of uncorrect classifications. Examining the performance of our algorithm through f-measure, we discovered that the relation that was more misclassified was the production-producer one (which reported an f-measure value of 0,3 versus the 0,6 reported by cause-effect relation and agency-instrument relation). This was probably due to the ambiguity of some of the seeds chosen for the experiment. We decided not to exclude ambiguous seeds because, despite being

well aware that they may worsen the performance of our algorithm, we were interested in seeing how the algorithm itself would perform in ambiguous contexts. Probably removing them would have lead to a more correct classification, but ambiguity is one of the main characteristics of semantical relations.
Misclassification could also depend on the patterns that happened to be found frequently with the seeds, and the weight said patterns were associated with in every vector.
Using LMI as a weighting measure for building the vectors, we obtained more accurate results: we obtained a 80% correct classification, versus a 20% uncorrect classification. So, using LMI as the weighting measure, we improved the precision of our classier form 55% of right answer to 80% of right answer. This could mean that we had a lot of sparse data in our corpus and between the ones we chose to analyze.
Analyzing the results obtained computing n this case the script performed better with Cause-Effect relation and Product-Producer (which instead was the one MI had the worst results with), and reported some errors in classifying the Instrument-Agency relation. Apart from that, the results were pretty good, since we obtained a 0,75 for the agency-instrument relation, a 0,86 for the cause-effect relation and 0,82 for the product-producer relation.
Examining the misclassified couples, we found out that the misclassification were due to the ambiguity of the seeds. Some misclassified couples were the same, using mutual information and local mutual information. From this we can assume that, even if the sparseness of data could be a reason of some of the misclassified couple of words while using mutual information as a weighting measure, the biggest problem to deal with was the ambiguity of all semantics relations, that could easily lead to misclassification, since it is not really easy to classify couples as members of a determined relation to begin with.

Once we were done with this experiment, we thought about a way of improving the results we got, deciding to develop an algorithm for the automatic extraction of seed, thinking that this could be an effective way for reducing sparseness of data.

We then extracted seed pairs from our corpus trough a Perl script that searched for all occurrences of prototypical patterns for every relation, and extracted all nominal couples found to occur with them.
We extracted 34 seed pairs per relation while working on the training set, and 15 seed pairs per relation while working on the test set.
This method provided very good results, the best results we achieved in all the experiments. Still, this approach was not useful in getting rid of some incorrect classifications due to ambiguity. Using the automatically extracted seeds both for training our algorithm and for testing it, we improve dramatically its performance. In fact, while using mutual information as weighting measure, we obtain a 92,3% correct classification versus 7,7% uncorrect classification. This was a very good result, especially considering that using human selected seeds leaded to a percentage of just 55% correct answers. There were just 2 incorrect classifications, and one of them was a very good example of relation's ambiguity. The couple "guerra-debito" (Italian for "war-debt") was extracted as representative of the Producer-Product relation, and was here classied from our script as a Cause-Effect relation. Actually, in this case, we can not say that the script got it wrong, since, any Italian native speaker would probably classify this couple as being representative of the Cause-Effect relation, reasoning that a war causes debt. Still, we could infer that a war produces debt, so, neither our extracting script was wrong.
Using LMI as weighting measure, we obtained slightly worse results, since we got the 88,5% correct classification versus 11,5% wrong classification. This confirmed that automatically extracting the patterns reduced sparseness of data, as we predicted.
Evaluating f-measure, we discovered that the worst classified relation, even tough they all achieved pretty good results, was the Agency-Instrument relation, which reported a 0,86 value, versus the 0,87 achieved by cause-effect relation and a 0,93 achieved by product-producer relation.Two of the three misclassified couples, were also misclassified while using the mutual information as association measure. One of these two couple, is the already examined "guerra-debito" (Italian for "war-debt") which is probably not an error at all, since this couple could be classied

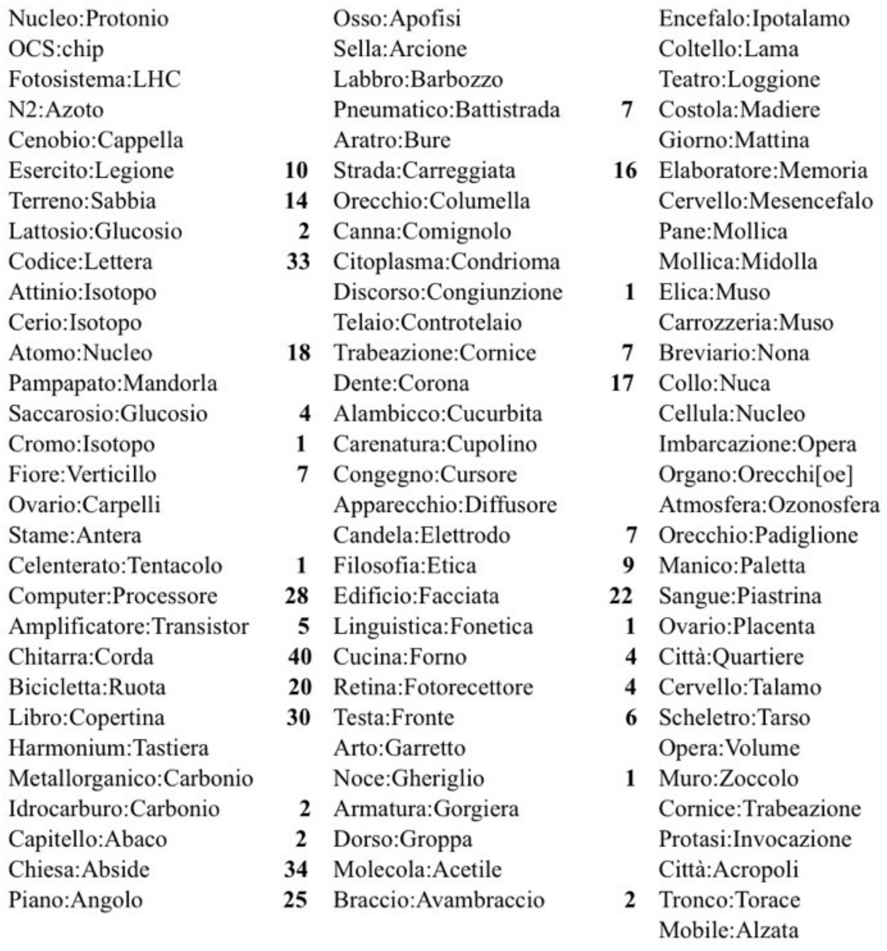
both as Cause-Effect and Product-Producer. So, basically, this error is having reference to the ambiguity of semantical relations.

We discovered that the main problem while trying to extract patterns representing semantical relations, and while trying to classify them, reside in the ambiguity of the semantical relations themselves, since this ambiguity made it really difficult to build effective method to distinguish between them.
Also, we could infer that using manually selected pattern is somewhat a limit, since it may cause data sparseness. In this case results can be improved implementing an algorithm that automatically extracts the seeds from the corpus, even though it does not completely solve the problem.
We also discovered that another problem to take care of while using our variation of vector space model to classify semantical relation, is the way we build our vector. But we can say that this is another consequence of the sparseness of data due to manually selected seeds, since seeds are used to extract the patterns, and patterns are used to build the vectors.

In the end, we can assume one of the biggest difficulty to consider is the ambiguity of semantical relations, which is part of their nature anyway, so there was not much we could do about it.

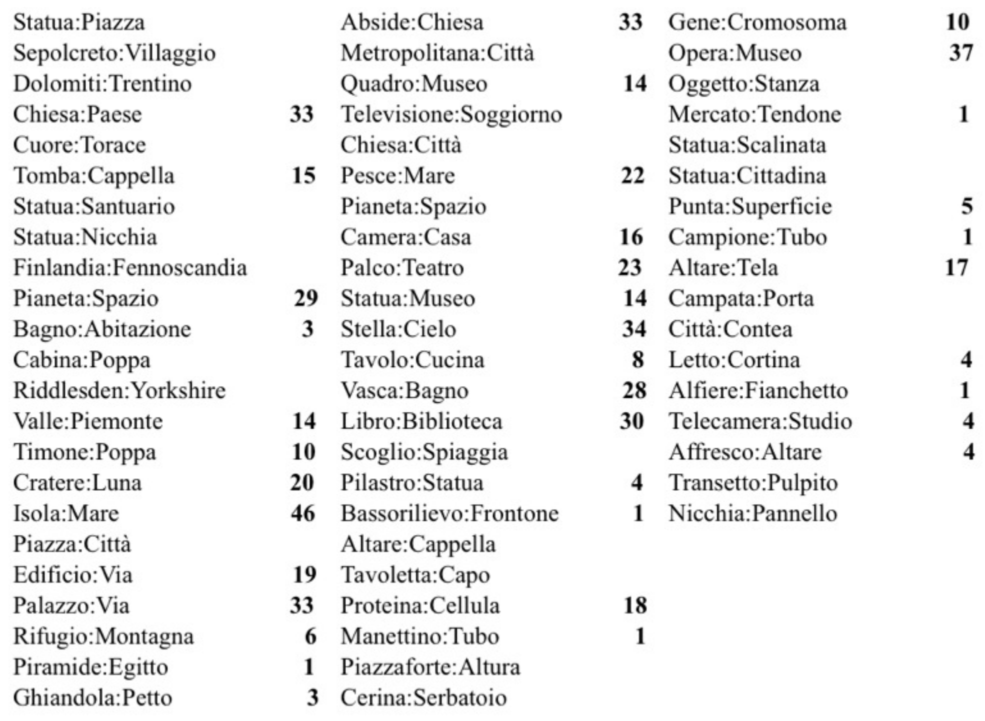
Another big problem we found, as already said, was data sparseness. We decided not to implement resources such as Google Counts to avoid this, because we were actually interested in seeing how the algorithm would perform in these conditions, even though having implemented something like that would have probably solved the problem.

Tables



Nucleo:Protonio		Osso:Apofisi		Encefalo:Ipotalamo	
OCS:chip		Sella:Arcione		Coltello:Lama	**17**
Fotosistema:LHC		Labbro:Barbozzo		Teatro:Loggione	**1**
N2:Azoto		Pneumatico:Battistrada	**7**	Costola:Madiere	
Cenobio:Cappella		Aratro:Bure		Giorno:Mattina	**19**
Esercito:Legione	**10**	Strada:Carreggiata	**16**	Elaboratore:Memoria	**2**
Terreno:Sabbia	**14**	Orecchio:Columella		Cervello:Mesencefalo	**1**
Lattosio:Glucosio	**2**	Canna:Comignolo		Pane:Mollica	**10**
Codice:Lettera	**33**	Citoplasma:Condrioma		Mollica:Midolla	
Attinio:Isotopo		Discorso:Congiunzione	**1**	Elica:Muso	**3**
Cerio:Isotopo		Telaio:Controtelaio		Carrozzeria:Muso	
Atomo:Nucleo	**18**	Trabeazione:Cornice	**7**	Breviario:Nona	
Pampapato:Mandorla		Dente:Corona	**17**	Collo:Nuca	**6**
Saccarosio:Glucosio	**4**	Alambicco:Cucurbita		Cellula:Nucleo	**23**
Cromo:Isotopo	**1**	Carenatura:Cupolino		Imbarcazione:Opera	
Fiore:Verticillo	**7**	Congegno:Cursore		Organo:Orecchi[oe]	
Ovario:Carpelli		Apparecchio:Diffusore		Atmosfera:Ozonosfera	
Stame:Antera		Candela:Elettrodo	**7**	Orecchio:Padiglione	
Celenterato:Tentacolo	**1**	Filosofia:Etica	**9**	Manico:Paletta	
Computer:Processore	**28**	Edificio:Facciata	**22**	Sangue:Piastrina	**2**
Amplificatore:Transistor	**5**	Linguistica:Fonetica	**1**	Ovario:Placenta	
Chitarra:Corda	**40**	Cucina:Forno	**4**	Città:Quartiere	
Bicicletta:Ruota	**20**	Retina:Fotorecettore	**4**	Cervello:Talamo	
Libro:Copertina	**30**	Testa:Fronte	**6**	Scheletro:Tarso	
Harmonium:Tastiera		Arto:Garretto		Opera:Volume	**44**
Metallorganico:Carbonio		Noce:Gheriglio	**1**	Muro:Zoccolo	**2**
Idrocarburo:Carbonio	**2**	Armatura:Gorgiera		Cornice:Trabeazione	**7**
Capitello:Abaco	**2**	Dorso:Groppa		Protasi:Invocazione	
Chiesa:Abside	**34**	Molecola:Acetile		Città:Acropoli	
Piano:Angolo	**25**	Braccio:Avambraccio	**2**	Tronco:Torace	
				Mobile:Alzata	

Figure 1: Seeds selected for the meronymy relation, and their frequencies in the corpus with our patterns



Statua:Piazza		Abside:Chiesa	**33**	Gene:Cromosoma	**10**
Sepolcreto:Villaggio		Metropolitana:Città		Opera:Museo	**37**
Dolomiti:Trentino		Quadro:Museo	**14**	Oggetto:Stanza	
Chiesa:Paese	**33**	Televisione:Soggiorno		Mercato:Tendone	**1**
Cuore:Torace		Chiesa:Città		Statua:Scalinata	
Tomba:Cappella	**15**	Pesce:Mare	**22**	Statua:Cittadina	
Statua:Santuario		Pianeta:Spazio		Punta:Superficie	**5**
Statua:Nicchia		Camera:Casa	**16**	Campione:Tubo	**1**
Finlandia:Fennoscandia		Palco:Teatro	**23**	Altare:Tela	**17**
Pianeta:Spazio	**29**	Statua:Museo	**14**	Campata:Porta	
Bagno:Abitazione	**3**	Stella:Cielo	**34**	Città:Contea	
Cabina:Poppa		Tavolo:Cucina	**8**	Letto:Cortina	**4**
Riddlesden:Yorkshire		Vasca:Bagno	**28**	Alfiere:Fianchetto	**1**
Valle:Piemonte	**14**	Libro:Biblioteca	**30**	Telecamera:Studio	**4**
Timone:Poppa	**10**	Scoglio:Spiaggia		Affresco:Altare	**4**
Cratere:Luna	**20**	Pilastro:Statua	**4**	Transetto:Pulpito	
Isola:Mare	**46**	Bassorilievo:Frontone	**1**	Nicchia:Pannello	
Piazza:Città		Altare:Cappella			
Edificio:Via	**19**	Tavoletta:Capo			
Palazzo:Via	**33**	Proteina:Cellula	**18**		
Rifugio:Montagna	**6**	Manettino:Tubo	**1**		
Piramide:Egitto	**1**	Piazzaforte:Altura			
Ghiandola:Petto	**3**	Cerina:Serbatoio			

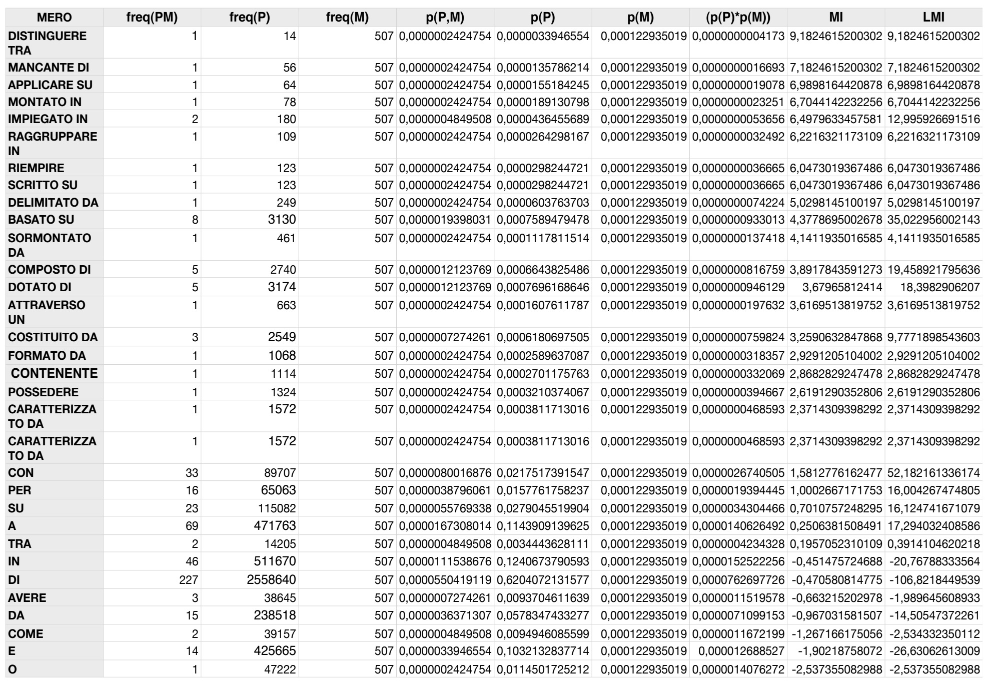
Figure 2: Seeds selected for the location relation, and their frequencies in the corpus with our patterns

Amplificatore/ Transistor	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
O	1	47222	5	0,00000024248	0,01145017252	0,00000121238	0,00000001388	4,12655875913	4,12655875913
CON	1	89707	5	0,00000024248	0,02175173915	0,00000121238	0,00000002637	3,20079733901	3,20079733901
A	2	471763	5	0,00000048495	0,11439091396	0,00000121238	0,00000013868	1,80602753619	3,61205507237
DI	1	2558640	5	0,00000024248	0,62040721316	0,00000121238	0,00000075217	-1,63321545995	-1,63321545995
Bicicletta/ Ruota	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	1	2740	20	0,00000024248	0,00066438255	0,00000484951	0,00000000322	6,23377010636	6,23377010636
DOTATO DI	1	3174	20	0,00000024248	0,00076961686	0,00000484951	0,00000000373	6,02164387137	6,02164387137
CON	3	89707	20	0,00000072743	0,02175173915	0,00000484951	0,00000010549	2,78575983973	8,35727951918
PER	1	65063	20	0,00000024248	0,01577617582	0,00000484951	0,00000007651	1,66418055929	1,66418055929
A	3	471763	20	0,00000072743	0,11439091396	0,00000484951	0,00000055474	0,39099003691	1,17297011072
DI	11	2558640	20	0,00000266723	0,62040721316	0,00000484951	0,00000300867	-0,17378384131	-1,91162225444
Cellula/Nucleo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
MANCANTE DI	1	56	23	0,00000024248	0,00001357862	0,00000557693	7,57271E-11	11,644741501	11,644741501
RAGGRUPPARE IN	1	109	23	0,00000024248	0,00002642982	0,00000557693	0,00000000015	10,6839120983	10,6839120983
COMPOSTO DI	1	2740	23	0,00000024248	0,00066438255	0,00000557693	0,00000000371	6,03213624519	6,03213624519
DOTATO DI	1	3174	23	0,00000024248	0,00076961686	0,00000557693	0,00000000429	5,8200100102	5,8200100102
CON	1	89707	23	0,00000024248	0,02175173915	0,00000557693	0,00000012131	0,99916347784	0,99916347784
IN	4	511670	23	0,0000009699	0,12406737906	0,00000557693	0,00000069192	0,4872423002	1,94896920081
DI	13	2558640	23	0,00000315218	0,62040721316	0,00000557693	0,00000345997	-0,13440960298	-1,74732483872
A	2	471763	23	0,00000048495	0,11439091396	0,00000557693	0,00000063795	-0,39560632498	-0,79121264997
Cervello/ Mesencefalo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
A	1	471763	1	0,00000024248	0,11439091396	0,00000024248	0,00000002774	3,12795563107	3,12795563107
Chiesa/Abside	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	5	89707	34	0,00000121238	0,02175173915	0,00000824416	0,00000017932	2,75719068753	13,7859534376
AVERE	1	38645	34	0,00000024248	0,00937046116	0,00000824416	0,00000007725	1,65020139205	1,65020139205
DI	26	2558640	34	0,00000630436	0,62040721316	0,00000824416	0,00000511474	0,30168951183	7,84392730754
A	2	471763	34	0,00000048495	0,11439091396	0,00000824416	0,00000094306	-0,95950721018	-1,91901442035
Chitarra/Corda	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CARATTERIZZATO DA	1	1572	40	0,00000024248	0,0003811713	0,00000969902	0,0000000037	6,03534478195	6,03534478195
A	21	471763	40	0,00000509198	0,11439091396	0,00000969902	0,00000110948	2,19834495897	46,1652441383
CON	1	89707	40	0,00000024248	0,02175173915	0,00000969902	0,00000021097	0,20079733901	0,20079733901
SU	1	115082	40	0,00000024248	0,02790455199	0,00000969902	0,00000027065	-0,15857238911	-0,15857238911
DI	13	2558640	40	0,00000315218	0,62040721316	0,00000969902	0,00000601734	-0,93277574181	-12,1260846435
DA	1	238518	40	0,00000024248	0,05783474333	0,00000969902	0,00000056094	-1,210008335	-1,210008335
IN	1	511670	40	0,00000024248	0,12406737906	0,00000969902	0,00000120333	-2,31112383863	-2,31112383863
Codice/Lettera	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
APPLICATO SU	1	64	33	0,00000024248	0,00001551842	0,00000800169	0,00000000012	10,9312642597	10,9312642597
SCRITTO SU	1	123	33	0,00000024248	0,00002982447	0,00000800169	0,00000000024	9,98874975439	9,98874975439
ATTRAVERSO UN	1	663	33	0,00000024248	0,00016076118	0,00000800169	0,00000000129	7,55839919962	7,55839919962
FORMATO DA	1	1068	33	0,00000024248	0,00025896371	0,00000800169	0,00000000207	6,87056832805	6,87056832805
COME	1	39157	33	0,00000024248	0,00949460856	0,00000800169	0,00000007597	1,67428164259	1,67428164259
A	9	471763	33	0,00000218228	0,11439091396	0,00000800169	0,00000091532	1,25348651316	11,2813786184
SU	2	115082	33	0,00000048495	0,02790455199	0,00000800169	0,00000022328	1,11896158642	2,23792317283
PER	1	65063	33	0,00000024248	0,01577617582	0,00000800169	0,00000012624	0,94171453482	0,94171453482
DA	3	238518	33	0,00000072743	0,05783474333	0,00000800169	0,00000046278	0,65248814125	1,95746442375
CON	1	89707	33	0,00000024248	0,02175173915	0,00000800169	0,00000017405	0,47833131453	0,47833131453
E	2	425665	33	0,00000048495	0,10321328377	0,00000800169	0,00000082588	-0,76809468513	-1,53618937027
DI	12	2558640	33	0,0000029097	0,62040721316	0,00000800169	0,0000049643	-0,7707189837	-9,24862780439
IN	1	511670	33	0,00000024248	0,12406737906	0,00000800169	0,00000099275	-2,0335898631	-2,0335898631
Coltello/Lama	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	3	89707	17	0,00000072743	0,02175173915	0,00000412208	0,00000008966	3,02022509336	9,06067528009
DA	2	238518	17	0,00000048495	0,05783474333	0,00000412208	0,0000002384	1,02445691864	2,04891383727
A	3	471763	17	0,00000072743	0,11439091396	0,00000412208	0,00000047153	0,62545529055	1,87636587164
E	1	425665	17	0,00000024248	0,10321328377	0,00000412208	0,00000042545	-0,81116340702	-0,81116340702
DI	6	2558640	17	0,00000145485	0,62040721316	0,00000412208	0,00000255737	-0,81378770559	-4,88272623355
Dente/Corona	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SU	1	115082	17	0,00000024248	0,02790455199	0,00000412208	0,00000011502	1,07589286453	1,07589286453
A	4	471763	17	0,0000009699	0,11439091396	0,00000412208	0,00000047153	1,04049278982	4,1619711593
DI	10	2558640	17	0,00000242475	0,62040721316	0,00000412208	0,00000255737	-0,07682211143	-0,76822111425
Esercito/ Legione	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COSTITUITO DA	2	2549	10	0,00000048495	0,00061806975	0,00000242475	0,0000000015	8,33801462618	16,6760292524
DA	3	238518	10	0,00000072743	0,05783474333	0,00000242475	0,00000014024	2,37495416572	7,12486249716
E	1	425665	10	0,00000024248	0,10321328377	0,00000242475	0,00000025027	-0,04562866066	-0,04562866066
A	1	471763	10	0,00000024248	0,11439091396	0,00000242475	0,00000027737	-0,19397246381	-0,19397246381
IN	1	511670	10	0,00000024248	0,12406737906	0,00000242475	0,00000030083	-0,31112383863	-0,31112383863
DI	2	2558640	10	0,00000048495	0,62040721316	0,00000242475	0,00000150433	-1,63321545995	-3,2664309199
Fiore/Verticillo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
POSSEDERE	1	1324	5	0,00000024248	0,00032103741	0,00000121238	0,00000000039	9,2830428774	9,2830428774
IN	3	511670	7	0,00000072743	0,12406737906	0,00000169733	0,00000021058	1,78841183492	5,36523550477
A	1	471763	7	0,00000024248	0,11439091396	0,00000169733	0,00000019416	0,32060070902	0,32060070902
DI	2	2558640	7	0,00000048495	0,62040721316	0,00000169733	0,00000105303	-1,11864228712	-2,23728457424
Libro/ Copertina	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PER	5	65063	30	0,00000121238	0,01577617582	0,00000727426	0,00000011476	3,40114615346	17,0057307673
DA	2	238518	30	0,00000048495	0,05783474333	0,00000727426	0,00000042071	0,20502916428	0,41005832856
DI	18	2558640	30	0,00000436456	0,62040721316	0,00000727426	0,000004513	-0,04825295923	-0,86855326611
A	2	471763	30	0,00000048495	0,11439091396	0,00000727426	0,00000083211	-0,77893496453	-1,55786992907
IN	2	511670	30	0,00000048495	0,12406737906	0,00000727426	0,0000009025	-0,89608633935	-1,7921726787

Opera/Volume	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
RIEMPIRE	1	123	44	0,00000024248	0,00002982447	0,00001066892	0,00000000032	9,57371225511	9,57371225511
TRA	1	14305	44	0,00000024248	0,00346861035	0,00001066892	0,00000003701	2,71199488324	2,71199488324
IN	21	511670	44	0,00000509198	0,12406737906	0,00001066892	0,00000132366	1,9436900604	40,8174912684
CON	2	89707	44	0,00000048495	0,02175173915	0,00001066892	0,00000023207	1,06329381526	2,12658763051
SU	1	115082	44	0,00000024248	0,02790455199	0,00001066892	0,00000029771	-0,29607591286	-0,29607591286
DI	16	2558640	44	0,00000387961	0,62040721316	0,00001066892	0,00000661907	-0,7707189837	-12,3315037392
E	1	425665	44	0,00000024248	0,10321328377	0,00001066892	0,00000110117	-2,18313218441	-2,18313218441
A	1	471763	44	0,00000024248	0,11439091396	0,00001066892	0,00000122043	-2,33147598756	-2,33147598756
Piano/Angolo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
TRA	1	14205	25	0,00000024248	0,00344436281	0,00000606188	0,00000002088	3,53769097824	3,53769097824
AVERE	1	38645	25	0,00000024248	0,00937046116	0,00000606188	0,0000000568	2,09380804353	2,09380804353
CON	2	89707	25	0,00000048495	0,02175173915	0,00000606188	0,00000013186	1,87886924412	3,75773848824
SU	2	115082	25	0,00000048495	0,02790455199	0,00000606188	0,00000016915	1,519499516	3,038999032
IN	4	511670	25	0,0000009699	0,12406737906	0,00000606188	0,00000075208	0,36694806648	1,46779226594
DI	11	2558640	25	0,00000266723	0,62040721316	0,00000606188	0,00000376084	-0,4957119362	-5,4528312982
A	1	471763	25	0,00000024248	0,11439091396	0,00000606188	0,00000069342	-1,5159005587	-1,5159005587
Strada/ Carreggiata	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COSTITUITO DA	1	2549	16	0,00000024248	0,00061806975	0,00000387961	0,0000000024	6,65994272107	6,65994272107
A	13	471763	16	0,00000315218	0,11439091396	0,00000387961	0,00000044379	2,82839534922	36,7691395398
DI	1	2558640	16	0,00000024248	0,62040721316	0,00000387961	0,00000240694	-3,31128736506	-3,31128736506
Terreno/Sabbia	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
A	3	471763	4	0,00000072743	0,11439091396	0,0000009699	0,00000011095	2,7129181318	8,13875439539
IN	1	511670	4	0,00000024248	0,12406737906	0,0000009699	0,00000012033	1,01080425626	1,01080425626
Edificio/ Facciata	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	1	2740	22	0,00000024248	0,00066438255	0,00000533446	0,00000000354	6,09626658261	6,09626658261
DOTATO DI	1	3174	22	0,00000024248	0,00076961686	0,00000533446	0,00000000411	5,88414034762	5,88414034762
CON	4	89707	22	0,0000009699	0,02175173915	0,00000533446	0,00000011603	3,06329381526	12,253175261
AVERE	1	38645	22	0,00000024248	0,00937046116	0,00000533446	0,00000004999	2,27823261467	2,27823261467
PER	1	65063	22	0,00000024248	0,01577617582	0,00000533446	0,00000008416	1,52667703554	1,52667703554
SU	1	115082	22	0,00000024248	0,02790455199	0,00000533446	0,00000014886	0,70392408714	0,70392408714
DA	1	238518	22	0,00000024248	0,05783474333	0,00000533446	0,00000030852	-0,34751185875	-0,34751185875
DI	8	2558640	22	0,0000019398	0,62040721316	0,00000533446	0,00000330954	-0,7707189837	-6,1657518696
Computer/ Processore	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
MONTATO IN	1	78	28	0,00000024248	0,00001891308	0,00000678931	0,00000000013	10,8829012382	10,8829012382
IMPIEGATO IN	2	180	28	0,00000048495	0,00004364557	0,00000678931	0,0000000003	10,6764503607	21,3529007214
BASATO SU	8	3130	28	0,0000019398	0,00075894795	0,00000678931	0,00000000515	8,55635651521	68,4508521217
COMPOSTO DI	1	2740	28	0,00000024248	0,00066438255	0,00000678931	0,00000000451	5,74834327919	5,74834327919
DOTATO DI	1	3174	28	0,00000024248	0,00076961686	0,00000678931	0,00000000523	5,5362170442	5,5362170442
PER	7	65063	28	0,00000169733	0,01577617582	0,00000678931	0,00000010711	3,98610865418	27,9027605793
SU	8	115082	28	0,0000019398	0,02790455199	0,00000678931	0,00000018945	3,35600078372	26,8480062697
CON	4	89707	28	0,0000009699	0,02175173915	0,00000678931	0,00000014768	2,71537051183	10,8614820473
IN	1	511670	28	0,00000024248	0,12406737906	0,00000678931	0,00000084233	-1,7965506658	-1,7965506658
DI	3	2558640	28	0,00000072743	0,62040721316	0,00000678931	0,00000421214	-2,5336797864	-7,6010393592
Filosofia/Etica	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COME	1	39157	9	0,00000024248	0,00949460856	0,00000218228	0,00000002072	3,5487507605	3,5487507605
E	2	425665	9	0,00000048495	0,10321328377	0,00000218228	0,00000022524	1,10637443278	2,21274886557
IN	2	511670	9	0,00000048495	0,12406737906	0,00000218228	0,00000027075	0,84087925482	1,68175850963
DI	5	2558640	9	0,00000121238	0,62040721316	0,00000218228	0,0000013539	-0,15928427162	-0,79642135809
Candela/ Elettrodo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	1	2740	7	0,00000024248	0,00066438255	0,00000169733	0,00000000113	7,74834327919	7,74834327919
DOTATO DI	1	3174	7	0,00000024248	0,00076961686	0,00000169733	0,00000000131	7,5362170442	7,5362170442
PER	**1**	65063	7	0,00000024248	0,01577617582	0,00000169733	0,00000002678	3,17875373212	3,17875373212
DI	4	2558640	7	0,0000009699	0,62040721316	0,00000169733	0,00000105303	-0,11864228712	-0,47456914848
Trabeazione/ Cornice	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	3	89707	7	0,00000072743	0,02175173915	0,00000169733	0,00000003692	4,30033301256	12,9009990377
E	1	425665	7	0,00000024248	0,10321328377	0,00000169733	0,00000017519	0,46894451217	0,46894451217
IN	1	511670	7	0,00000024248	0,12406737906	0,00000169733	0,00000021058	0,2034493342	0,2034493342
Pneumatico/ Battistrada	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	2	89707	7	0,00000048495	0,02175173915	0,00000169733	0,00000003692	3,71537051183	7,43074102367
DA	1	238518	7	0,00000024248	0,05783474333	0,00000169733	0,00000009816	1,30456483783	1,30456483783
DI	4	2558640	7	0,0000009699	0,62040721316	0,00000169733	0,00000105303	-0,11864228712	-0,47456914848
Testa/Fronte	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SU	2	115082	6	0,00000048495	0,02790455199	0,00000145485	0,0000000406	3,57839320505	7,15678641011
CON	1	89707	6	0,00000024248	0,02175173915	0,00000145485	0,00000003165	2,93776293317	2,93776293317
DI	3	2558640	6	0,00000072743	0,62040721316	0,00000145485	0,0000009026	-0,31128736506	-0,93386209519
Idrocarburo/ Carbonio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DELIMITATO DA	1	249	2	0,00000024248	0,00006037637	0,00000048495	2,92796E-11	13,015656447	13,015656447
CONTENENTE	1	1114	2	0,00000024248	0,00027011758	0,00000048495	0,00000000013	10,8541248618	10,8541248618
DI	1	2558640	2	0,00000024248	0,62040721316	0,00000048495	0,00000030087	-0,31128736506	-0,31128736506
Discorso/ Congiunzione	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	1	238518	1	0,00000024248	0,05783474333	0,00000024248	0,00000001402	4,11191975989	4,11191975989
Giorno/Mattina	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	18	2558640	19	0,00000436456	0,62040721316	0,00000460703	0,00000285824	0,61071012294	10,9927822129
DA	1	238518	19	0,00000024248	0,05783474333	0,00000460703	0,00000026645	-0,13600775356	-0,13600775356
Pane/Mollica	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	10	2558640	10	0,00000242465	0,62040721316	0,00000242475	0,00000150433	0,68864966907	6,88649669066
Retina/ Fotorecettore	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	4	2558640	4	0,0000009699	0,62040721316	0,0000009699	0,00000060173	0,68871263494	2,75485053975
Sangue/ Piastrina	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	2	2558640	2	0,00000048495	0,62040721316	0,00000048495	0,00000030087	0,68871263494	1,37742526988
Elaboratore/ Memoria	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	2	2558640	2	0,00000048495	0,62040721316	0,00000048495	0,00000030087	0,68871263494	1,37742526988

Capitello/ Abaco	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DI	2	2558640	2	0,00000048495	0,62040721316	0,00000048495	0,00000030087	0,68871263494	1,37742526988
Teatro/ Loggione	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DI	1	2558640	1	0,00000024248	0,62040721316	0,00000024248	0,00000015043	0,68871263494	0,68871263494
Noce/Gheriglio	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DI	1	2558640	1	0,00000024248	0,62040721316	0,00000024248	0,00000015043	0,68871263494	0,68871263494
Cromo/Isotopo	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DI	1	2558640	1	0,00000024248	0,62040721316	0,00000024248	0,00000015043	0,68871263494	0,68871263494
Celenterato/ Tentacolo	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DI	1	2558640	1	0,00000024248	0,62040721316	0,00000024248	0,00000015043	0,68871263494	0,68871263494
Collo/Nuca	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
SU	1	115082	6	0,00000024248	0,02790455199	0,00000145485	0,0000000406	2,57839320505	2,57839320505
E	2	425665	6	0,00000048495	0,10321328377	0,00000145485	0,00000015016	1,6913369335	3,38267386701
DI	3	2558640	6	0,00000072743	0,62040721316	0,00000145485	0,0000009026	-0,31128736506	-0,93386209519
Saccarosio/ Glucosio	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
E	1	425665	4	0,00000024248	0,10321328377	0,0000009699	0,00000010011	1,27629943423	1,27629943423
IN	1	511670	4	0,00000024248	0,12406737906	0,0000009699	0,00000012033	1,01080425626	1,01080425626
DI	2	2558640	4	0,00000048495	0,62040721316	0,0000009699	0,00000060173	-0,31128736506	-0,62257473012
Cucina/Forno	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
E	1	425665	4	0,00000024248	0,10321328377	0,0000009699	0,00000010011	1,27629943423	1,27629943423
DI	2	2558640	4	0,00000048495	0,62040721316	0,0000009699	0,00000060173	-0,31128736506	-0,62257473012
Atomo/Nucleo	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DI	7	2558640	18	0,00000169733	0,62040721316	0,00000436456	0,0000027078	-0,67385744445	-4,71700211113
IN	1	511670	18	0,00000024248	0,12406737906	0,00000436456	0,0000005415	-1,15912074518	-1,15912074518
Elica/Muso	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
SU	2	115082	3	0,00000048495	0,02790455199	0,00000072743	0,0000000203	4,57839320505	9,15678641011
DI	1	2558640	3	0,00000024248	0,62040721316	0,00000072743	0,0000004513	-0,89624986578	-0,89624986578
Linguistica/ Fonetica	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DISTINGUE TRA	1	14	1	0,00000024248	0,00000339466	0,00000024248	8,2312E-13	18,168303457	18,168303457
Braccio/ Avambraccio	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
E	2	425665	2	0,00000048495	0,10321328377	0,00000048495	0,00000005005	3,27629943423	6,55259886845
Lattosio/ Glucosio	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
IN	2	511670	2	0,00000048495	0,12406737906	0,00000048495	0,00000006017	3,01080425626	6,02160851252
Cornice/ Trabeazione	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
SORMONTATO DA	1	461	7	0,00000024248	0,000111781151	0,00000169733	0,00000000019	10,3196805166	10,3196805166
Muro/Zoccolo	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
SU	2	115082	2	0,00000048495	0,02790455199	0,00000048495	0,00000001353	5,16335570578	10,3267114116

FIGURE 3: Meronymy Relation expressed through seeds



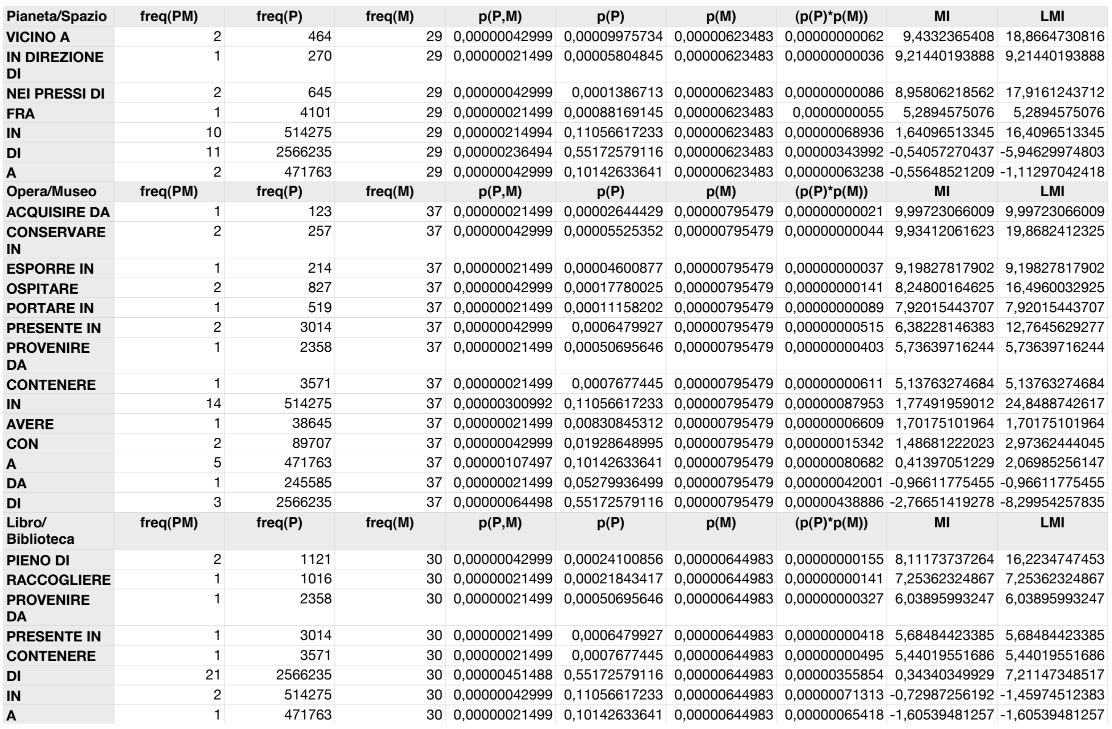
MERO	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
DISTINGUERE TRA	1	14	507	0,0000002424754	0,0000033946554	0,000122935019	0,0000000004173	9,1824615200302	9,1824615200302
MANCANTE DI	1	56	507	0,0000002424754	0,0000135786214	0,000122935019	0,0000000016693	7,1824615200302	7,1824615200302
APPLICARE SU	1	64	507	0,0000002424754	0,0000155184245	0,000122935019	0,0000000019078	6,9898164420878	6,9898164420878
MONTATO IN	1	78	507	0,0000002424754	0,0000189130798	0,000122935019	0,0000000023251	6,7044142232256	6,7044142232256
IMPIEGATO IN	2	180	507	0,0000004849508	0,0000436455689	0,000122935019	0,0000000053656	6,4979633457581	12,995926691516
RAGGRUPPARE IN	1	109	507	0,0000002424754	0,0000264298167	0,000122935019	0,0000000032492	6,2216321173109	6,2216321173109
RIEMPIRE	1	123	507	0,0000002424754	0,0000298244721	0,000122935019	0,0000000036665	6,0473019367486	6,0473019367486
SCRITTO SU	1	123	507	0,0000002424754	0,0000298244721	0,000122935019	0,0000000036665	6,0473019367486	6,0473019367486
DELIMITATO DA	1	249	507	0,0000002424754	0,0000603763703	0,000122935019	0,0000000074224	5,0298145100197	5,0298145100197
BASATO SU	8	3130	507	0,0000019398031	0,0007589479478	0,000122935019	0,0000000933013	4,3778695002678	35,022956002143
SORMONTATO DA	1	461	507	0,0000002424754	0,0001117811514	0,000122935019	0,0000000137418	4,1411935016585	4,1411935016585
COMPOSTO DI	5	2740	507	0,0000012123769	0,0006643825486	0,000122935019	0,0000000816759	3,8917843591273	19,458921795636
DOTATO DI	5	3174	507	0,0000012123769	0,0007696168646	0,000122935019	0,0000000946129	3,67965812414	18,3982906207
ATTRAVERSO UN	1	663	507	0,0000002424754	0,0001607611787	0,000122935019	0,0000000197632	3,6169513819752	3,6169513819752
COSTITUITO DA	3	2549	507	0,0000007274261	0,0006180697505	0,000122935019	0,0000000759824	3,2590632847868	9,7771898543603
FORMATO DA	1	1068	507	0,0000002424754	0,0002589637087	0,000122935019	0,0000000318357	2,9291205104002	2,9291205104002
CONTENENTE	1	1114	507	0,0000002424754	0,0002701175763	0,000122935019	0,0000000332069	2,8682829247478	2,8682829247478
POSSEDERE	1	1324	507	0,0000002424754	0,0003210374067	0,000122935019	0,0000000394667	2,6191290352806	2,6191290352806
CARATTERIZZA TO DA	1	1572	507	0,0000002424754	0,0003811713016	0,000122935019	0,0000000468593	2,3714309398292	2,3714309398292
CARATTERIZZA TO DA	1	1572	507	0,0000002424754	0,0003811713016	0,000122935019	0,0000000468593	2,3714309398292	2,3714309398292
CON	33	89707	507	0,0000080016876	0,0217517391547	0,000122935019	0,0000026740505	1,5812776162477	52,182161336174
PER	16	65063	507	0,0000038796061	0,0157761758237	0,000122935019	0,0000019394445	1,0002667171753	16,004267474805
SU	23	115082	507	0,0000055769338	0,0279045519904	0,000122935019	0,0000034304466	0,7010757248295	16,124741671079
A	69	471763	507	0,0000167308014	0,1143909139625	0,000122935019	0,0000140626492	0,2506381508491	17,294032408586
TRA	2	14205	507	0,0000004849508	0,0034443628111	0,000122935019	0,0000004234328	0,1957052310109	0,3914104620218
IN	46	511670	507	0,0000111538676	0,1240673790593	0,000122935019	0,0000152522256	-0,451475724688	-20,76788333564
DI	227	2558640	507	0,0000550419119	0,6204072131577	0,000122935019	0,0000762697726	-0,470580814775	-106,8218449539
AVERE	3	38645	507	0,0000007274261	0,0093704611639	0,000122935019	0,0000011519578	-0,663215202978	-1,989645608933
DA	15	238518	507	0,0000036371307	0,0578347433277	0,000122935019	0,0000071099153	-0,967031581507	-14,50547372261
COME	2	39157	507	0,0000004849508	0,0094946085599	0,000122935019	0,0000011672199	-1,267166175056	-2,534332350112
E	14	425665	507	0,0000033946554	0,1032132837714	0,000122935019	0,000012688527	-1,90218758072	-26,63062613009
O	1	47222	507	0,0000002424754	0,0114501725212	0,000122935019	0,0000014076272	-2,537355082988	-2,537355082988

Figure 4: Patterns expressing meronymy relation

Affresco/Altare	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	1	89707	4	0,00000021499	0,01928648995	0,00000085998	0,00000001659	3,69626558585	3,69626558585
DI	1	2566235	4	0,00000021499	0,55172579116	0,00000085998	0,00000047447	-1,14202332788	-1,14202332788
E	1	425665	4	0,00000021499	0,09151553108	0,00000085998	0,0000000787	1,44983958619	1,44983958619
Vasca/Bagno	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	1	89707	28	0,00000021499	0,01928648995	0,00000601984	0,0000001161	0,8889106638	0,8889106638
DA	25	245585	28	0,00000537486	0,05279936499	0,00000601984	0,00000031784	4,0798368788	101,99592197
IN	2	514275	28	0,00000042999	0,11056617233	0,00000601984	0,00000066559	-0,63033688837	-1,26067377673
Stella/Cielo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SORGERE	1	44	34	0,00000021499	0,00000945975	0,0000073098	6,91489E-11	11,6023040712	11,6023040712
PIENO DI	2	1121	34	0,00000042999	0,00024100856	0,0000073098	0,00000000176	7,931165127	15,862330254
NEI PRESSI DI	1	645	34	0,00000021499	0,0001386713	0,0000073098	0,00000000101	7,72858033949	7,72858033949
SENZA	1	8659	34	0,00000021499	0,00186163529	0,0000073098	0,00000001361	3,98175098277	3,98175098277
CON	2	89707	34	0,00000042999	0,01928648995	0,0000073098	0,00000014098	1,6088027446	3,21760548921
IN	8	514275	34	0,00000171995	0,11056617233	0,0000073098	0,00000080822	1,08955519244	8,71644153953
DI	18	2566235	34	0,0000038699	0,55172579116	0,0000073098	0,00000403301	-0,05956116768	-1,07210101832
E	1	425665	34	0,00000021499	0,09151553108	0,0000073098	0,00000066896	-1,63762325506	-1,63762325506
Tavolo/Cucina	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	2	89707	8	0,00000042999	0,01928648995	0,00000171995	0,00000003317	3,69626558585	7,39253117171
DI	6	2566235	8	0,00000128997	0,55172579116	0,00000171995	0,00000094894	0,44293917284	2,65763503707
Camera/Casa	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	1	89707	16	0,00000021499	0,01928648995	0,00000343991	0,00000006634	1,69626558585	1,69626558585
IN	5	514275	16	0,00000107497	0,11056617233	0,00000343991	0,00000038034	1,49894612858	7,4947306429
DI	9	2566235	16	0,00000193495	0,55172579116	0,00000343991	0,00000189789	0,02790167357	0,25111506209
E	1	425665	16	0,00000021499	0,09151553108	0,00000343991	0,00000031481	-0,55016041381	-0,55016041381
Campione/ Tubo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CONSERVARE IN	1	257	1	0,00000021499	0,00005525352	0,00000021499	1,18792E-11	14,1435739819	14,1435739819
Oggetto/Stanza	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PIENO DI	1	1121	6	0,00000021499	0,00024100856	0,00000128997	0,00000000031	9,43366546753	9,43366546753
PRESENTE	2	3014	6	0,00000042999	0,0006479927	0,00000128997	0,00000000084	9,00677232873	18,0135446575
DI	3	2566235	6	0,00000064498	0,55172579116	0,00000128997	0,00000071171	-0,14202332788	-0,42606998363
Bagno/ Abitazione	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
E	1	425665	3	0,00000021499	0,09151553108	0,00000064498	0,00000005903	1,86487708547	1,86487708547
DI	2	2566235	3	0,00000042999	0,55172579116	0,00000064498	0,00000035585	0,2730141714	0,5460283428
Ghiandola/ Petto	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SITUARE SU	1	338	3	0,00000021499	0,00007266806	0,00000064498	4,68696E-11	12,1633565941	12,1633565941
POSTO SU	1	348	3	0,00000021499	0,000074818	0,00000064498	4,82563E-11	12,1212925345	12,1212925345
DI	1	2566235	3	0,00000021499	0,55172579116	0,00000064498	0,00000035585	-0,7269858286	-0,7269858286
Rifugio/ Montagna	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SU	1	118500	6	0,00000021499	0,02547681964	0,00000128997	0,00000003286	2,70970849677	2,70970849677
IN	1	514275	6	0,00000021499	0,11056617233	0,00000128997	0,00000014263	0,59205553297	0,59205553297
DI	4	2566235	6	0,00000085998	0,55172579116	0,00000128997	0,00000071171	0,2730141714	1,09205668561
Mercato/ Tendone	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	1	2566235	1	0,00000021499	0,55172579116	0,00000021499	0,00000011862	0,85797667212	0,85797667212
Valle/Piemonte	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
ESSERE PRESENTE	1	1560	14	0,00000021499	0,00033539104	0,00000300992	0,00000000101	7,73451329525	7,73451329525
SITUARE IN	1	9804	14	0,00000021499	0,00210780371	0,00000300992	0,00000000634	5,08268884013	5,08268884013
DI	11	2566235	14	0,00000236494	0,55172579116	0,00000300992	0,00000166065	0,5100533687	5,61058705574
IN	1	514275	14	0,00000021499	0,11056617233	0,00000300992	0,0000003328	-0,63033688837	-0,63033688837
Letto/Cortina	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
VENIRE COLLOCATI	1	46	4	0,00000021499	0,00000988974	0,00000085998	8,50495E-12	14,625636575	14,625636575
DI	3	2566235	4	0,00000064498	0,55172579116	0,00000085998	0,00000047447	0,44293917284	1,32881751853
Manettino/ Tubo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
FISSARE SU	1	48	1	0,00000021499	0,00001031972	0,00000021499	2,21868E-12	16,5642360303	16,5642360303
Piramide/Egitto	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
IN	1	514275	1	0,00000021499	0,11056617233	0,00000021499	0,00000002377	3,17701803369	3,17701803369
Gene/ Cromosoma	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
LOCALIZZARE SU	4	24	10	0,00000085998	0,00000515986	0,00000214994	1,10934E-11	16,2423079354	64,9692317418
SITUARE SU	1	338	10	0,00000021499	0,00007266806	0,00000214994	0,00000000016	10,4263909999	10,4263909999
SU	4	118500	10	0,00000085998	0,02547681964	0,00000214994	0,00000005477	3,97274290261	15,8909716104
IN	1	514275	10	0,00000021499	0,11056617233	0,00000214994	0,00000023771	-0,1449100612	-0,1449100612
Alfiere/ Fianchetto	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
IN	1	514275	1	0,00000021499	0,11056617233	0,00000021499	0,00000002377	3,17701803369	3,17701803369

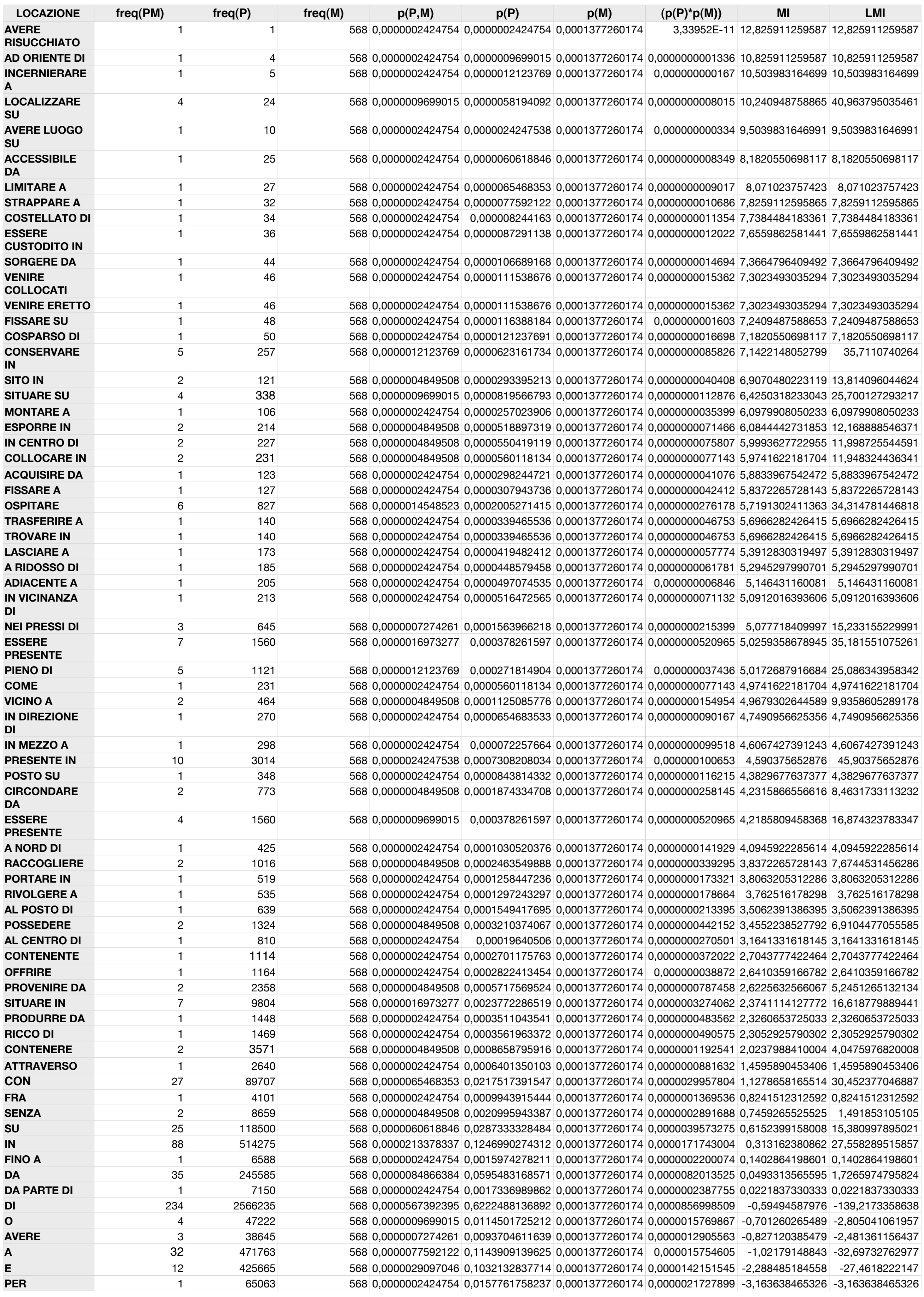
Statua/Museo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
TRASFERIRE A	1	140	14	0,00000021499	0,0000300992	0,00000300992	9,05962E-11	11,2125605921	11,2125605921
COLLOCARE IN	1	231	14	0,00000021499	0,00004966367	0,00000300992	0,00000000015	10,4900945676	10,4900945676
COME	1	231	14	0,00000021499	0,00004966367	0,00000300992	0,00000000015	10,4900945676	10,4900945676
CONSERVARE IN	1	257	14	0,00000021499	0,00005525352	0,00000300992	0,00000000017	10,3362190598	10,3362190598
OSPITARE	2	827	14	0,00000042999	0,00017780025	0,00000300992	0,00000000054	9,65010008982	19,3002001796
IN	4	514275	14	0,00000085998	0,11056617233	0,00000300992	0,0000003328	1,36966311163	5,47865244654
A	1	471763	14	0,00000021499	0,10142633641	0,00000300992	0,00000030529	-0,50585913902	-0,50585913902
DI	3	2566235	14	0,00000064498	0,55172579116	0,00000300992	0,00000166065	-1,36441574921	-4,09324724764
Edificio/Via	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SITUARE SU	1	338	19	0,00000021499	0,00007266806	0,00000408489	0,0000000003	9,50039158133	9,50039158133
ESSERE PRESENTE	3	1560	19	0,00000064498	0,00033539104	0,00000408489	0,00000000137	8,87890320458	26,6367096137
RIVOLGERE A	1	535	19	0,00000021499	0,00011502193	0,00000408489	0,00000000047	8,83787593632	8,83787593632
SITUARE IN	3	9804	19	0,00000064498	0,00210780371	0,00000408489	0,00000000861	6,22707874946	18,6812362484
IN	6	514275	19	0,00000128997	0,11056617233	0,00000408489	0,00000045165	1,51405302097	9,08431812582
SU	1	118500	19	0,00000021499	0,02547681964	0,00000408489	0,00000010407	1,04674348405	1,04674348405
A	1	471763	19	0,00000021499	0,10142633641	0,00000408489	0,00000041432	-0,94643173041	-0,94643173041
DI	3	2566235	19	0,00000064498	0,55172579116	0,00000408489	0,00000225374	-1,8049883406	-5,4149650218
Timone/Poppa	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
INCERNIERARE A	1	5	10	0,00000021499	0,00000107497	0,00000214994	2,31113E-12	16,5053423413	16,5053423413
MONTARE A	1	106	10	0,00000021499	0,00002278939	0,00000214994	4,89959E-11	12,0993499816	12,0993499816
FISSARE A	1	127	10	0,00000021499	0,00002730427	0,00000214994	5,87026E-11	11,8385857494	11,8385857494
AL POSTO DI	1	639	10	0,00000021499	0,00013738133	0,00000214994	0,0000000003	9,50759831522	9,50759831522
CON	2	89707	10	0,00000042999	0,01928648995	0,00000214994	0,00000004146	3,37433749097	6,74867498194
A	3	471763	10	0,00000064498	0,10142633641	0,00000214994	0,00000021806	1,56453018887	4,69359056661
DI	1	2566235	10	0,00000021499	0,55172579116	0,00000214994	0,00000118618	-2,46395142276	-2,46395142276
Isola/Mare	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
STRAPPARE A	1	32	46	0,00000021499	0,00000687982	0,00000988974	6,80396E-11	11,625636575	11,625636575
COSTELLATO DI	1	34	46	0,00000021499	0,0000073098	0,00000988974	7,2292E-11	11,5381737337	11,5381737337
COSPARSO DI	1	50	46	0,00000021499	0,00001074971	0,00000988974	0,00000000011	10,9817803852	10,9817803852
IN VICINANZA DI	1	213	46	0,00000021499	0,00004579378	0,00000988974	0,00000000045	8,89092695477	8,89092695477
IN MEZZO A	1	298	46	0,00000021499	0,00006406829	0,00000988974	0,00000000063	8,40646805453	8,40646805453
A NORD DI	1	425	46	0,00000021499	0,00009137256	0,00000988974	0,0000000009	7,89431754397	7,89431754397
CIRCONDARE DA	1	773	46	0,00000021499	0,00016619056	0,00000988974	0,00000000164	7,03131197107	7,03131197107
ESSERE PRESENTE	2	1560	46	0,00000042999	0,00033539104	0,00000988974	0,00000000332	7,01830626125	14,0366125225
ESSERE PRESENTE	2	1560	46	0,00000042999	0,00033539104	0,00000988974	0,00000000332	7,01830626125	14,0366125225
AL CENTRO DI	1	810	46	0,00000021499	0,00017414535	0,00000988974	0,00000000172	6,96385847722	6,96385847722
PRESENTE IN	2	3014	46	0,00000042999	0,0006479927	0,00000988974	0,00000000641	6,0681728734	12,1363457468
SITUARE IN	2	9804	46	0,00000042999	0,00210780371	0,00000988974	0,00000002085	4,36648180613	8,73296361226
FINO A	1	6588	46	0,00000021499	0,00141638218	0,00000988974	0,00000001401	3,94001173527	3,94001173527
DA PARTE DI	1	7150	46	0,00000021499	0,00153720895	0,00000988974	0,0000000152	3,82190904844	3,82190904844
SU	2	118500	46	0,00000042999	0,02547681964	0,00000988974	0,00000025196	0,77110904144	1,54221808287
A	4	471763	46	0,00000085998	0,10142633641	0,00000988974	0,00000100308	-0,22206617302	-0,88826469208
IN	4	514275	46	0,00000085998	0,11056617233	0,00000988974	0,00000109347	-0,34654392237	-1,38617568946
DI	18	2566235	46	0,0000038699	0,55172579116	0,00000988974	0,00000545642	-0,49566028249	-8,92188508484
Palco/Teatro	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
A RIDOSSO DI	1	185	23	0,00000021499	0,00003977394	0,00000494487	0,0000000002	10,0942551145	10,0942551145
A	6	471763	23	0,00000128997	0,10142633641	0,00000494487	0,00000050154	1,3628963277	8,1773779662
CON	1	89707	23	0,00000021499	0,01928648995	0,00000494487	0,0000009537	1,1727036298	1,1727036298
DI	15	2566235	23	0,00000322491	0,55172579116	0,00000494487	0,00000272821	0,24130531168	3,61957967513
Abside/Chiesa	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	5	89707	33	0,00000107497	0,01928648995	0,00000709481	0,00000013683	2,97379956138	14,8689978069
AVERE	1	38645	33	0,00000021499	0,00830845312	0,00000709481	0,00000005895	1,86681026591	1,86681026591
DI	25	2566235	33	0,00000537486	0,55172579116	0,00000709481	0,00000391439	0,45743874254	11,4359685635
A	2	471763	33	0,00000042999	0,10142633641	0,00000709481	0,0000007196	-0,74289833632	-1,48579667264
Cratere/Luna	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
AVERE LUOGO SU	1	10	20	0,00000021499	0,00000214994	0,00000429989	9,24451E-12	14,5053423413	14,5053423413
SU	13	118500	20	0,00000279493	0,02547681964	0,00000429989	0,00000010955	4,67318262075	60,7513740697
A	1	471763	20	0,00000021499	0,10142633641	0,00000429989	0,00000043612	-1,02043231185	-1,02043231185
DI	5	2566235	20	0,00000107497	0,55172579116	0,00000429989	0,00000237236	-1,14202332788	-5,71011663938
Punta/ Superficie	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	1	89707	5	0,00000021499	0,01928648995	0,00000107497	0,00000002073	3,37433749097	3,37433749097
DA	1	245585	5	0,00000021499	0,05279936499	0,00000107497	0,00000005676	1,92140751619	1,92140751619
E	1	425665	5	0,00000021499	0,09151553108	0,00000107497	0,00000009838	1,1279114913	1,1279114913
A	1	471763	5	0,00000021499	0,10142633641	0,00000107497	0,00000010903	0,97956768815	0,97956768815
DI	1	2566235	5	0,00000021499	0,55172579116	0,00000107497	0,00000059309	-1,46395142276	-1,46395142276
Chiesa/Paese	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
IN CENTRO DI	2	227	33	0,00000042999	0,0000488037	0,00000709481	0,00000000035	10,2782559244	20,5565118488
SITO IN	1	121	33	0,00000021499	0,00002601431	0,00000709481	0,00000000018	10,1859411744	10,1859411744
NEI PRESSI DI	1	645	33	0,00000021499	0,0001386713	0,00000709481	0,00000000098	7,77164906138	7,77164906138
POSSEDERE	2	1324	33	0,00000042999	0,0002846524	0,00000709481	0,00000000202	7,73411700489	15,4682340098
ESSERE PRESENTE	1	1560	33	0,00000021499	0,00033539104	0,00000709481	0,00000000238	6,49747409795	6,49747409795
PRESENTE IN	1	3014	33	0,00000021499	0,0006479927	0,00000709481	0,0000000046	5,5473407101	5,5473407101
DI	22	2566235	33	0,00000472987	0,55172579116	0,00000709481	0,00000391439	0,2730141714	6,00631177085
A	2	471763	33	0,00000042999	0,10142633641	0,00000709481	0,0000007196	-0,74289833632	-1,48579667264
IN	1	514275	33	0,00000021499	0,11056617233	0,00000709481	0,00000078445	-1,86737608567	-1,86737608567

Proteina/ Cellula	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
TROVARE IN	1	140	18	0,00000021499	0,0000300992	0,0000038699	0,00000000012	10,8499905127	10,8499905127
PRODURRE DA	1	1448	18	0,00000021499	0,00031131169	0,0000038699	0,0000000012	7,47942764253	7,47942764253
RICCO DI	1	1469	18	0,00000021499	0,00031582657	0,0000038699	0,00000000122	7,45865484905	7,45865484905
PRESENTE IN	2	3014	18	0,00000042999	0,0006479927	0,0000038699	0,00000000251	7,42180982801	14,843619656
ESSERE PRESENTE	1	1560	18	0,00000021499	0,00033539104	0,0000038699	0,0000000013	7,37194321586	7,37194321586
ATTRAVERSO	1	2640	18	0,00000021499	0,00056758484	0,0000038699	0,0000000022	6,61295131537	6,61295131537
DA	3	245585	18	0,00000064498	0,05279936499	0,0000038699	0,00000020433	1,65837311036	4,97511933108
IN	3	514275	18	0,00000064498	0,11056617233	0,0000038699	0,00000042788	0,59205553297	1,77616659891
E	1	425665	18	0,00000021499	0,09151553108	0,0000038699	0,00000035416	-0,72008541525	-0,72008541525
DI	4	2566235	18	0,00000085998	0,55172579116	0,0000038699	0,00000213512	-1,31194832932	-5,24779331727
Quadro/Museo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
ACCESSIBILE DA	1	25	14	0,00000021499	0,00000537486	0,00000300992	1,61779E-11	13,6979874192	13,6979874192
ESSERE CUSTODITO IN	1	36	14	0,00000021499	0,00000773979	0,00000300992	2,32962E-11	13,1719186076	13,1719186076
LASCIARE A	1	173	14	0,00000021499	0,00003719401	0,00000300992	0,00000000011	10,9072153814	10,9072153814
ESPORRE IN	1	214	14	0,00000021499	0,00004600877	0,00000300992	0,00000000014	10,6003766226	10,6003766226
OSPITARE	1	827	14	0,00000021499	0,00017780025	0,00000300992	0,00000000054	8,65010008982	8,65010008982
CONTENENTE	1	1114	14	0,00000021499	0,0002395036	0,00000300992	0,00000000072	8,22031009166	8,22031009166
OFFRIRE	1	1164	14	0,00000021499	0,00025025332	0,00000300992	0,00000000075	8,15696826609	8,15696826609
CON	1	89707	14	0,00000021499	0,01928648995	0,00000300992	0,00000005805	1,8889106638	1,8889106638
DA	1	245585	14	0,00000021499	0,05279936499	0,00000300992	0,00000015892	0,43598068902	0,43598068902
IN	2	514275	14	0,00000042999	0,11056617233	0,00000300992	0,0000003328	0,36966311163	0,73932622327
DI	3	2566235	14	0,00000064498	0,55172579116	0,00000300992	0,00000166065	-1,36441574921	-4,09324724764
Palazzo/Via	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
LIMITARE A	1	27	33	0,00000021499	0,00000580484	0,00000709481	4,11843E-11	12,3499169095	12,3499169095
VENIRE ERETTO	1	46	33	0,00000021499	0,00000988974	0,00000709481	7,01658E-11	11,5812424556	11,5812424556
SITO IN	1	121	33	0,00000021499	0,00002601431	0,00000709481	0,00000000018	10,1859411744	10,1859411744
ADIACENTE A	1	205	33	0,00000021499	0,00004407382	0,00000709481	0,00000000031	9,42532431219	9,42532431219
SITUARE SU	1	338	33	0,00000021499	0,00007266806	0,00000709481	0,00000000052	8,70392497541	8,70392497541
ESSERE PRESENTE	1	1560	33	0,00000021499	0,00033539104	0,00000709481	0,00000000238	6,49747409795	6,49747409795
SITUARE IN	1	9804	33	0,00000021499	0,00210780371	0,00000709481	0,00000001495	3,84564964283	3,84564964283
IN	11	514275	33	0,00000236494	0,11056617233	0,00000709481	0,00000078445	1,59205553297	17,5126108627
PER	1	65063	33	0,00000021499	0,01398817145	0,00000709481	0,00000009924	1,11525468678	1,11525468678
E	3	425665	33	0,00000064498	0,09151553108	0,00000709481	0,00000064929	-0,00959203245	-0,02877609735
DI	13	2566235	33	0,00000279493	0,55172579116	0,00000709481	0,00000391439	-0,48597772909	-6,31771047822
DA	1	245585	33	0,00000021499	0,05279936499	0,00000709481	0,0000003746	-0,80105850828	-0,80105850828
Tomba/ Cappella	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
A ORIENTE DI	1	4	15	0,00000021499	0,00000085998	0,00000322491	2,77335E-12	16,2423079354	16,2423079354
RACCOGLIERE	1	1016	15	0,00000021499	0,00021843417	0,00000322491	0,0000000007	8,25362324867	8,25362324867
O	1	47222	15	0,00000021499	0,01015245888	0,00000322491	0,00000003274	2,71513641037	2,71513641037
IN	5	514275	15	0,00000107497	0,11056617233	0,00000322491	0,00000035657	1,59205553297	7,96027766485
DA	1	245585	15	0,00000021499	0,05279936499	0,00000322491	0,00000017027	0,33644501547	0,33644501547
E	1	425665	15	0,00000021499	0,09151553108	0,00000322491	0,00000029513	-0,45705100942	-0,45705100942
DI	5	2566235	15	0,00000107497	0,55172579116	0,00000322491	0,00000177927	-0,7269858286	-3,63492914299
Altare/Tela	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CONSERVARE IN	1	257	17	0,00000021499	0,00005525352	0,0000036549	0,0000000002	10,0561111406	10,0561111406
OSPITARE	1	827	17	0,00000021499	0,00017780025	0,0000036549	0,00000000065	8,36999217063	8,36999217063
CON	5	89707	17	0,00000107497	0,01928648995	0,0000036549	0,00000007049	3,93073083949	19,6536541975
AVERE	1	38645	17	0,00000021499	0,00830845312	0,0000036549	0,00000003037	2,82374154402	2,82374154402
SU	1	118500	17	0,00000021499	0,02547681964	0,0000036549	0,00000009312	1,20720815624	1,20720815624
DA	1	245585	17	0,00000021499	0,05279936499	0,0000036549	0,00000019298	0,15587276983	0,15587276983
IN	2	514275	17	0,00000042999	0,11056617233	0,0000036549	0,00000040411	0,08955519244	0,17911038488
E	1	425665	17	0,00000021499	0,09151553108	0,0000036549	0,00000033448	-0,63762325506	-0,63762325506
DI	4	2566235	17	0,00000085998	0,55172579116	0,0000036549	0,0000020165	-1,22948616913	-4,91794467651
Pesce/Mare	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
AVERE RISUCCHIATO	1	1	22	0,00000021499	0,00000021499	0,00000472987	1,0169E-12	17,6897669124	17,6897669124
CIRCONDARE DA	1	773	22	0,00000021499	0,00016619056	0,00000472987	0,00000000079	8,09544230849	8,09544230849
SENZA	1	8659	22	0,00000021499	0,00186163529	0,00000472987	0,00000000881	4,60978220538	4,60978220538
O	3	47222	22	0,00000064498	0,01015245888	0,00000472987	0,00000004802	3,74755788806	11,2426736642
DI	13	2566235	22	0,00000279493	0,55172579116	0,00000472987	0,00000260959	0,09898477163	1,28680203116
IN	2	514275	22	0,00000042999	0,11056617233	0,00000472987	0,00000052296	-0,28241358495	-0,56482716989
E	1	425665	22	0,00000021499	0,09151553108	0,00000472987	0,00000043286	-1,00959203245	-1,00959203245
Telecamera/ Studio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COLLOCARE IN	1	231	4	0,00000021499	0,00004966367	0,00000085998	4,27096E-11	12,2974494896	12,2974494896
IN	1	514275	4	0,00000021499	0,11056617233	0,00000085998	0,00000009508	1,17701803369	1,17701803369
DI	2	2566235	4	0,00000042999	0,55172579116	0,00000085998	0,00000047447	-0,14202332788	-0,28404665575
Pilastro/Statua	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	2	89707	4	0,00000042999	0,01928648995	0,00000085998	0,00000001659	4,69626558585	9,39253117171
DA	1	245585	4	0,00000021499	0,05279936499	0,00000085998	0,00000004541	2,24333561108	2,24333561108
SU	1	118500	4	0,00000021499	0,02547681964	0,00000085998	0,00000002191	3,29467099749	3,29467099749
Bassorilievo/ Frontone	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	1	89707	1	0,00000021499	0,01928648995	0,00000021499	0,00000000415	5,69626558585	5,69626558585



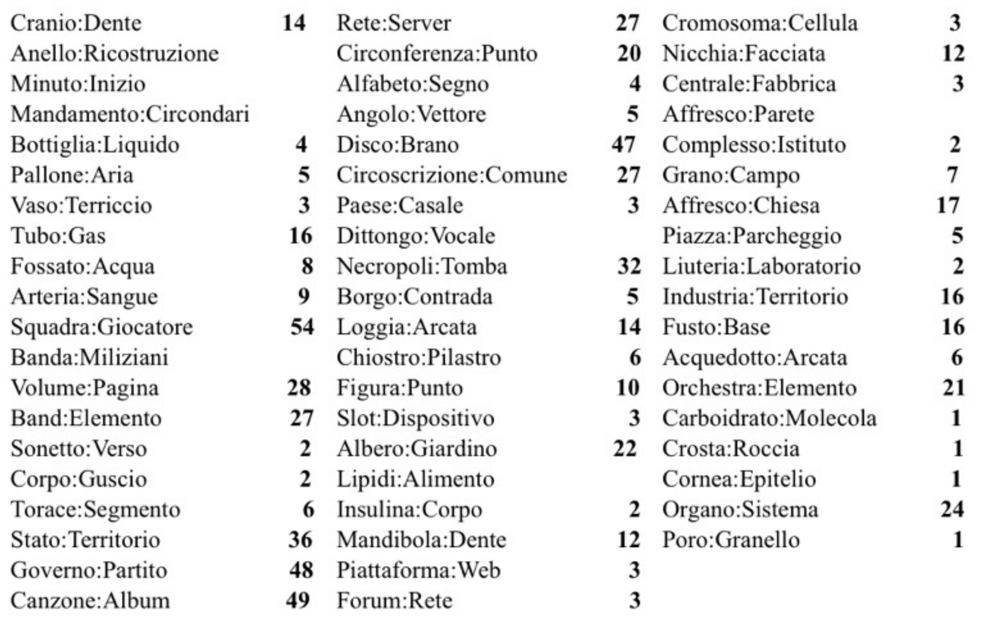
Pianeta/Spazio	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
VICINO A	2	464	29	0,00000042999	0,00009975734	0,00000623483	0,00000000062	9,4332365408	18,8664730816
IN DIREZIONE DI	1	270	29	0,00000021499	0,00005804845	0,00000623483	0,00000000036	9,21440193888	9,21440193888
NEI PRESSI DI	2	645	29	0,00000042999	0,0001386713	0,00000623483	0,00000000086	8,95806218562	17,9161243712
FRA	1	4101	29	0,00000021499	0,00088169145	0,00000623483	0,0000000055	5,2894575076	5,2894575076
IN	10	514275	29	0,00000214994	0,11056617233	0,00000623483	0,00000068936	1,64096513345	16,4096513345
DI	11	2566235	29	0,00000236494	0,55172579116	0,00000623483	0,00000343992	-0,54057270437	-5,94629974803
A	2	471763	29	0,00000042999	0,10142633641	0,00000623483	0,00000063238	-0,55648521209	-1,11297042418
Opera/Museo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
ACQUISIRE DA	1	123	37	0,00000021499	0,00002644429	0,00000795479	0,00000000021	9,99723066009	9,99723066009
CONSERVARE IN	2	257	37	0,00000042999	0,00005525352	0,00000795479	0,00000000044	9,93412061623	19,8682412325
ESPORRE IN	1	214	37	0,00000021499	0,00004600877	0,00000795479	0,00000000037	9,19827817902	9,19827817902
OSPITARE	2	827	37	0,00000042999	0,00017780025	0,00000795479	0,00000000141	8,24800164625	16,4960032925
PORTARE IN	1	519	37	0,00000021499	0,00011158202	0,00000795479	0,00000000089	7,92015443707	7,92015443707
PRESENTE IN	2	3014	37	0,00000042999	0,0006479927	0,00000795479	0,00000000515	6,38228146383	12,7645629277
PROVENIRE DA	1	2358	37	0,00000021499	0,00050695646	0,00000795479	0,00000000403	5,73639716244	5,73639716244
CONTENERE	1	3571	37	0,00000021499	0,0007677445	0,00000795479	0,00000000611	5,13763274684	5,13763274684
IN	14	514275	37	0,00000300992	0,11056617233	0,00000795479	0,00000087953	1,77491959012	24,8488742617
AVERE	1	38645	37	0,00000021499	0,00830845312	0,00000795479	0,00000006609	1,70175101964	1,70175101964
CON	2	89707	37	0,00000042999	0,01928648995	0,00000795479	0,00000015342	1,48681222023	2,97362444045
A	5	471763	37	0,00000107497	0,10142633641	0,00000795479	0,00000080682	0,41397051229	2,06985256147
DA	1	245585	37	0,00000021499	0,05279936499	0,00000795479	0,00000042001	-0,96611775455	-0,96611775455
DI	3	2566235	37	0,00000064498	0,55172579116	0,00000795479	0,00000438886	-2,76651419278	-8,29954257835
Libro/ Biblioteca	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PIENO DI	2	1121	30	0,00000042999	0,00024100856	0,00000644983	0,00000000155	8,11173737264	16,2234747453
RACCOGLIERE	1	1016	30	0,00000021499	0,00021843417	0,00000644983	0,00000000141	7,25362324867	7,25362324867
PROVENIRE DA	1	2358	30	0,00000021499	0,00050695646	0,00000644983	0,00000000327	6,03895993247	6,03895993247
PRESENTE IN	1	3014	30	0,00000021499	0,0006479927	0,00000644983	0,00000000418	5,68484423385	5,68484423385
CONTENERE	1	3571	30	0,00000021499	0,0007677445	0,00000644983	0,00000000495	5,44019551686	5,44019551686
DI	21	2566235	30	0,00000451488	0,55172579116	0,00000644983	0,00000355854	0,34340349929	7,21147348517
IN	2	514275	30	0,00000042999	0,11056617233	0,00000644983	0,00000071313	-0,72987256192	-1,45974512383
A	1	471763	30	0,00000021499	0,10142633641	0,00000644983	0,00000065418	-1,60539481257	-1,60539481257

Figure 5: Location Relation expressed through seeds



LOCAZIONE	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
AVERE RISUCCHIATO	1	1	568	0,0000002424754	0,0000002424754	0,0001377260174	3,33952E-11	12,825911259587	12,825911259587
AD ORIENTE DI	1	4	568	0,0000002424754	0,0000009699015	0,0001377260174	0,0000000001336	10,825911259587	10,825911259587
INCERNIERARE A	1	5	568	0,0000002424754	0,0000012123769	0,0001377260174	0,000000000167	10,503983164699	10,503983164699
LOCALIZZARE SU	4	24	568	0,0000009699015	0,0000058194092	0,0001377260174	0,0000000008015	10,240948758865	40,963795035461
AVERE LUOGO SU	1	10	568	0,0000002424754	0,0000024247538	0,0001377260174	0,000000000334	9,5039831646991	9,5039831646991
ACCESSIBILE DA	1	25	568	0,0000002424754	0,0000060618846	0,0001377260174	0,0000000008349	8,1820550698117	8,1820550698117
LIMITARE A	1	27	568	0,0000002424754	0,0000065468353	0,0001377260174	0,0000000009017	8,071023757423	8,071023757423
STRAPPARE A	1	32	568	0,0000002424754	0,0000077592122	0,0001377260174	0,0000000010686	7,8259112595865	7,8259112595865
COSTELLATO DI	1	34	568	0,0000002424754	0,000008244163	0,0001377260174	0,0000000011354	7,7384484183361	7,7384484183361
ESSERE CUSTODITO IN	1	36	568	0,0000002424754	0,0000087291138	0,0001377260174	0,0000000012022	7,6559862581441	7,6559862581441
SORGERE DA	1	44	568	0,0000002424754	0,0000106689168	0,0001377260174	0,0000000014694	7,3664796409492	7,3664796409492
VENIRE COLLOCATI	1	46	568	0,0000002424754	0,0000111538676	0,0001377260174	0,0000000015362	7,3023493035294	7,3023493035294
VENIRE ERETTO	1	46	568	0,0000002424754	0,0000111538676	0,0001377260174	0,0000000015362	7,3023493035294	7,3023493035294
FISSARE SU	1	48	568	0,0000002424754	0,0000116388184	0,0001377260174	0,000000001603	7,2409487588653	7,2409487588653
COSPARSO DI	1	50	568	0,0000002424754	0,0000121237691	0,0001377260174	0,0000000016698	7,1820550698117	7,1820550698117
CONSERVARE IN	5	257	568	0,0000012123769	0,0000623161734	0,0001377260174	0,0000000085826	7,1422148052799	35,7110740264
SITO IN	2	121	568	0,0000004849508	0,0000293395213	0,0001377260174	0,0000000040408	6,9070480223119	13,814096044624
SITUARE SU	4	338	568	0,0000009699015	0,0000819566793	0,0001377260174	0,0000000112876	6,4250318233043	25,700127293217
MONTARE A	1	106	568	0,0000002424754	0,0000257023906	0,0001377260174	0,0000000035399	6,0979908050233	6,0979908050233
ESPORRE IN	2	214	568	0,0000004849508	0,0000518897319	0,0001377260174	0,0000000071466	6,0844442731853	12,168888546371
IN CENTRO DI	2	227	568	0,0000004849508	0,0000550419119	0,0001377260174	0,0000000075807	5,9993627722955	11,998725544591
COLLOCARE IN	2	231	568	0,0000004849508	0,0000560118134	0,0001377260174	0,0000000077143	5,9741622181704	11,948324436341
ACQUISIRE DA	1	123	568	0,0000002424754	0,0000298244721	0,0001377260174	0,0000000041076	5,8833967542472	5,8833967542472
FISSARE A	1	127	568	0,0000002424754	0,0000307943736	0,0001377260174	0,0000000042412	5,8372265728143	5,8372265728143
OSPITARE	6	827	568	0,0000014548523	0,0002005271415	0,0001377260174	0,0000000276178	5,7191302411363	34,314781446818
TRASFERIRE A	1	140	568	0,0000002424754	0,0000339465536	0,0001377260174	0,0000000046753	5,6966282426415	5,6966282426415
TROVARE IN	1	140	568	0,0000002424754	0,0000339465536	0,0001377260174	0,0000000046753	5,6966282426415	5,6966282426415
LASCIARE A	1	173	568	0,0000002424754	0,0000419482412	0,0001377260174	0,0000000057774	5,3912830319497	5,3912830319497
A RIDOSSO DI	1	185	568	0,0000002424754	0,0000448579458	0,0001377260174	0,0000000061781	5,2945297990701	5,2945297990701
ADIACENTE A	1	205	568	0,0000002424754	0,0000497074535	0,0001377260174	0,000000006846	5,146431160081	5,146431160081
IN VICINANZA DI	1	213	568	0,0000002424754	0,0000516472565	0,0001377260174	0,0000000071132	5,0912016393606	5,0912016393606
NEI PRESSI DI	3	645	568	0,0000007274261	0,0001563966218	0,0001377260174	0,0000000215399	5,077718409997	15,233155229991
ESSERE PRESENTE	7	1560	568	0,0000016973277	0,000378261597	0,0001377260174	0,0000000520965	5,0259358678945	35,181551075261
PIENO DI	5	1121	568	0,0000012123769	0,000271814904	0,0001377260174	0,000000037436	5,0172687916684	25,086343958342
COME	1	231	568	0,0000002424754	0,0000560118134	0,0001377260174	0,0000000077143	4,9741622181704	4,9741622181704
VICINO A	2	464	568	0,0000004849508	0,0001125085776	0,0001377260174	0,0000000154954	4,9679302644589	9,9358605289178
IN DIREZIONE DI	1	270	568	0,0000002424754	0,0000654683533	0,0001377260174	0,0000000090167	4,7490956625356	4,7490956625356
IN MEZZO A	1	298	568	0,0000002424754	0,000072257664	0,0001377260174	0,0000000099518	4,6067427391243	4,6067427391243
PRESENTE IN	10	3014	568	0,0000024247538	0,0007308208034	0,0001377260174	0,000000100653	4,590375652876	45,90375652876
POSTO SU	1	348	568	0,0000002424754	0,0000843814332	0,0001377260174	0,0000000116215	4,3829677637377	4,3829677637377
CIRCONDARE DA	2	773	568	0,0000004849508	0,0001874334708	0,0001377260174	0,0000000258145	4,2315866556616	8,4631733113232
ESSERE PRESENTE	4	1560	568	0,0000009699015	0,000378261597	0,0001377260174	0,0000000520965	4,2185809458368	16,874323783347
A NORD DI	1	425	568	0,0000002424754	0,0001030520376	0,0001377260174	0,0000000141929	4,0945922285614	4,0945922285614
RACCOGLIERE	2	1016	568	0,0000004849508	0,0002463549888	0,0001377260174	0,0000000339295	3,8372265728143	7,6744531456286
PORTARE IN	1	519	568	0,0000002424754	0,0001258447236	0,0001377260174	0,0000000173321	3,8063205312286	3,8063205312286
RIVOLGERE A	1	535	568	0,0000002424754	0,0001297243297	0,0001377260174	0,0000000178664	3,762516178298	3,762516178298
AL POSTO DI	1	639	568	0,0000002424754	0,0001549417695	0,0001377260174	0,0000000213395	3,5062391386395	3,5062391386395
POSSEDERE	2	1324	568	0,0000004849508	0,0003210374067	0,0001377260174	0,0000000442152	3,4552238527792	6,9104477055585
AL CENTRO DI	1	810	568	0,0000002424754	0,00019640506	0,0001377260174	0,0000000270501	3,1641331618145	3,1641331618145
CONTENENTE	1	1114	568	0,0000002424754	0,0002701175763	0,0001377260174	0,0000000372022	2,7043777422464	2,7043777422464
OFFRIRE	1	1164	568	0,0000002424754	0,0002822413454	0,0001377260174	0,000000038872	2,6410359166782	2,6410359166782
PROVENIRE DA	2	2358	568	0,0000004849508	0,0005717569524	0,0001377260174	0,0000000787458	2,6225632566067	5,2451265132134
SITUARE IN	7	9804	568	0,0000016973277	0,0023772286519	0,0001377260174	0,0000003274062	2,3741114127772	16,618779889441
PRODURRE DA	1	1448	568	0,0000002424754	0,0003511043541	0,0001377260174	0,0000000483562	2,3260653725033	2,3260653725033
RICCO DI	1	1469	568	0,0000002424754	0,0003561963372	0,0001377260174	0,0000000490575	2,3052925790302	2,3052925790302
CONTENERE	2	3571	568	0,0000004849508	0,0008658795916	0,0001377260174	0,0000001192541	2,0237988410004	4,0475976820008
ATTRAVERSO	1	2640	568	0,0000002424754	0,0006401350103	0,0001377260174	0,0000000881632	1,4595890453406	1,4595890453406
CON	27	89707	568	0,0000065468353	0,0217517391547	0,0001377260174	0,0000029957804	1,1278658165514	30,452377046887
FRA	1	4101	568	0,0000002424754	0,0009943915444	0,0001377260174	0,0000001369536	0,8241512312592	0,8241512312592
SENZA	2	8659	568	0,0000004849508	0,0020995943387	0,0001377260174	0,0000002891688	0,7459265525525	1,491853105105
SU	25	118500	568	0,0000060618846	0,0287333328484	0,0001377260174	0,0000039573275	0,6152399158008	15,380997895021
IN	88	514275	568	0,0000213378337	0,1246990274312	0,0001377260174	0,0000171743004	0,313162380862	27,558289515857
FINO A	1	6588	568	0,0000002424754	0,0015974278211	0,0001377260174	0,0000002200074	0,1402864198601	0,1402864198601
DA	35	245585	568	0,0000084866384	0,0595483168571	0,0001377260174	0,0000082013525	0,0493313565595	1,7265974795824
DA PARTE DI	1	7150	568	0,0000002424754	0,0017336989862	0,0001377260174	0,0000002387755	0,0221837330333	0,0221837330333
DI	234	2566235	568	0,0000567392395	0,6222488136892	0,0001377260174	0,0000856998509	-0,59494587976	-139,2173358638
O	4	47222	568	0,0000009699015	0,0114501725212	0,0001377260174	0,0000015769867	-0,701260265489	-2,805041061957
AVERE	3	38645	568	0,0000007274261	0,0093704611639	0,0001377260174	0,0000012905563	-0,827120385479	-2,481361156437
A	32	471763	568	0,0000077592122	0,1143909139625	0,0001377260174	0,000015754605	-1,02179148843	-32,69732762977
E	12	425665	568	0,0000029097046	0,1032132837714	0,0001377260174	0,0000142151545	-2,288485184558	-27,4618222147
PER	1	65063	568	0,0000002424754	0,0157761758237	0,0001377260174	0,0000021727899	-3,163638465326	-3,163638465326

Figure 6: Patterns expressing location relations



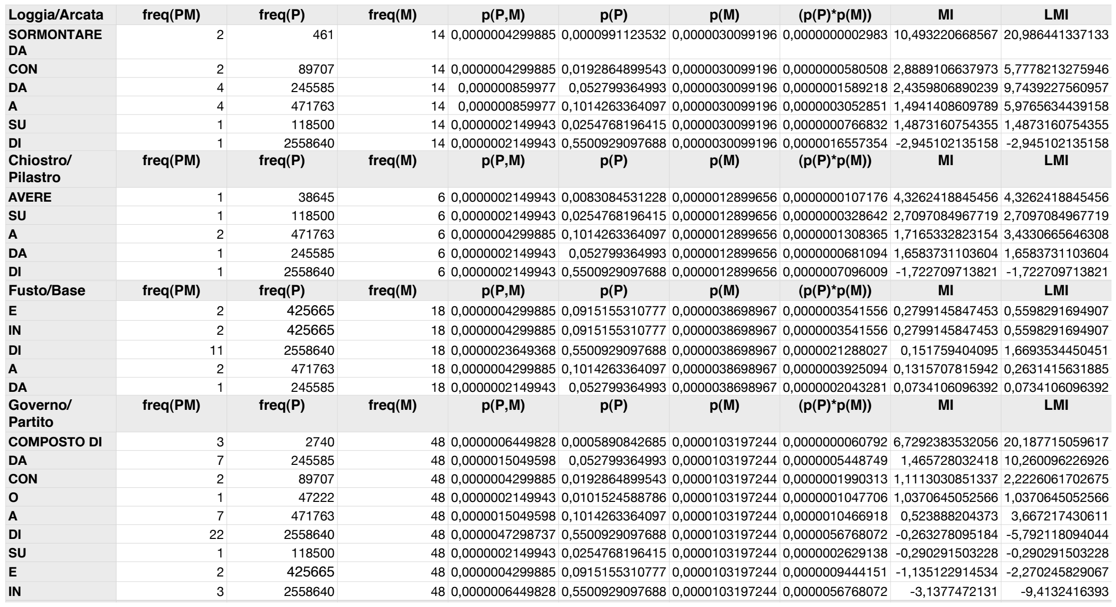
Cranio:Dente	**14**	Rete:Server	**27**	Cromosoma:Cellula	**3**
Anello:Ricostruzione		Circonferenza:Punto	**20**	Nicchia:Facciata	**12**
Minuto:Inizio		Alfabeto:Segno	**4**	Centrale:Fabbrica	**3**
Mandamento:Circondari		Angolo:Vettore	**5**	Affresco:Parete	
Bottiglia:Liquido	**4**	Disco:Brano	**47**	Complesso:Istituto	**2**
Pallone:Aria	**5**	Circoscrizione:Comune	**27**	Grano:Campo	**7**
Vaso:Terriccio	**3**	Paese:Casale	**3**	Affresco:Chiesa	**17**
Tubo:Gas	**16**	Dittongo:Vocale		Piazza:Parcheggio	**5**
Fossato:Acqua	**8**	Necropoli:Tomba	**32**	Liuteria:Laboratorio	**2**
Arteria:Sangue	**9**	Borgo:Contrada	**5**	Industria:Territorio	**16**
Squadra:Giocatore	**54**	Loggia:Arcata	**14**	Fusto:Base	**16**
Banda:Miliziani		Chiostro:Pilastro	**6**	Acquedotto:Arcata	**6**
Volume:Pagina	**28**	Figura:Punto	**10**	Orchestra:Elemento	**21**
Band:Elemento	**27**	Slot:Dispositivo	**3**	Carboidrato:Molecola	**1**
Sonetto:Verso	**2**	Albero:Giardino	**22**	Crosta:Roccia	**1**
Corpo:Guscio	**2**	Lipidi:Alimento		Cornea:Epitelio	**1**
Torace:Segmento	**6**	Insulina:Corpo	**2**	Organo:Sistema	**24**
Stato:Territorio	**36**	Mandibola:Dente	**12**	Poro:Granello	**1**
Governo:Partito	**48**	Piattaforma:Web	**3**		
Canzone:Album	**49**	Forum:Rete	**3**		

FIGURE 7: List of all the words used as seeds for the relation classification experiment and their frequencies with our patterns

Acquedotto/ Arcata	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
SU	1	118500	6	0,0000002149943	0,0254768196415	0,0000012899656	0,0000000328642	2,7097084967719	2,7097084967719
DA	1	245585	6	0,0000002149943	0,052799364993	0,0000012899656	0,0000000681094	1,6583731103604	1,6583731103604
A	1	471763	6	0,0000002149943	0,1014263364097	0,0000012899656	0,0000001308365	0,7165332823154	0,7165332823154
DI	3	2558640	6	0,0000006449828	0,5500929097688	0,0000012899656	0,0000007096009	-0,1377472131	-0,4132416393
Circonferenza/ Punto	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SU	2	118500	20	0,0000004299885	0,0254768196415	0,0000042998852	0,0000001095474	1,9727429026057	3,9454858052115
PER	1	65063	20	0,0000002149943	0,0139881714459	0,0000042998852	0,0000000601475	1,8377207112536	1,8377207112536
CON	1	89707	20	0,0000002149943	0,0192864899543	0,0000042998852	0,0000000829297	1,3743374909675	1,3743374909675
IN	3	514275	20	0,0000006449828	0,1105661723304	0,0000042998852	0,0000004754218	0,4400524395255	1,3201573185765
A	2	471763	20	0,0000004299885	0,1014263364097	0,0000042998852	0,0000004361216	-0,020432311851	-0,040864623702
DA	1	245585	20	0,0000002149943	0,052799364993	0,0000042998852	0,0000002270312	-0,078592483806	-0,078592483806
DI	10	2558640	20	0,0000021499426	0,5500929097688	0,0000042998852	0,0000023653363	-0,1377472131	-1,377472131
Necropoli/ Tomba	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	1	2740	32	0,0000002149943	0,0005890842685	0,0000068798163	0,0000000040528	5,7292383532056	5,7292383532056
CON	6	89707	32	0,0000012899656	0,0192864899543	0,0000068798163	0,0000001326875	3,2812280865761	19,687368519456
DA	2	245585	32	0,0000004299885	0,052799364993	0,0000068798163	0,0000003632499	0,2433356110815	0,486671222163
E	2	425665	32	0,0000004299885	0,0915155310777	0,0000068798163	0,00000062961	-0,550160413812	-1,100320827625
DI	11	2558640	32	0,0000023649368	0,5500929097688	0,0000068798163	0,0000037845382	-0,678315594463	-7,461471539089
A	2	471763	32	0,0000004299885	0,1014263364097	0,0000068798163	0,0000006977946	-0,698504216963	-1,397008433927
IN	8	2558640	32	0,0000017199541	0,5500929097688	0,0000068798163	0,0000037845382	-1,1377472131	-9,1019777048
Organo/ Sistema	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PRESENTE IN	1	3014	24	0,0000002149943	0,0006479926954	0,0000051598622	0,0000000033436	6,0067723287345	6,0067723287345
PER	1	65063	24	0,0000002149943	0,0139881714459	0,0000051598622	0,000000072177	1,5746863054198	1,5746863054198
E	5	425665	24	0,0000010749713	0,0915155310777	0,0000051598622	0,0000004722075	1,1868051803539	5,9340259017693
CON	1	89707	24	0,0000002149943	0,0192864899543	0,0000051598622	0,0000000995156	1,1113030851337	1,1113030851337
DI	13	2558640	24	0,0000027949254	0,5500929097688	0,0000051598622	0,0000028384036	-0,02226999568	-0,28950994384
A	2	471763	24	0,0000004299885	0,1014263364097	0,0000051598622	0,0000005233459	-0,283466717685	-0,566933435369
IN	1	514275	24	0,0000002149943	0,1105661723304	0,0000051598622	0,0000005705062	-1,407944467029	-1,407944467029
Nicchia/ Facciata	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
AL CENTRO DI	1	810	12	0,0000002149943	0,0001741453494	0,0000025799311	0,0000000004493	8,9024579325603	8,9024579325603
ESSERE PRESENTE	1	1560	12	0,0000002149943	0,0003353910434	0,0000025799311	0,0000000008653	7,9569057165827	7,9569057165827
IN	2	514275	12	0,0000004299885	0,1105661723304	0,0000025799311	0,0000002852531	0,5920555329705	1,1841110659411
DI	7	2558640	12	0,0000015049598	0,5500929097688	0,0000025799311	0,0000014192018	0,0846452082365	0,5925164576553
A	1	471763	12	0,0000002149943	0,1014263364097	0,0000025799311	0,000000261673	-0,283466717685	-0,283466717685
Tubo/Gas	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
RIEMPIRE	1	123	16	0,0000002149943	0,0000264442938	0,0000034399081	9,09659E-11	11,206684025714	11,206684025714
CONTENENTE	1	1114	16	0,0000002149943	0,0002395036041	0,0000034399081	0,0000000008239	8,0276650137134	8,0276650137134
COSTITUIRE DA	1	1114	16	0,0000002149943	0,0002395036041	0,0000034399081	0,0000000008239	8,0276650137134	8,0276650137134
PIENO DI	1	1121	16	0,0000002149943	0,0002410085639	0,0000034399081	0,000000000829	8,018627968248	8,018627968248
CONTENERE	2	3571	16	0,0000004299885	0,0007677444974	0,0000034399081	0,000000002641	7,3470861124674	14,694172224935
CON	1	89707	16	0,0000002149943	0,0192864899543	0,0000034399081	0,0000000663438	1,6962655858549	1,6962655858549
DI	7	2558640	16	0,0000015049598	0,5500929097688	0,0000034399081	0,0000018922691	-0,330392291042	-2,312746037297
A	1	471763	16	0,0000002149943	0,1014263364097	0,0000034399081	0,0000003488973	-0,698504216963	-0,698504216963
IN	2	2558640	16	0,0000004299885	0,5500929097688	0,0000034399081	0,0000018922691	-2,1377472131	-4,2754944262
Stato/Territorio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
O	2	47222	36	0,0000004299885	0,0101524588786	0,0000077397933	0,0000000785779	2,4521020045354	4,9042040090708
COME	1	39157	36	0,0000002149943	0,0084185301832	0,0000077397933	0,0000000651577	1,7222909124673	1,7222909124673
SU	2	118500	36	0,0000004299885	0,0254768196415	0,0000077397933	0,0000001971853	1,1247459960508	2,2494919921016
DI	26	2558640	36	0,0000055898507	0,5500929097688	0,0000077397933	0,0000042576054	0,3927675035988	10,211955093569
IN	4	514275	36	0,000000859977	0,1105661723304	0,0000077397933	0,0000008557593	0,0070930322494	0,0283721289975
A	1	471763	36	0,0000002149943	0,1014263364097	0,0000077397933	0,0000007850189	-1,868429218406	-1,868429218406
Squadra/ Giocatore	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	8	2740	54	0,0000017199541	0,0005890842685	0,00001160969	0,0000000068391	7,9743508510421	63,794806808337
DA	10	245585	54	0,0000021499426	0,052799364993	0,00001160969	0,0000006129843	1,8103762038054	18,103762038054
AVERE	1	38645	54	0,0000002149943	0,0083084531228	0,00001160969	0,0000000964586	1,1563168831033	1,1563168831033
CON	2	89707	54	0,0000004299885	0,0192864899543	0,00001160969	0,0000002239102	0,9413780836914	1,8827561673828
DI	24	2558640	54	0,0000051598622	0,5500929097688	0,00001160969	0,0000063864081	-0,307672214542	-7,384133149015
E	3	425665	54	0,0000006449828	0,0915155310777	0,00001160969	0,0000010624669	-0,720085415255	-2,160256245764
A	2	471763	54	0,0000004299885	0,1014263364097	0,00001160969	0,0000011775283	-1,453391719127	-2,906783438254
IN	4	2558640	54	0,000000859977	0,5500929097688	0,00001160969	0,0000063864081	-2,892634715263	-11,57053886105
Albero/ Giardino	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PRESENTE IN	2	3014	22	0,0000004299885	0,0006479926954	0,0000047298737	0,0000000030649	7,1323032108183	14,264606421637
CON	3	89707	22	0,0000006449828	0,0192864899543	0,0000047298737	0,0000000912227	2,8217964679388	8,4653894038163
IN	5	514275	22	0,0000010749713	0,1105661723304	0,0000047298737	0,000000522964	1,0395145099418	5,1975725497088
DI	11	2558640	22	0,0000023649368	0,5500929097688	0,0000047298737	0,00000260187	-0,1377472131	-1,5152193441
A	1	471763	22	0,0000002149943	0,1014263364097	0,0000047298737	0,0000004797338	-1,157935835601	-1,157935835601
Volume/Pagina	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
ESSERE PRESENTE	1	1560	28	0,0000002149943	0,0003353910434	0,0000060198392	0,000000002019	6,7345132952462	6,7345132952462
DA	2	245585	28	0,0000004299885	0,052799364993	0,0000060198392	0,0000003178437	0,4359806890239	0,8719613780478
DI	19	2558640	28	0,0000040848909	0,5500929097688	0,0000060198392	0,0000033114709	0,302825378286	5,7536821874342
E	3	425665	28	0,0000006449828	0,0915155310777	0,0000060198392	0,0000005509088	0,2274471648512	0,6823414945536
IN	3	514275	28	0,0000006449828	0,1105661723304	0,0000060198392	0,0000006655906	-0,045374387645	-0,136123162934
A	1	471763	28	0,0000002149943	0,1014263364097	0,0000060198392	0,0000006105702	-1,505859139021	-1,505859139021
Mandibola/ Dente	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
AVERE	1	38645	12	0,0000002149943	0,0083084531228	0,0000025799311	0,0000000214352	3,3262418845456	3,3262418845456
PER	1	65063	12	0,0000002149943	0,0139881714459	0,0000025799311	0,0000000360885	2,5746863054198	2,5746863054198
CON	1	89707	12	0,0000002149943	0,0192864899543	0,0000025799311	0,0000000497578	2,1113030851337	2,1113030851337
E	2	425665	12	0,0000004299885	0,0915155310777	0,0000025799311	0,0000002361038	0,8648770854665	1,729754170933
IN	2	514275	12	0,0000004299885	0,1105661723304	0,0000025799311	0,0000002852531	0,5920555329705	1,1841110659411
DI	5	2558640	12	0,0000010749713	0,5500929097688	0,0000025799311	0,0000014192018	-0,400781618934	-2,003908094669

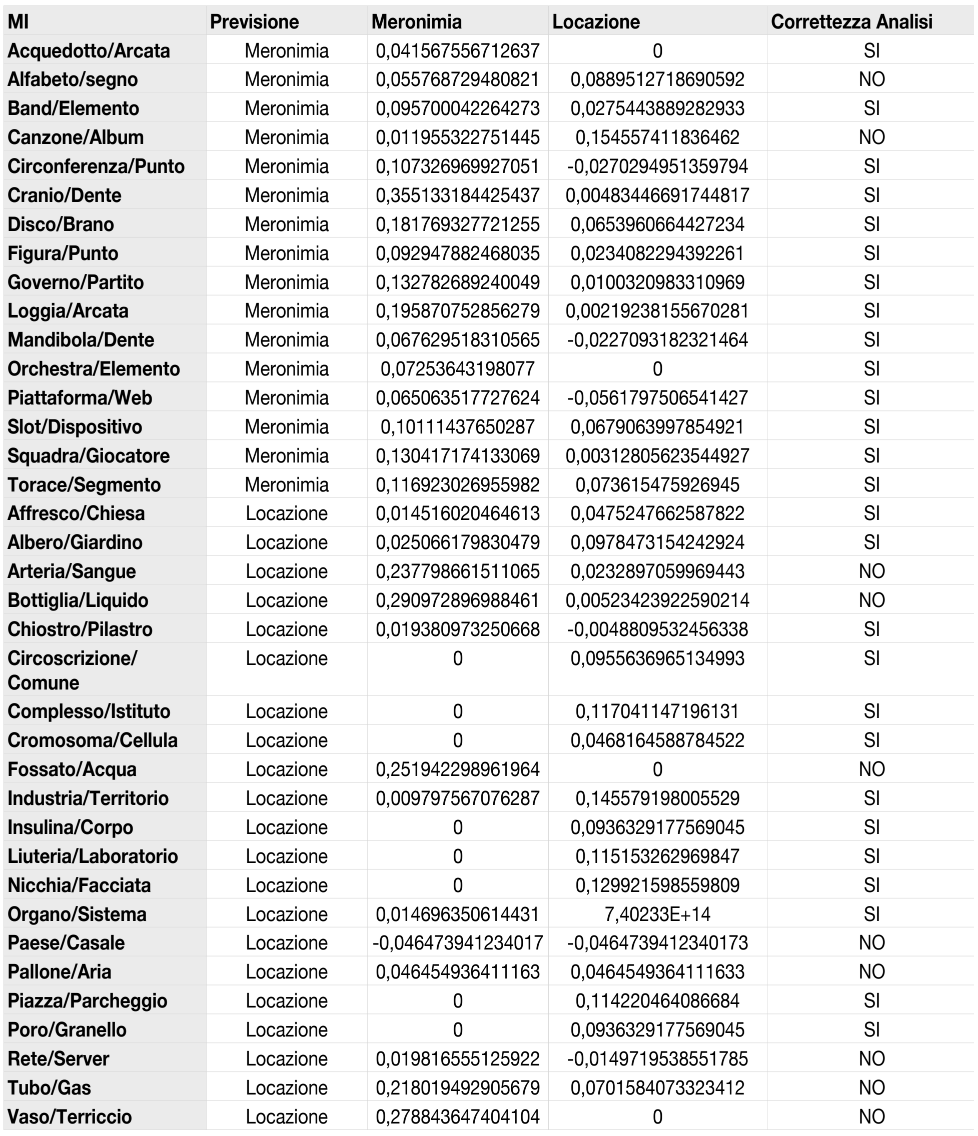
Band/Elemento	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
COMPOSTO DI	3	2740	27	0,0000006449828	0,0005890842685	0,000005804845	0,0000000034195	7,5593133517632	22,67794005529
PROVENIRE DA	1	2358	27	0,0000002149943	0,0005069564617	0,000005804845	0,0000000029428	6,1909630259102	6,1909630259102
AVERE	2	38645	27	0,0000004299885	0,0083084531228	0,000005804845	0,0000000482293	3,1563168831033	6,3126337662066
DA	5	245585	27	0,0000010749713	0,052799364993	0,000005804845	0,0000003064921	1,8103762038054	9,0518810190271
CON	1	89707	27	0,0000002149943	0,0192864899543	0,000005804845	0,0000001119551	0,9413780836914	0,9413780836914
DI	14	2558640	27	0,0000030099196	0,5500929097688	0,000005804845	0,0000031932041	-0,085279793206	-1,193917104882
IN	1	2558640	27	0,0000002149943	0,5500929097688	0,000005804845	0,0000031932041	-3,892634715263	-3,892634715263
Cranio/Dente	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
MANCANTE DI	1	56	14	0,0000002149943	0,0000120396785	0,0000030099196	3,62385E-11	12,534488686938	12,534488686938
DOTATO DI	2	3174	14	0,0000004299885	0,0006823917767	0,0000030099196	0,0000000020539	7,7097571961607	15,419514392321
CON	3	89707	14	0,0000006449828	0,0192864899543	0,0000030099196	0,0000000580508	3,4738731645184	10,421619493555
AVERE	1	38645	14	0,0000002149943	0,0083084531228	0,0000030099196	0,0000000250078	3,1038494632092	3,1038494632092
DI	7	2558640	14	0,0000015049598	0,5500929097688	0,0000030099196	0,0000016557354	-0,1377472131	-0,9642304917
Canzone/ Album	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CONTENERE	7	3571	49	0,0000015049598	0,0007677444974	0,0000105347187	0,000000008088	7,5397311904098	52,778118332868
RACCOGLIERE	1	1016	49	0,0000002149943	0,0002184341667	0,0000105347187	0,0000000023011	6,5458040001661	6,5458040001661
ESSERE PRESENTE	1	1560	49	0,0000002149943	0,0003353910434	0,0000105347187	0,0000000035333	5,9271583731886	5,9271583731886
PRESENTE IN	1	3014	49	0,0000002149943	0,0006479926954	0,0000105347187	0,0000000068264	4,9770249853404	4,9770249853404
CON	5	89707	49	0,0000010749713	0,0192864899543	0,0000105347187	0,0000002031777	2,4034838366271	12,017419183135
O	2	47222	49	0,0000004299885	0,0101524588786	0,0000105347187	0,0000001069533	2,0073171618625	4,014634323725
AVERE	1	38645	49	0,0000002149943	0,0083084531228	0,0000105347187	0,0000000875272	1,2964945411516	1,2964945411516
DA	5	245585	49	0,0000010749713	0,052799364993	0,0000105347187	0,0000005562265	0,9505538618537	4,7527693092684
PER	1	65063	49	0,0000002149943	0,0139881714459	0,0000105347187	0,0000001473615	0,5449389620257	0,5449389620257
IN	6	514275	49	0,0000012899656	0,1105661723304	0,0000105347187	0,0000011647835	0,1472706902976	0,8836241417859
SU	1	118500	49	0,0000002149943	0,0254768196415	0,0000105347187	0,0000002683911	-0,320038846622	-0,320038846622
DI	16	2558640	49	0,0000034399081	0,5500929097688	0,0000105347187	0,000005795074	-0,752457057215	-12,03931291544
E	2	425665	49	0,0000004299885	0,0915155310777	0,0000105347187	0,0000009640904	-1,164870257928	-2,329740515855
Alfabeto/Segno	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COME	1	39157	4	0,0000002149943	0,0084185301832	0,000000859977	0,0000000072397	4,8922159139096	4,8922159139096
CON	1	89707	4	0,0000002149943	0,0192864899543	0,000000859977	0,0000000165859	3,6962655858549	3,6962655858549
DI	2	2558640	4	0,0000004299885	0,5500929097688	0,000000859977	0,0000004730673	-0,1377472131	-0,2754944262
Circoscrizione/ Comune	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COME	1	39157	27	0,0000002149943	0,0084185301832	0,000005804845	0,0000000488683	2,1373284117461	2,1373284117461
O	1	47222	27	0,0000002149943	0,0101524588786	0,000005804845	0,0000000589334	1,8671395038142	1,8671395038142
IN	4	514275	27	0,000000859977	0,1105661723304	0,000005804845	0,0000006418195	0,4221305315282	1,6885221261129
DI	19	2558640	27	0,0000040848909	0,5500929097688	0,000005804845	0,0000031932041	0,3552927981801	6,7505631654228
DA	1	245585	27	0,0000002149943	0,052799364993	0,000005804845	0,0000003064921	-0,511551891082	-0,511551891082
E	1	425665	27	0,0000002149943	0,0915155310777	0,000005804845	0,0000005312335	-1,305047915976	-1,305047915976
Disco/Brano	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	1	2740	47	0,0000002149943	0,0005890842685	0,0000101047302	0,0000000059525	5,1746495015279	5,1746495015279
CON	4	89707	47	0,000000859977	0,0192864899543	0,0000101047302	0,0000001948848	2,1416767341773	8,566706936709
CONTENENTE	2	1114	47	0,0000004299885	0,0002395036041	0,0000101047302	0,0000000024201	7,4730761620358	14,946152324072
COSTITUIRE DA	2	1114	47	0,0000004299885	0,0002395036041	0,0000101047302	0,0000000024201	7,4730761620358	14,946152324072
DA	2	245585	47	0,0000004299885	0,052799364993	0,0000101047302	0,0000005335233	-0,311253240596	-0,622506481192
DI	18	2558640	47	0,0000038698967	0,5500929097688	0,0000101047302	0,0000055585404	-0,522411063335	-9,403399140035
E	1	425665	47	0,0000002149943	0,0915155310777	0,0000101047302	0,0000009247397	-2,10474926549	-2,10474926549
IN	7	514275	47	0,0000015049598	0,1105661723304	0,0000101047302	0,0000011172413	0,4297841040717	3,0084887285017
PRESENTE IN	1	3014	47	0,0000002149943	0,0006479926954	0,0000101047302	0,0000000065478	5,037145977778	5,037145977778
SU	1	118500	47	0,0000002149943	0,0254768196415	0,0000101047302	0,0000002574364	-0,259917854185	-0,259917854185
CONTENERE	8	3571	47	0,0000017199541	0,0007677444974	0,0000101047302	0,0000000077579	7,7924972607897	62,339978086318
Orchestra/ Elemento	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
COMPOSTO DI	1	2740	21	0,0000002149943	0,0005890842685	0,0000045148794	0,0000000026596	6,3369209304268	6,3369209304268
DA	3	245585	21	0,0000006449828	0,052799364993	0,0000045148794	0,0000002383828	1,4359806890239	4,3079420670717
DI	17	2558640	21	0,0000036549024	0,5500929097688	0,0000045148794	0,0000024836032	0,5573982053716	9,4757694913173
Torace/ Segmento	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
POSSEDERE	1	1324	6	0,0000002149943	0,0002846523984	0,0000012899656	0,0000000003672	9,1935486235251	9,1935486235251
CON	1	89707	6	0,0000002149943	0,0192864899543	0,0000012899656	0,0000000248789	3,1113030851337	3,1113030851337
DA	1	245585	6	0,0000002149943	0,052799364993	0,0000012899656	0,0000000681094	1,6583731103604	1,6583731103604
DI	3	2558640	6	0,0000006449828	0,5500929097688	0,0000012899656	0,0000007096009	-0,1377472131	-0,4132416393
Figura/Punto	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CON	1	89707	10	0,0000002149943	0,0192864899543	0,0000021499426	0,0000000414648	2,3743374909675	2,3743374909675
DA	1	245585	10	0,0000002149943	0,052799364993	0,0000021499426	0,0000001135156	0,9214075161942	0,9214075161942
IN	2	514275	10	0,0000004299885	0,1105661723304	0,0000021499426	0,0000002377109	0,8550899388043	1,7101798776087
DI	6	2558640	10	0,0000012899656	0,5500929097688	0,0000021499426	0,0000011826682	0,1252871927338	0,7517231564029
Bottiglia/ Liquido	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
RIEMPIRE	1	123	4	0,0000002149943	0,0000264442938	0,000000859977	2,27415E-11	13,206684025714	13,206684025714
CON	1	89707	4	0,0000002149943	0,0192864899543	0,000000859977	0,0000000165859	3,6962655858549	3,6962655858549
DI	1	2558640	4	0,0000002149943	0,5500929097688	0,000000859977	0,0000004730673	-1,1377472131	-1,1377472131
IN	1	2558640	4	0,0000002149943	0,5500929097688	0,000000859977	0,0000004730673	-1,1377472131	-1,1377472131
Affresco/ Chiesa	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CONTENERE	1	3571	17	0,0000002149943	0,0007677444974	0,0000036549024	0,000000002806	6,259623271217	6,259623271217
IN	9	514275	17	0,0000019349483	0,1105661723304	0,0000036549024	0,0000004041086	2,2594801938837	20,335321744953
CON	1	89707	17	0,0000002149943	0,0192864899543	0,0000036549024	0,0000000704902	1,6088027446046	1,6088027446046
DA	1	245585	17	0,0000002149943	0,052799364993	0,0000036549024	0,0000001929765	0,1558727698312	0,1558727698312
DI	5	2558640	17	0,0000010749713	0,5500929097688	0,0000036549024	0,0000020105359	-0,903281959463	-4,516409797315
Carboidrato/ Molecola	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	1	245585	1	0,0000002149943	0,052799364993	0,0000002149943	0,0000000113516	4,2433356110815	4,2433356110815
Cornea/Epitelio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	1	245585	1	0,0000002149943	0,052799364993	0,0000002149943	0,0000000113516	4,2433356110815	4,2433356110815
Angolo/Vettore	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	4	245585	5	0,000000859977	0,052799364993	0,0000010749713	0,0000000567578	3,9214075161942	15,685630064777
FRA	1	4101	5	0,0000002149943	0,0008816914544	0,0000010749713	0,0000000009478	7,8255104078389	7,8255104078389

Fossato/Acqua	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
RIEMPIRE	1	123	8	0,0000002149943	0,0000264442938	0,0000017199541	4,5483E-11	12,206684025714	12,206684025714
O	1	47222	8	0,0000002149943	0,0101524588786	0,0000017199541	0,0000000174618	3,6220270059777	3,6220270059777
DA	2	245585	8	0,0000004299885	0,052799364993	0,0000017199541	0,0000000908125	2,2433356110815	4,486671222163
IN	3	514275	8	0,0000006449828	0,1105661723304	0,0000017199541	0,0000001901687	1,7619805344129	5,2859416032386
DI	1	2558640	8	0,0000002149943	0,5500929097688	0,0000017199541	0,0000009461345	-2,1377472131	-2,1377472131
Corpo/Guscio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	1	245585	2	0,0000002149943	0,052799364993	0,0000004299885	0,0000000227031	3,2433356110815	3,2433356110815
DI	1	2558640	2	0,0000002149943	0,5500929097688	0,0000004299885	0,0000002365336	-0,1377472131	-0,1377472131
Sonetto/Verso	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	1	245585	2	0,0000002149943	0,052799364993	0,0000004299885	0,0000000227031	3,2433356110815	3,2433356110815
DI	4	2558640	2	0,000000859977	0,5500929097688	0,0000004299885	0,0000002365336	1,8622527869	7,4490111476001
Borgo/ Contrada	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DA	1	245585	5	0,0000002149943	0,052799364993	0,0000010749713	0,0000000567578	1,9214075161942	1,9214075161942
DI	3	2558640	5	0,0000006449828	0,5500929097688	0,0000010749713	0,0000005913341	0,1252871927338	0,3758615782015
IN	1	514275	5	0,0000002149943	0,1105661723304	0,0000010749713	0,0000001188555	0,8550899388043	0,8550899388043
Pallone/Aria	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CONTENERE	1	3571	5	0,0000002149943	0,0007677444974	0,0000010749713	0,0000000008253	8,02515801758	8,02515801758
DA	1	245585	5	0,0000002149943	0,052799364993	0,0000010749713	0,0000000567578	1,9214075161942	1,9214075161942
IN	2	2558640	5	0,0000004299885	0,5500929097688	0,0000010749713	0,0000005913341	-0,459675307987	-0,919350615975
Rete/Server	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SENZA	1	8659	27	0,0000002149943	0,001861635285	0,000005804845	0,0000000108065	4,314326321856	4,314326321856
SU	2	118500	27	0,0000004299885	0,0254768196415	0,000005804845	0,000000147889	1,5397834953296	3,0795669906593
PER	1	65063	27	0,0000002149943	0,0139881714459	0,000005804845	0,0000000811992	1,4047613039775	1,4047613039775
E	3	425665	27	0,0000006449828	0,0915155310777	0,000005804845	0,0000005312335	0,2799145847453	0,839743754236
CON	16	2558640	27	0,0000034399081	0,5500929097688	0,000005804845	0,0000031932041	0,1073652847366	1,717844555785
DA	1	245585	27	0,0000002149943	0,052799364993	0,000005804845	0,0000003064921	-0,511551891082	-0,511551891082
IN	3	2558640	27	0,0000006449828	0,5500929097688	0,000005804845	0,0000031932041	-2,307672214542	-6,923016643627
Industria/ Territorio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
ESSERE PRESENTE	1	1560	16	0,0000002149943	0,0003353910434	0,0000034399081	0,0000000011537	7,5418682173038	7,5418682173038
PRESENTE IN	1	3014	16	0,0000002149943	0,0006479926954	0,0000034399081	0,000000002229	6,5917348294556	6,5917348294556
SU	2	118500	16	0,0000004299885	0,0254768196415	0,0000034399081	0,0000000876379	2,2946709974931	4,5893419949862
IN	5	514275	16	0,0000010749713	0,1105661723304	0,0000034399081	0,0000003803375	1,4989461285791	7,4947306428953
DA	1	245585	16	0,0000002149943	0,052799364993	0,0000034399081	0,000000181625	0,2433356110815	0,2433356110815
DI	6	2558640	16	0,0000012899656	0,5500929097688	0,0000034399081	0,0000018922691	-0,552784712379	-3,316708274273
Vaso/Terriccio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
RIEMPIRE	1	123	3	0,0000002149943	0,0000264442938	0,0000006449828	1,70561E-11	13,621721524993	13,621721524993
DI	2	2558640	3	0,0000004299885	0,5500929097688	0,0000006449828	0,0000003548005	0,2772902861789	0,5545805723577
Grano/Campo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	4	2558640	7	0,000000859977	0,5500929097688	0,0000015049598	0,0000008278677	0,0548978648424	0,2195914593697
SU	3	118500	7	0,0000006449828	0,0254768196415	0,0000015049598	0,0000000383416	4,0722785761567	12,21683572847
Crosta/Roccia	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	2	2558640	1	0,0000004299885	0,5500929097688	0,0000002149943	0,0000001182668	1,8622527869	3,7245055738001
Centrale/ Fabbrica	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
DI	1	2558640	3	0,0000002149943	0,5500929097688	0,0000006449828	0,0000003548005	-0,722709713821	-0,722709713821
IN	2	514275	3	0,0000004299885	0,1105661723304	0,0000006449828	0,0000000713133	2,5920555329705	5,1841110659411
Arteria/Sangue	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
RIEMPIRE	1	123	9	0,0000002149943	0,0000264442938	0,0000019349483	5,11683E-11	12,036759024272	12,036759024272
CONTENERE	1	3571	9	0,0000002149943	0,0007677444974	0,0000019349483	0,0000000014855	7,1771611110251	7,1771611110251
IN	5	514275	9	0,0000010749713	0,1105661723304	0,0000019349483	0,0000002139398	2,3290211271368	11,645105635684
DI	2	2558640	9	0,0000004299885	0,5500929097688	0,0000019349483	0,0000010644014	-1,307672214542	-2,615344429085
Cromosoma/ Cellula	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
CONTENERE	1	3571	3	0,0000002149943	0,0007677444974	0,0000006449828	0,0000000004952	8,7621236117462	8,7621236117462
DI	2	2558640	3	0,0000004299885	0,5500929097688	0,0000006449828	0,0000003548005	0,2772902861789	0,5545805723577
Paese/Casale	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
E	2	425665	3	0,0000004299885	0,0915155310777	0,0000006449828	0,0000000590259	2,8648770854665	5,729754170933
DI	1	2558640	3	0,0000002149943	0,5500929097688	0,0000006449828	0,0000003548005	-0,722709713821	-0,722709713821
Complesso/ Istituto	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
OSPITARE	1	827	2	0,0000002149943	0,0001778002518	0,0000004299885	7,64521E-11	11,457455011882	11,457455011882
DI	1	2558640	2	0,0000002149943	0,5500929097688	0,0000004299885	0,0000002365336	-0,1377472131	-0,1377472131
Piattaforma/ Web	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PER	1	65063	3	0,0000002149943	0,0139881714459	0,0000006449828	0,0000000090221	4,5746863054198	4,5746863054198
SU	1	118500	3	0,0000002149943	0,0254768196415	0,0000006449828	0,0000000164321	3,7097084967719	3,7097084967719
DI	2	2558640	3	0,0000004299885	0,5500929097688	0,0000006449828	0,0000003548005	0,2772902861789	0,5545805723577
Piazza/ Parcheggio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
OSPITARE	1	827	5	0,0000002149943	0,0001778002518	0,0000010749713	0,0000000001911	10,135526916995	10,135526916995
IN	3	514275	5	0,0000006449828	0,1105661723304	0,0000010749713	0,0000001188555	2,4400524395255	7,3201573185765
DI	1	2558640	5	0,0000002149943	0,5500929097688	0,0000010749713	0,0000005913341	-1,459675307987	-1,459675307987
Liuteria/ Laboratorio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
OSPITARE	1	827	2	0,0000002149943	0,0001778002518	0,0000004299885	7,64521E-11	11,457455011882	11,457455011882
IN	1	514275	2	0,0000002149943	0,1105661723304	0,0000004299885	0,0000000475422	2,1770180336917	2,1770180336917
Forum/Rete	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
SU	1	118500	3	0,0000002149943	0,0254768196415	0,0000006449828	0,0000000164321	3,7097084967719	3,7097084967719
IN	2	514275	3	0,0000004299885	0,1105661723304	0,0000006449828	0,0000000713133	2,5920555329705	5,1841110659411
Slot/ Dispositivo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PRESENTE IN	1	3014	3	0,0000002149943	0,0006479926954	0,0000006449828	0,0000000004179	9,0067723287345	9,0067723287345
DOTATO DI	1	3174	3	0,0000002149943	0,0006823917767	0,0000006449828	0,0000000004401	8,9321496174971	8,9321496174971
SU	1	118500	3	0,0000002149943	0,0254768196415	0,0000006449828	0,0000000164321	3,7097084967719	3,7097084967719
Necropoli/ Tomba	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
IN	8	2558640	8	0,0000017199541	0,5500929097688	0,0000017199541	0,0000009461345	0,8622527869	6,8980222952002
Liuteria/ Laboratorio	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
OSPITARE	1	827	1	0,0000002149943	0,0001778002518	0,0000002149943	3,8226E-11	12,457455011882	12,457455011882
Poro/Granello	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PRESENTE IN	1	3014	1	0,0000002149943	0,0006479926954	0,0000002149943	0,0000000001393	10,591734829456	10,591734829456
Insulina/Corpo	**freq(PM)**	**freq(P)**	**freq(M)**	**p(P,M)**	**p(P)**	**p(M)**	**(p(P)*p(M))**	**MI**	**LMI**
PRESENTE IN	2	3014	2	0,0000004299885	0,0006479926954	0,0000004299885	0,000000002786	10,591734829456	21,183469658911



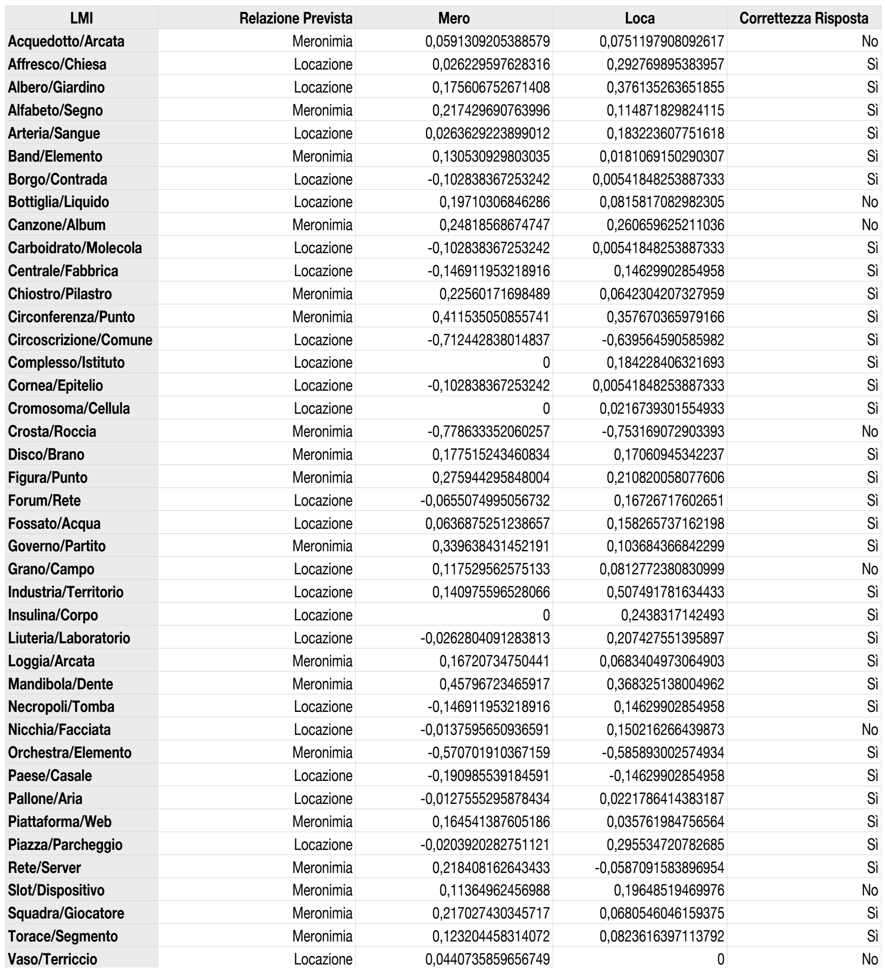
Loggia/Arcata	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
SORMONTARE DA	2	461	14	0,0000004299885	0,0000991123532	0,0000030099196	0,0000000002983	10,493220668567	20,986441337133
CON	2	89707	14	0,0000004299885	0,0192864899543	0,0000030099196	0,0000000580508	2,8889106637973	5,7778213275946
DA	4	245585	14	0,000000859977	0,052799364993	0,0000030099196	0,0000001589218	2,4359806890239	9,7439227560957
A	4	471763	14	0,000000859977	0,1014263364097	0,0000030099196	0,0000003052851	1,4941408609789	5,9765634439158
SU	1	118500	14	0,0000002149943	0,0254768196415	0,0000030099196	0,0000000766832	1,4873160754355	1,4873160754355
DI	1	2558640	14	0,0000002149943	0,5500929097688	0,0000030099196	0,0000016557354	-2,945102135158	-2,945102135158
Chiostro/ Pilastro	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
AVERE	1	38645	6	0,0000002149943	0,0083084531228	0,0000012899656	0,0000000107176	4,3262418845456	4,3262418845456
SU	1	118500	6	0,0000002149943	0,0254768196415	0,0000012899656	0,0000000328642	2,7097084967719	2,7097084967719
A	2	471763	6	0,0000004299885	0,1014263364097	0,0000012899656	0,0000001308365	1,7165332823154	3,4330665646308
DA	1	245585	6	0,0000002149943	0,052799364993	0,0000012899656	0,0000000681094	1,6583731103604	1,6583731103604
DI	1	2558640	6	0,0000002149943	0,5500929097688	0,0000012899656	0,0000007096009	-1,722709713821	-1,722709713821
Fusto/Base	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
E	2	425665	18	0,0000004299885	0,0915155310777	0,0000038698967	0,0000003541556	0,2799145847453	0,5598291694907
IN	2	425665	18	0,0000004299885	0,0915155310777	0,0000038698967	0,0000003541556	0,2799145847453	0,5598291694907
DI	11	2558640	18	0,0000023649368	0,5500929097688	0,0000038698967	0,0000021288027	0,151759404095	1,6693534450451
A	2	471763	18	0,0000004299885	0,1014263364097	0,0000038698967	0,0000003925094	0,1315707815942	0,2631415631885
DA	1	245585	18	0,0000002149943	0,052799364993	0,0000038698967	0,0000002043281	0,0734106096392	0,0734106096392
Governo/ Partito	freq(PM)	freq(P)	freq(M)	p(P,M)	p(P)	p(M)	(p(P)*p(M))	MI	LMI
COMPOSTO DI	3	2740	48	0,0000006449828	0,0005890842685	0,0000103197244	0,0000000060792	6,7292383532056	20,187715059617
DA	7	245585	48	0,0000015049598	0,052799364993	0,0000103197244	0,0000005448749	1,465728032418	10,260096226926
CON	2	89707	48	0,0000004299885	0,0192864899543	0,0000103197244	0,0000001990313	1,1113030851337	2,2226061702675
O	1	47222	48	0,0000002149943	0,0101524588786	0,0000103197244	0,0000001047706	1,0370645052566	1,0370645052566
A	7	471763	48	0,0000015049598	0,1014263364097	0,0000103197244	0,0000010466918	0,523888204373	3,667217430611
DI	22	2558640	48	0,0000047298737	0,5500929097688	0,0000103197244	0,0000056768072	-0,263278095184	-5,792118094044
SU	1	118500	48	0,0000002149943	0,0254768196415	0,0000103197244	0,0000002629138	-0,290291503228	-0,290291503228
E	2	425665	48	0,0000004299885	0,0915155310777	0,0000103197244	0,0000009444151	-1,135122914534	-2,270245829067
IN	3	2558640	48	0,0000006449828	0,5500929097688	0,0000103197244	0,0000056768072	-3,1377472131	-9,4132416393

Figure 8: List of all the retrieved pattern for the test set



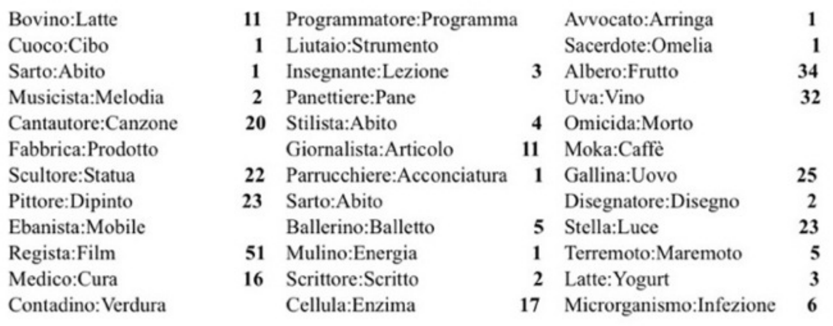
MI	Previsione	Meronimia	Locazione	Correttezza Analisi
Acquedotto/Arcata	Meronimia	0,041567556712637	0	SI
Alfabeto/segno	Meronimia	0,055768729480821	0,0889512718690592	NO
Band/Elemento	Meronimia	0,095700042264273	0,0275443889282933	SI
Canzone/Album	Meronimia	0,011955322751445	0,154557411836462	NO
Circonferenza/Punto	Meronimia	0,107326969927051	-0,0270294951359794	SI
Cranio/Dente	Meronimia	0,355133184425437	0,00483446691744817	SI
Disco/Brano	Meronimia	0,181769327721255	0,0653960664427234	SI
Figura/Punto	Meronimia	0,092947882468035	0,0234082294392261	SI
Governo/Partito	Meronimia	0,132782689240049	0,0100320983310969	SI
Loggia/Arcata	Meronimia	0,195870752856279	0,00219238155670281	SI
Mandibola/Dente	Meronimia	0,067629518310565	-0,0227093182321464	SI
Orchestra/Elemento	Meronimia	0,07253643198077	0	SI
Piattaforma/Web	Meronimia	0,065063517727624	-0,0561797506541427	SI
Slot/Dispositivo	Meronimia	0,10111437650287	0,0679063997854921	SI
Squadra/Giocatore	Meronimia	0,130417174133069	0,00312805623544927	SI
Torace/Segmento	Meronimia	0,116923026955982	0,073615475926945	SI
Affresco/Chiesa	Locazione	0,014516020464613	0,0475247662587822	SI
Albero/Giardino	Locazione	0,025066179830479	0,0978473154242924	SI
Arteria/Sangue	Locazione	0,237798661511065	0,0232897059969443	NO
Bottiglia/Liquido	Locazione	0,290972896988461	0,00523423922590214	NO
Chiostro/Pilastro	Locazione	0,019380973250668	-0,0048809532456338	SI
Circoscrizione/ Comune	Locazione	0	0,0955636965134993	SI
Complesso/Istituto	Locazione	0	0,117041147196131	SI
Cromosoma/Cellula	Locazione	0	0,0468164588784522	SI
Fossato/Acqua	Locazione	0,251942298961964	0	NO
Industria/Territorio	Locazione	0,009797567076287	0,145579198005529	SI
Insulina/Corpo	Locazione	0	0,0936329177569045	SI
Liuteria/Laboratorio	Locazione	0	0,115153262969847	SI
Nicchia/Facciata	Locazione	0	0,129921598559809	SI
Organo/Sistema	Locazione	0,014696350614431	7,40233E+14	SI
Paese/Casale	Locazione	-0,046473941234017	-0,0464739412340173	NO
Pallone/Aria	Locazione	0,046454936411163	0,0464549364111633	NO
Piazza/Parcheggio	Locazione	0	0,114220464086684	SI
Poro/Granello	Locazione	0	0,0936329177569045	SI
Rete/Server	Locazione	0,019816555125922	-0,0149719538551785	NO
Tubo/Gas	Locazione	0,218019492905679	0,0701584073323412	NO
Vaso/Terriccio	Locazione	0,278843647404104	0	NO

Figure 9: List of results accuracy obtained using MI as association measure



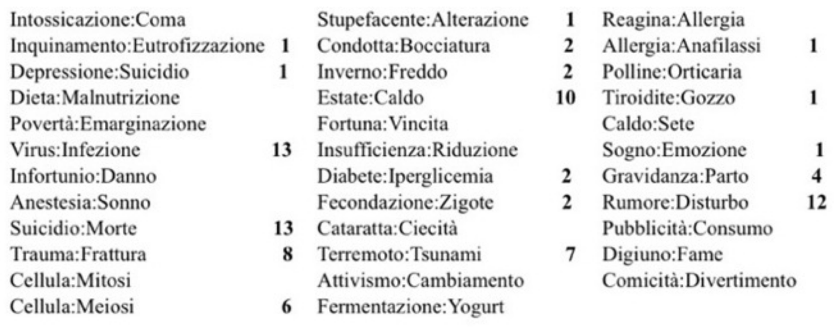
LMI	Relazione Prevista	Mero	Loca	Correttezza Risposta
Acquedotto/Arcata	Meronimia	0,0591309205388579	0,0751197908092617	No
Affresco/Chiesa	Locazione	0,026229597628316	0,292769895383957	Sì
Albero/Giardino	Locazione	0,175606752671408	0,376135263651855	Sì
Alfabeto/Segno	Meronimia	0,217429690763996	0,114871829824115	Sì
Arteria/Sangue	Locazione	0,0263629223899012	0,183223607751618	Sì
Band/Elemento	Meronimia	0,130530929803035	0,0181069150290307	Sì
Borgo/Contrada	Locazione	-0,102838367253242	0,00541848253887333	Sì
Bottiglia/Liquido	Locazione	0,19710306846286	0,0815817082982305	No
Canzone/Album	Meronimia	0,24818568674747	0,260659625211036	No
Carboidrato/Molecola	Locazione	-0,102838367253242	0,00541848253887333	Sì
Centrale/Fabbrica	Locazione	-0,146911953218916	0,14629902854958	Sì
Chiostro/Pilastro	Meronimia	0,22560171698489	0,0642304207327959	Sì
Circonferenza/Punto	Meronimia	0,411535050855741	0,357670365979166	Sì
Circoscrizione/Comune	Locazione	-0,712442838014837	-0,639564590585982	Sì
Complesso/Istituto	Locazione	0	0,184228406321693	Sì
Cornea/Epitelio	Locazione	-0,102838367253242	0,00541848253887333	Sì
Cromosoma/Cellula	Locazione	0	0,0216739301554933	Sì
Crosta/Roccia	Meronimia	-0,778633352060257	-0,753169072903393	No
Disco/Brano	Meronimia	0,177515243460834	0,17060945342237	Sì
Figura/Punto	Meronimia	0,275944295848004	0,210820058077606	Sì
Forum/Rete	Locazione	-0,0655074995056732	0,16726717602651	Sì
Fossato/Acqua	Locazione	0,0636875251238657	0,158265737162198	Sì
Governo/Partito	Meronimia	0,339638431452191	0,103684366842299	Sì
Grano/Campo	Locazione	0,117529562575133	0,0812772380830999	No
Industria/Territorio	Locazione	0,140975596528066	0,507491781634433	Sì
Insulina/Corpo	Locazione	0	0,2438317142493	Sì
Liuteria/Laboratorio	Locazione	-0,0262804091283813	0,207427551395897	Sì
Loggia/Arcata	Meronimia	0,16720734750441	0,0683404973064903	Sì
Mandibola/Dente	Meronimia	0,45796723465917	0,368325138004962	Sì
Necropoli/Tomba	Locazione	-0,146911953218916	0,14629902854958	Sì
Nicchia/Facciata	Locazione	-0,0137595650936591	0,150216266439873	No
Orchestra/Elemento	Meronimia	-0,570701910367159	-0,585893002574934	Sì
Paese/Casale	Locazione	-0,190985539184591	-0,14629902854958	Sì
Pallone/Aria	Locazione	-0,0127555295878434	0,0221786414383187	Sì
Piattaforma/Web	Meronimia	0,164541387605186	0,035761984756564	Sì
Piazza/Parcheggio	Locazione	-0,0203920282751121	0,295534720782685	Sì
Rete/Server	Meronimia	0,218408162643433	-0,0587091583896954	Sì
Slot/Dispositivo	Meronimia	0,11364962456988	0,19648519469976	No
Squadra/Giocatore	Meronimia	0,217027430345717	0,0680546046159375	Sì
Torace/Segmento	Meronimia	0,123204458314072	0,0823616397113792	Sì
Vaso/Terriccio	Locazione	0,0440735859656749	0	No

Figure 10: List of results accuracy obtained using LMI as association measure



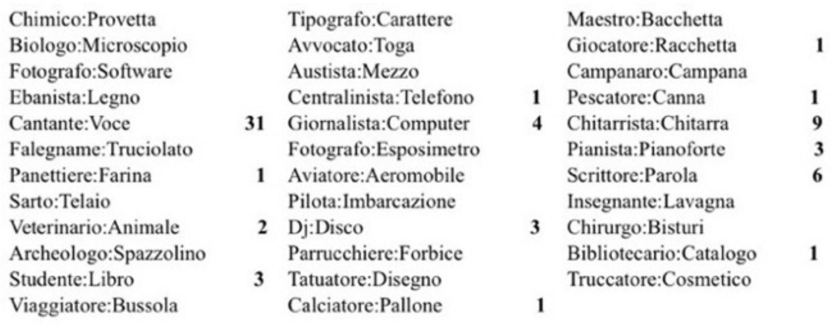
Bovino:Latte	**11**	Programmatore:Programma		Avvocato:Arringa	**1**
Cuoco:Cibo	**1**	Liutaio:Strumento		Sacerdote:Omelia	**1**
Sarto:Abito	**1**	Insegnante:Lezione	**3**	Albero:Frutto	**34**
Musicista:Melodia	**2**	Panettiere:Pane		Uva:Vino	**32**
Cantautore:Canzone	**20**	Stilista:Abito	**4**	Omicida:Morto	
Fabbrica:Prodotto		Giornalista:Articolo	**11**	Moka:Caffè	
Scultore:Statua	**22**	Parrucchiere:Acconciatura	**1**	Gallina:Uovo	**25**
Pittore:Dipinto	**23**	Sarto:Abito		Disegnatore:Disegno	**2**
Ebanista:Mobile		Ballerino:Balletto	**5**	Stella:Luce	**23**
Regista:Film	**51**	Mulino:Energia	**1**	Terremoto:Maremoto	**5**
Medico:Cura	**16**	Scrittore:Scritto	**2**	Latte:Yogurt	**3**
Contadino:Verdura		Cellula:Enzima	**17**	Microrganismo:Infezione	**6**

Figure 11: Manually selected seeds for the training set of Product-Producer relation, and their frequencies in the corpus with our patterns



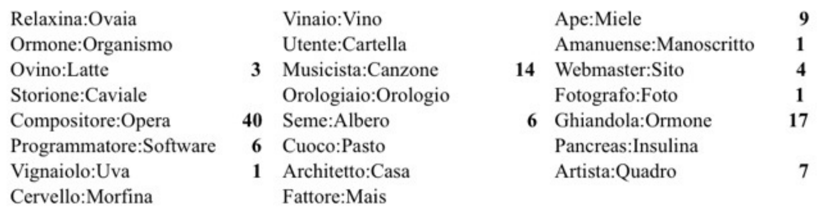
Intossicazione:Coma		Stupefacente:Alterazione	**1**	Reagina:Allergia	
Inquinamento:Eutrofizzazione	**1**	Condotta:Bocciatura	**2**	Allergia:Anafilassi	**1**
Depressione:Suicidio	**1**	Inverno:Freddo	**2**	Polline:Orticaria	
Dieta:Malnutrizione		Estate:Caldo	**10**	Tiroidite:Gozzo	**1**
Povertà:Emarginazione		Fortuna:Vincita		Caldo:Sete	
Virus:Infezione	**13**	Insufficienza:Riduzione		Sogno:Emozione	**1**
Infortunio:Danno		Diabete:Iperglicemia	**2**	Gravidanza:Parto	**4**
Anestesia:Sonno		Fecondazione:Zigote	**2**	Rumore:Disturbo	**12**
Suicidio:Morte	**13**	Cataratta:Cecità		Pubblicità:Consumo	
Trauma:Frattura	**8**	Terremoto:Tsunami	**7**	Digiuno:Fame	
Cellula:Mitosi		Attivismo:Cambiamento		Comicità:Divertimento	
Cellula:Meiosi	**6**	Fermentazione:Yogurt			

Figure 12: Manually selected seeds for the training set of Cause-Effect relation, and their frequencies in the corpus with our patterns



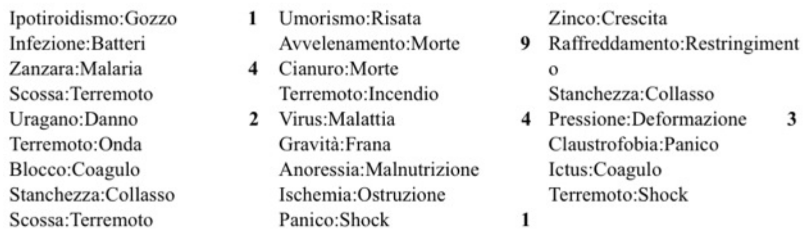
Chimico:Provetta		Tipografo:Carattere		Maestro:Bacchetta	
Biologo:Microscopio		Avvocato:Toga		Giocatore:Racchetta	**1**
Fotografo:Software		Austista:Mezzo		Campanaro:Campana	
Ebanista:Legno		Centralinista:Telefono	**1**	Pescatore:Canna	**1**
Cantante:Voce	**31**	Giornalista:Computer	**4**	Chitarrista:Chitarra	**9**
Falegname:Truciolato		Fotografo:Esposimetro		Pianista:Pianoforte	**3**
Panettiere:Farina	**1**	Aviatore:Aeromobile		Scrittore:Parola	**6**
Sarto:Telaio		Pilota:Imbarcazione		Insegnante:Lavagna	
Veterinario:Animale	**2**	Dj:Disco	**3**	Chirurgo:Bisturi	
Archeologo:Spazzolino		Parrucchiere:Forbice		Bibliotecario:Catalogo	**1**
Studente:Libro	**3**	Tatuatore:Disegno		Truccatore:Cosmetico	
Viaggiatore:Bussola		Calciatore:Pallone	**1**		

Figure 13: Manually selected seeds for the training set of Agency-Instrument relation, and their frequencies in the corpus with our patterns



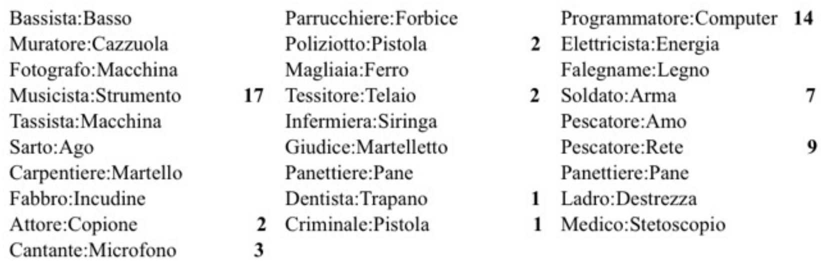
Relaxina:Ovaia		Vinaio:Vino		Ape:Miele	**9**
Ormone:Organismo		Utente:Cartella		Amanuense:Manoscritto	**1**
Ovino:Latte	**3**	Musicista:Canzone	**14**	Webmaster:Sito	**4**
Storione:Caviale		Orologiaio:Orologio		Fotografo:Foto	**1**
Compositore:Opera	**40**	Seme:Albero	**6**	Ghiandola:Ormone	**17**
Programmatore:Software	**6**	Cuoco:Pasto		Pancreas:Insulina	
Vignaiolo:Uva	**1**	Architetto:Casa		Artista:Quadro	**7**
Cervello:Morfina		Fattore:Mais			

Figure 14: Manually selected seeds for the test set of Product-Producer relation, and their frequencies in the corpus with our patterns



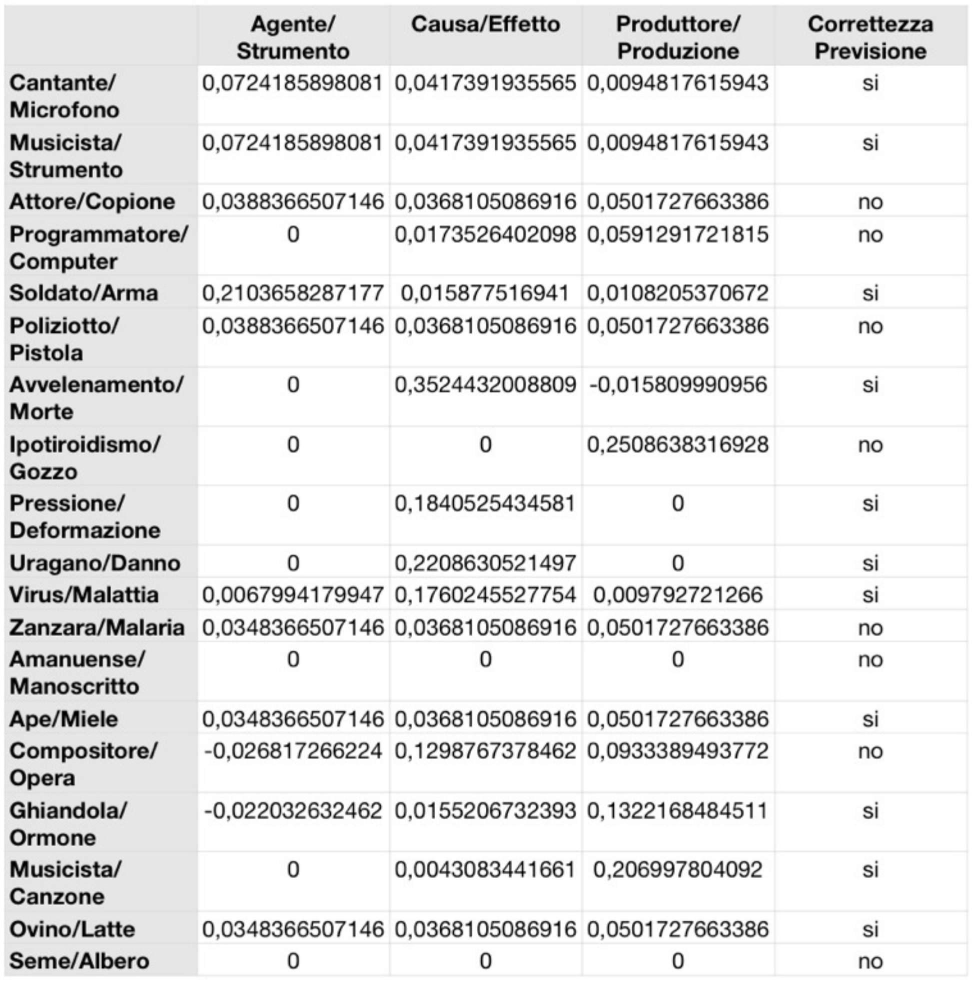
Ipotiroidismo:Gozzo	**1**	Umorismo:Risata		Zinco:Crescita	
Infezione:Batteri		Avvelenamento:Morte	**9**	Raffreddamento:Restringiment	
Zanzara:Malaria	**4**	Cianuro:Morte		o	
Scossa:Terremoto		Terremoto:Incendio		Stanchezza:Collasso	
Uragano:Danno	**2**	Virus:Malattia	**4**	Pressione:Deformazione	**3**
Terremoto:Onda		Gravità:Frana		Claustrofobia:Panico	
Blocco:Coagulo		Anoressia:Malnutrizione		Ictus:Coagulo	
Stanchezza:Collasso		Ischemia:Ostruzione		Terremoto:Shock	
Scossa:Terremoto		Panico:Shock	**1**		

Figure 15: Manually selected seeds for the test set of Cause-Effect relation, and their frequencies in the corpus with our patterns



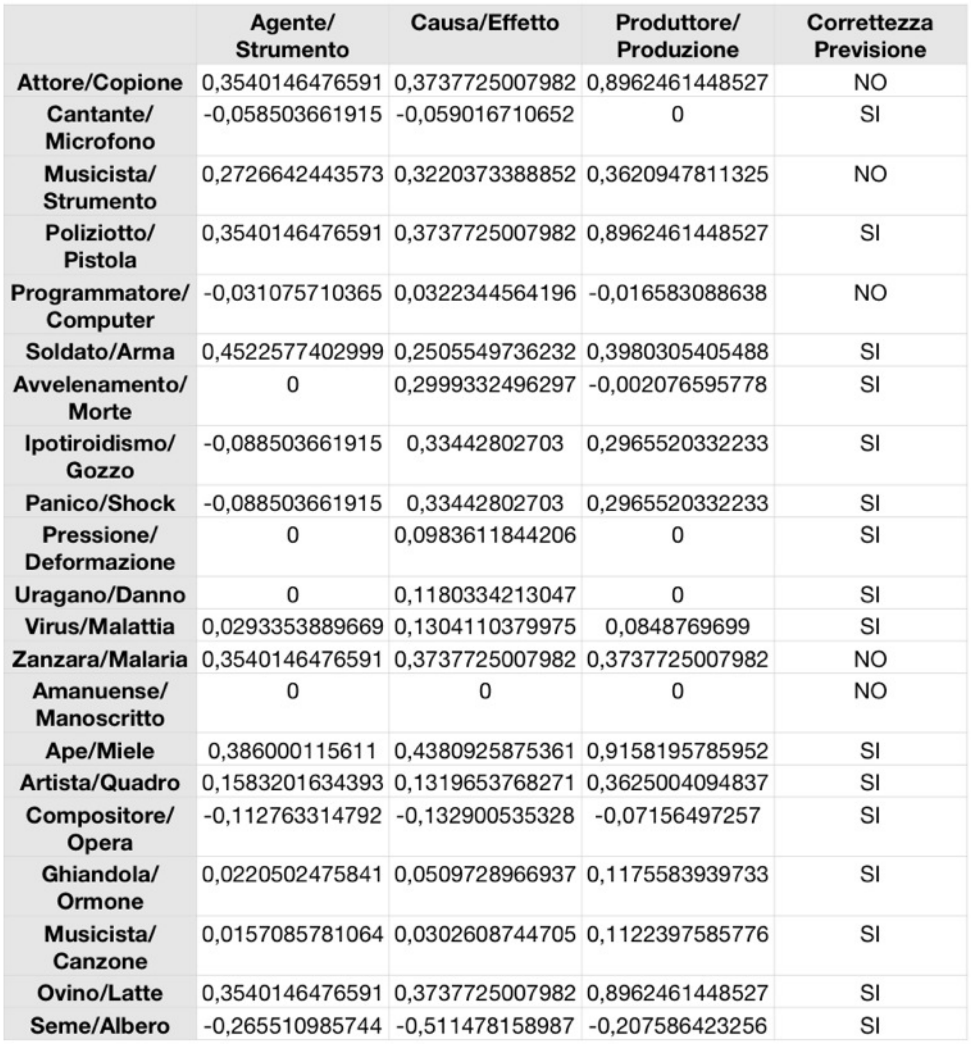
Bassista:Basso		Parrucchiere:Forbice		Programmatore:Computer	**14**
Muratore:Cazzuola		Poliziotto:Pistola	**2**	Elettricista:Energia	
Fotografo:Macchina		Magliaia:Ferro		Falegname:Legno	
Musicista:Strumento	**17**	Tessitore:Telaio	**2**	Soldato:Arma	**7**
Tassista:Macchina		Infermiera:Siringa		Pescatore:Amo	
Sarto:Ago		Giudice:Martelletto		Pescatore:Rete	**9**
Carpentiere:Martello		Panettiere:Pane		Panettiere:Pane	
Fabbro:Incudine		Dentista:Trapano	**1**	Ladro:Destrezza	
Attore:Copione	**2**	Criminale:Pistola	**1**	Medico:Stetoscopio	
Cantante:Microfono	**3**				

Figure 16: Manually selected seeds for the test set of Agency-Instrument relation, and their frequencies in the corpus with our patterns



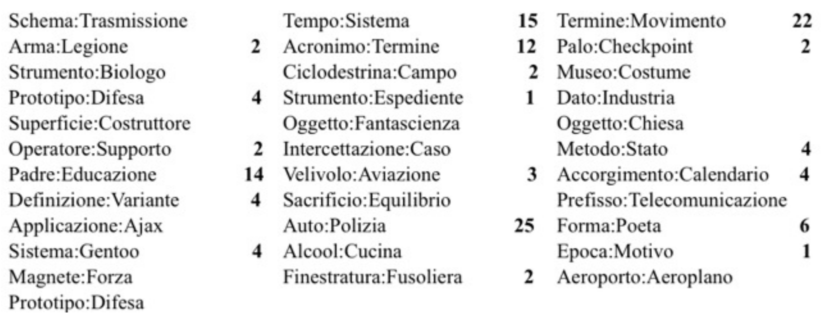
	Agente/ Strumento	Causa/Effetto	Produttore/ Produzione	Correttezza Previsione
Cantante/ Microfono	0,0724185898081	0,0417391935565	0,0094817615943	si
Musicista/ Strumento	0,0724185898081	0,0417391935565	0,0094817615943	si
Attore/Copione	0,0388366507146	0,0368105086916	0,0501727663386	no
Programmatore/ Computer	0	0,0173526402098	0,0591291721815	no
Soldato/Arma	0,2103658287177	0,015877516941	0,0108205370672	si
Poliziotto/ Pistola	0,0388366507146	0,0368105086916	0,0501727663386	no
Avvelenamento/ Morte	0	0,3524432008809	-0,015809990956	si
Ipotiroidismo/ Gozzo	0	0	0,2508638316928	no
Pressione/ Deformazione	0	0,1840525434581	0	si
Uragano/Danno	0	0,2208630521497	0	si
Virus/Malattia	0,0067994179947	0,1760245527754	0,009792721266	si
Zanzara/Malaria	0,0348366507146	0,0368105086916	0,0501727663386	no
Amanuense/ Manoscritto	0	0	0	no
Ape/Miele	0,0348366507146	0,0368105086916	0,0501727663386	si
Compositore/ Opera	-0,026817266224	0,1298767378462	0,0933389493772	no
Ghiandola/ Ormone	-0,022032632462	0,0155206732393	0,1322168484511	si
Musicista/ Canzone	0	0,0043083441661	0,206997804092	si
Ovino/Latte	0,0348366507146	0,0368105086916	0,0501727663386	si
Seme/Albero	0	0	0	no

Figure 17: List of results accuracy obtained using MI as associational measure and manually selected seeds



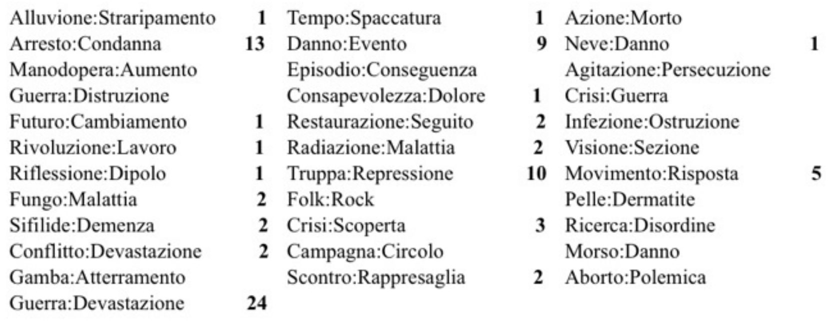
	Agente/ Strumento	Causa/Effetto	Produttore/ Produzione	Correttezza Previsione
Attore/Copione	0,3540146476591	0,3737725007982	0,8962461448527	NO
Cantante/ Microfono	-0,058503661915	-0,059016710652	0	SI
Musicista/ Strumento	0,2726642443573	0,3220373388852	0,3620947811325	NO
Poliziotto/ Pistola	0,3540146476591	0,3737725007982	0,8962461448527	SI
Programmatore/ Computer	-0,031075710365	0,0322344564196	-0,016583088638	NO
Soldato/Arma	0,4522577402999	0,2505549736232	0,3980305405488	SI
Avvelenamento/ Morte	0	0,2999332496297	-0,002076595778	SI
Ipotiroidismo/ Gozzo	-0,088503661915	0,33442802703	0,2965520332233	SI
Panico/Shock	-0,088503661915	0,33442802703	0,2965520332233	SI
Pressione/ Deformazione	0	0,0983611844206	0	SI
Uragano/Danno	0	0,1180334213047	0	SI
Virus/Malattia	0,0293353889669	0,1304110379975	0,0848769699	SI
Zanzara/Malaria	0,3540146476591	0,3737725007982	0,3737725007982	NO
Amanuense/ Manoscritto	0	0	0	NO
Ape/Miele	0,386000115611	0,4380925875361	0,9158195785952	SI
Artista/Quadro	0,1583201634393	0,1319653768271	0,3625004094837	SI
Compositore/ Opera	-0,112763314792	-0,132900535328	-0,07156497257	SI
Ghiandola/ Ormone	0,0220502475841	0,0509728966937	0,1175583939733	SI
Musicista/ Canzone	0,0157085781064	0,0302608744705	0,1122397585776	SI
Ovino/Latte	0,3540146476591	0,3737725007982	0,8962461448527	SI
Seme/Albero	-0,265510985744	-0,511478158987	-0,207586423256	SI

Figure 18: List of results accuracy obtained using LMI as associational measure and manually selected seeds



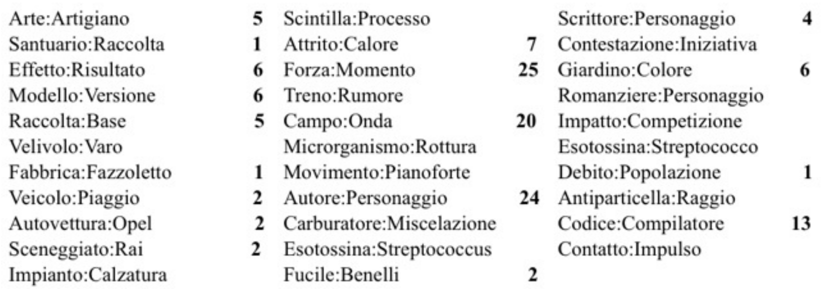
Schema:Trasmissione		Tempo:Sistema	**15**	Termine:Movimento	**22**
Arma:Legione	**2**	Acronimo:Termine	**12**	Palo:Checkpoint	**2**
Strumento:Biologo		Ciclodestrina:Campo	**2**	Museo:Costume	
Prototipo:Difesa	**4**	Strumento:Espediente	**1**	Dato:Industria	
Superficie:Costruttore		Oggetto:Fantascienza		Oggetto:Chiesa	
Operatore:Supporto	**2**	Intercettazione:Caso		Metodo:Stato	**4**
Padre:Educazione	**14**	Velivolo:Aviazione	**3**	Accorgimento:Calendario	**4**
Definizione:Variante	**4**	Sacrificio:Equilibrio		Prefisso:Telecomunicazione	
Applicazione:Ajax		Auto:Polizia	**25**	Forma:Poeta	**6**
Sistema:Gentoo	**4**	Alcool:Cucina		Epoca:Motivo	**1**
Magnete:Forza		Finestratura:Fusoliera	**2**	Aeroporto:Aeroplano	
Prototipo:Difesa					

FIGURE 19: Automatically selected seeds for the training set of Agency-Instrument relation, and their frequencies in the corpus with our patterns



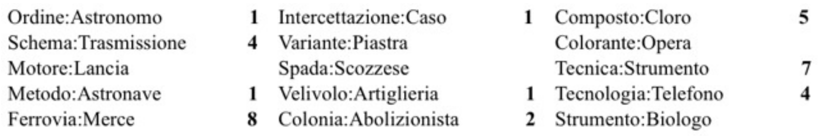
Alluvione:Straripamento	**1**	Tempo:Spaccatura	**1**	Azione:Morto	
Arresto:Condanna	**13**	Danno:Evento	**9**	Neve:Danno	**1**
Manodopera:Aumento		Episodio:Conseguenza		Agitazione:Persecuzione	
Guerra:Distruzione		Consapevolezza:Dolore	**1**	Crisi:Guerra	
Futuro:Cambiamento	**1**	Restaurazione:Seguito	**2**	Infezione:Ostruzione	
Rivoluzione:Lavoro	**1**	Radiazione:Malattia	**2**	Visione:Sezione	
Riflessione:Dipolo	**1**	Truppa:Repressione	**10**	Movimento:Risposta	**5**
Fungo:Malattia	**2**	Folk:Rock		Pelle:Dermatite	
Sifilide:Demenza	**2**	Crisi:Scoperta	**3**	Ricerca:Disordine	
Conflitto:Devastazione	**2**	Campagna:Circolo		Morso:Danno	
Gamba:Atterramento		Scontro:Rappresaglia	**2**	Aborto:Polemica	
Guerra:Devastazione	**24**				

FIGURE 20: Automatically selected seeds for the training set of Cause-Effect relation, and their frequencies in the corpus with our patterns



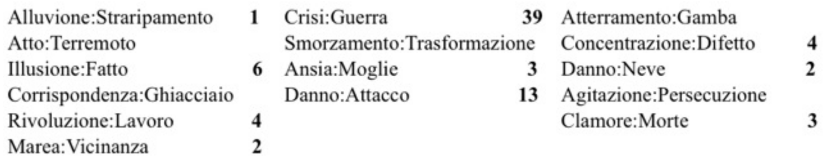
Arte:Artigiano	**5**	Scintilla:Processo		Scrittore:Personaggio	**4**
Santuario:Raccolta	**1**	Attrito:Calore	**7**	Contestazione:Iniziativa	
Effetto:Risultato	**6**	Forza:Momento	**25**	Giardino:Colore	**6**
Modello:Versione	**6**	Treno:Rumore		Romanziere:Personaggio	
Raccolta:Base	**5**	Campo:Onda	**20**	Impatto:Competizione	
Velivolo:Varo		Microrganismo:Rottura		Esotossina:Streptococco	
Fabbrica:Fazzoletto	**1**	Movimento:Pianoforte		Debito:Popolazione	**1**
Veicolo:Piaggio	**2**	Autore:Personaggio	**24**	Antiparticella:Raggio	
Autovettura:Opel	**2**	Carburatore:Miscelazione		Codice:Compilatore	**13**
Sceneggiato:Rai	**2**	Esotossina:Streptococcus		Contatto:Impulso	
Impianto:Calzatura		Fucile:Benelli	**2**		

FIGURE 21: Automatically selected seeds for the training set of Product-Producer relation, and their frequencies in the corpus with our patterns



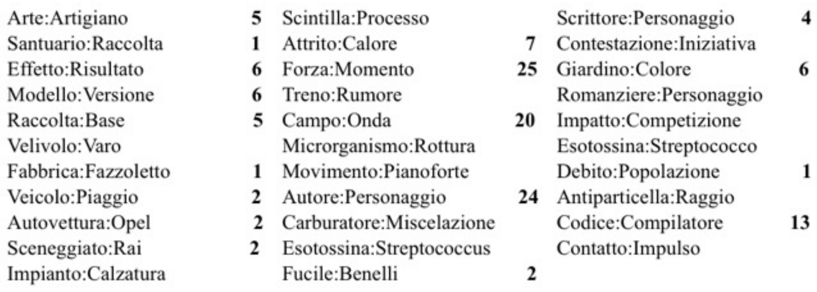
Ordine:Astronomo	**1**	Intercettazione:Caso	**1**	Composto:Cloro	**5**
Schema:Trasmissione	**4**	Variante:Piastra		Colorante:Opera	
Motore:Lancia		Spada:Scozzese		Tecnica:Strumento	**7**
Metodo:Astronave	**1**	Velivolo:Artiglieria	**1**	Tecnologia:Telefono	**4**
Ferrovia:Merce	**8**	Colonia:Abolizionista	**2**	Strumento:Biologo	

FIGURE 22: Automatically selected seeds for the test set of Agency-Instrument relation, and their frequencies in the corpus with our patterns



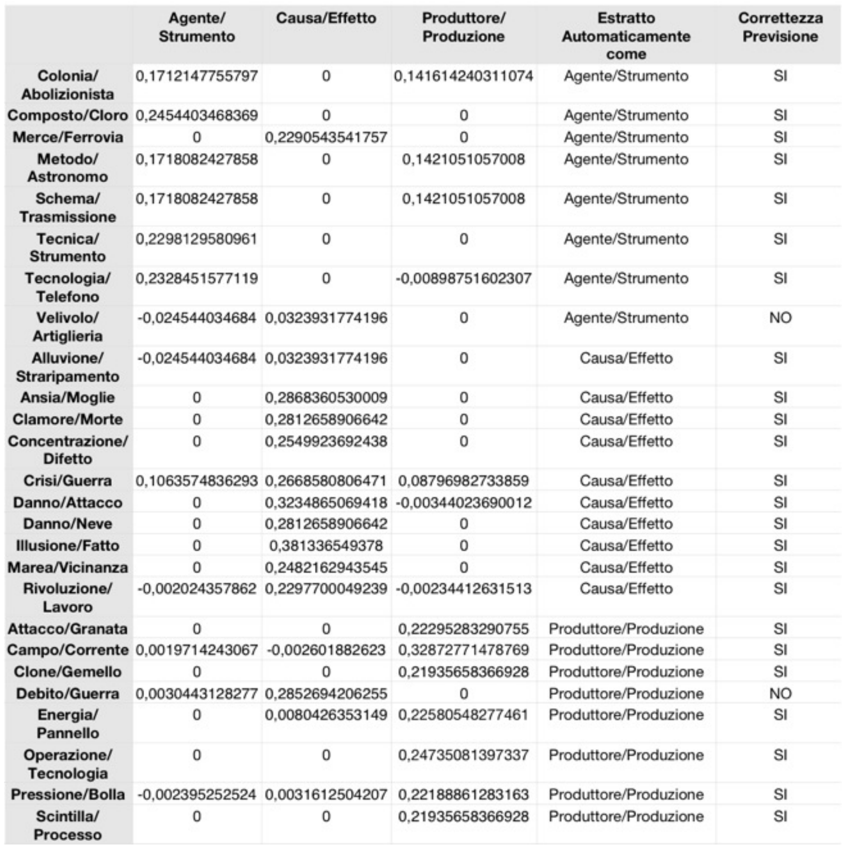
Alluvione:Straripamento	**1**	Crisi:Guerra	**39**	Atterramento:Gamba	
Atto:Terremoto		Smorzamento:Trasformazione		Concentrazione:Difetto	**4**
Illusione:Fatto	**6**	Ansia:Moglie	**3**	Danno:Neve	**2**
Corrispondenza:Ghiacciaio		Danno:Attacco	**13**	Agitazione:Persecuzione	
Rivoluzione:Lavoro	**4**			Clamore:Morte	**3**
Marea:Vicinanza	**2**				

FIGURE 23: Automatically selected seeds for the test set of Cause-Effect relation, and their frequencies in the corpus with our patterns



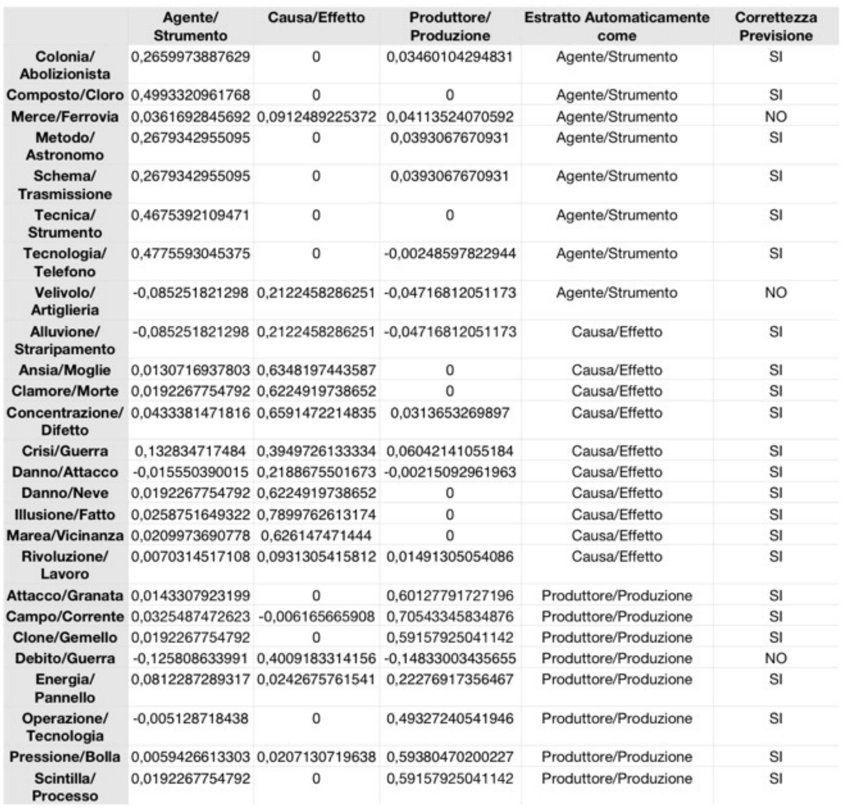
Arte:Artigiano	**5**	Scintilla:Processo		Scrittore:Personaggio	**4**
Santuario:Raccolta	**1**	Attrito:Calore	**7**	Contestazione:Iniziativa	
Effetto:Risultato	**6**	Forza:Momento	**25**	Giardino:Colore	**6**
Modello:Versione	**6**	Treno:Rumore		Romanziere:Personaggio	
Raccolta:Base	**5**	Campo:Onda	**20**	Impatto:Competizione	
Velivolo:Varo		Microrganismo:Rottura		Esotossina:Streptococco	
Fabbrica:Fazzoletto	**1**	Movimento:Pianoforte		Debito:Popolazione	**1**
Veicolo:Piaggio	**2**	Autore:Personaggio	**24**	Antiparticella:Raggio	
Autovettura:Opel	**2**	Carburatore:Miscelazione		Codice:Compilatore	**13**
Sceneggiato:Rai	**2**	Esotossina:Streptococcus		Contatto:Impulso	
Impianto:Calzatura		Fucile:Benelli	**2**		

FIGURE 24: Automatically selected seeds for the test set of Product-Producer relation, and their frequencies in the corpus with our patterns



	Agente/ Strumento	Causa/Effetto	Produttore/ Produzione	Estratto Automaticamente come	Correttezza Previsione
Colonia/ Abolizionista	0,1712147755797	0	0,141614240311074	Agente/Strumento	SI
Composto/Cloro	0,2454403468369	0	0	Agente/Strumento	SI
Merce/Ferrovia	0	0,2290543541757	0	Agente/Strumento	SI
Metodo/ Astronomo	0,1718082427858	0	0,1421051057008	Agente/Strumento	SI
Schema/ Trasmissione	0,1718082427858	0	0,1421051057008	Agente/Strumento	SI
Tecnica/ Strumento	0,2298129580961	0	0	Agente/Strumento	SI
Tecnologia/ Telefono	0,2328451577119	0	-0,00898751602307	Agente/Strumento	SI
Velivolo/ Artiglieria	-0,024544034684	0,0323931774196	0	Agente/Strumento	NO
Alluvione/ Straripamento	-0,024544034684	0,0323931774196	0	Causa/Effetto	SI
Ansia/Moglie	0	0,2868360530009	0	Causa/Effetto	SI
Clamore/Morte	0	0,2812658906642	0	Causa/Effetto	SI
Concentrazione/ Difetto	0	0,2549923692438	0	Causa/Effetto	SI
Crisi/Guerra	0,1063574836293	0,2668580806471	0,08796982733859	Causa/Effetto	SI
Danno/Attacco	0	0,3234865069418	-0,00344023690012	Causa/Effetto	SI
Danno/Neve	0	0,2812658906642	0	Causa/Effetto	SI
Illusione/Fatto	0	0,381336549378	0	Causa/Effetto	SI
Marea/Vicinanza	0	0,2482162943545	0	Causa/Effetto	SI
Rivoluzione/ Lavoro	-0,002024357862	0,2297700049239	-0,00234412631513	Causa/Effetto	SI
Attacco/Granata	0	0	0,22295283290755	Produttore/Produzione	SI
Campo/Corrente	0,0019714243067	-0,002601882623	0,32872771478769	Produttore/Produzione	SI
Clone/Gemello	0	0	0,21935658366928	Produttore/Produzione	SI
Debito/Guerra	0,0030443128277	0,2852694206255	0	Produttore/Produzione	NO
Energia/ Pannello	0	0,0080426353149	0,22580548277461	Produttore/Produzione	SI
Operazione/ Tecnologia	0	0	0,24735081397337	Produttore/Produzione	SI
Pressione/Bolla	-0,002395252524	0,0031612504207	0,22188861283163	Produttore/Produzione	SI
Scintilla/ Processo	0	0	0,21935658366928	Produttore/Produzione	SI

FIGURE 25: List of results accuracy obtained using MI as associational measure and automatically selected seeds



	Agente/ Strumento	Causa/Effetto	Produttore/ Produzione	Estratto Automaticamente come	Correttezza Previsione
Colonia/ Abolizionista	0,2659973887629	0	0,03460104294831	Agente/Strumento	SI
Composto/Cloro	0,4993320961768	0	0	Agente/Strumento	SI
Merce/Ferrovia	0,0361692845692	0,0912489225372	0,04113524070592	Agente/Strumento	NO
Metodo/ Astronomo	0,2679342955095	0	0,0393067670931	Agente/Strumento	SI
Schema/ Trasmissione	0,2679342955095	0	0,0393067670931	Agente/Strumento	SI
Tecnica/ Strumento	0,4675392109471	0	0	Agente/Strumento	SI
Tecnologia/ Telefono	0,4775593045375	0	-0,00248597822944	Agente/Strumento	SI
Velivolo/ Artiglieria	-0,085251821298	0,2122458286251	-0,04716812051173	Agente/Strumento	NO
Alluvione/ Straripamento	-0,085251821298	0,2122458286251	-0,04716812051173	Causa/Effetto	SI
Ansia/Moglie	0,0130716937803	0,6348197443587	0	Causa/Effetto	SI
Clamore/Morte	0,0192267754792	0,6224919738652	0	Causa/Effetto	SI
Concentrazione/ Difetto	0,0433381471816	0,6591472214835	0,0313653269897	Causa/Effetto	SI
Crisi/Guerra	0,132834717484	0,3949726133334	0,06042141055184	Causa/Effetto	SI
Danno/Attacco	-0,015550390015	0,2188675501673	-0,00215092961963	Causa/Effetto	SI
Danno/Neve	0,0192267754792	0,6224919738652	0	Causa/Effetto	SI
Illusione/Fatto	0,0258751649322	0,7899762613174	0	Causa/Effetto	SI
Marea/Vicinanza	0,0209973690778	0,626147471444	0	Causa/Effetto	SI
Rivoluzione/ Lavoro	0,0070314517108	0,0931305415812	0,01491305054086	Causa/Effetto	SI
Attacco/Granata	0,0143307923199	0	0,60127791727196	Produttore/Produzione	SI
Campo/Corrente	0,0325487472623	-0,006165665908	0,70543345834876	Produttore/Produzione	SI
Clone/Gemello	0,0192267754792	0	0,59157925041142	Produttore/Produzione	SI
Debito/Guerra	-0,125808633991	0,4009183314156	-0,14833003435655	Produttore/Produzione	NO
Energia/ Pannello	0,0812287289317	0,0242675761541	0,22276917356467	Produttore/Produzione	SI
Operazione/ Tecnologia	-0,005128718438	0	0,49327240541946	Produttore/Produzione	SI
Pressione/Bolla	0,0059426613303	0,0207130719638	0,59380470200227	Produttore/Produzione	SI
Scintilla/ Processo	0,0192267754792	0	0,59157925041142	Produttore/Produzione	SI

Figure 26: List of results accuracy obtained using LMI as associational measure and automatically selected seeds

Perl Scripts

```
#!/usr/bin/perl
use strict;
#use strict is an instruction which does two things. It makes you declare all your
    variables (\textquotedblleft strict vars\textquotedblright{}), and it makes it
    harder for Perl to mistake your intentions when you are using subs (\
    textquotedblleft strict subs\textquotedblright{}).
use warnings;
#turning on warnings will make Perl yelp and complain at a huge variety of things
    that are almost always sources of bugs in your programs.
open my $listaParole,"Input/Coppia2.txt" or die;
#Opens the text file containing our couple list and put inside the variable $listaParole
        my %hash;
        #Create a new hash variable
        while (my $line=<$listaParole>) {
        #We have to go throughout the whole file, to account all the words in the list
        chomp $line;
        my ($word1, $word2) = split /:/, $line;
        $hash{$word1} = $word2;
        #We divided the words using the colon as a discriminant and put the word left
    to the colon into a variable called $word1, and the word right to the colon into a
    variable called $word2
        }
open my $testo, "<Wiki_Pulito/Materiale/L/Terzo.txt";
# Opens up the text file that needs to be examined
open my $lista_relazioni, ">Wiki_Pulito/Prova/PatternNormali/Risultati_A3.txt";
```

```
# Opens the output file
        my %arrayris;
        my $indice=0;
        while (my $text=<$testo>){
        #Searches through all the text file
                for my $key (keys %hash){
                #Searches for every couple key-value

                 my $value = $hash{$key};
        while ($text =~ /(($key\/$key\/S\s{0,})([A-Za-z\u00C0 \u017F\u1e00-\
     u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/(E|C)\s{0,}){0,}([A-
     Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\
     u1ef9]{0,}\/S\s{0,}){0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-
     z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/(E|C)\s{0,}){0,}([A-Za-z1-9\u00C0-\
     u017F\u1e00-\u1ef9]{0,}[(),.;:]{0,}\/[A-Za-z1-9\u00C0-\u017F\u1e00-\
     u1ef9]{0,}\/(A|N|RIFS|RIMS|RIMP|RIFP)\s{0,}){0,}([A-Za-z\u00C0-\u017F\
     u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/(E|C)\s{0,})
     {0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\
     u1e00-\u1ef9]{0,}\/V\s{0,}){0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9
     ]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/(E|C)\s{0,}){0,}([A-Za-
     z1-9\u00C0-\u017F\u1e00-\u1ef9]{0,}[(),.;:]{0,}\/[A-Za-z1-9\u00C0-\
     u017F\u1e00-\u1ef9]{0,}\/(A|N|RIFS|RIMS|RIMP|RIFP)\s{0,}){0,}([A-Za-z\
     u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9
     ]{0,}\/S\s{0,}){0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\
     u00C0-\u017F\u1e00-\u1ef9]{0,}\/(E|C)\s{0,}){0,}($value\/$value\/S))/ig){

{

#This is the Regular Expression that searches in our corpus for every occurrences of
     the first word, followed by 6 words top (including an optional punctuation sign
     but not points or brackets) and by the second word
        my $arrkey=$key."-".$value;
        $arrayris{$arrkey}=$1;
        $indice++;
          }
        }
        }

while ( my ($k,$v) = each %arrayris ) {
    print $lista_relazioni "$v\n";
}
#This last loop allowed to print every extracted sentences inside our output text file
close $testo;
close $lista_relazioni;
```

Script that extracts the patterns

```
#!/usr/bin/perl
use strict;
use warnings;
open( INPUT, ”<Input/PatternLocazione.txt”) or die ”Can’t open Pattern.txt”;
open( LISTAPAROLE,”<Input/relazioneLocazione.txt”) or die ”Can’t open
    Coppie_Parole.txt”;
#Opens the text file containing our pattern list and put inside the file INPUT. It also
    opens the file containing our couple list and put inside the file LISTAPAROLE
my %hash;
        while (<LISTAPAROLE>) {
          chomp;
         my ($word1, $word2) = split /:/, $_;
                $hash{$word1} = $word2;
        #We divided the words using the colon as a discriminant and put the word left
    to the    #colon into a variable called $word1, and the word right to the colon
    into a variable   #called $word2
        }
close LISTAPAROLE;
open( CONTEGGIO, ”>Output/ConteggioA.txt”) or die ”Can’t open Conteggio.txt
    ”;
# Opens the output file
my $conto=0;
my %arrayris;
        while (my $text = <INPUT>){
        #Searches through all the text file
        for my $key (keys %hash){
        my $value = $hash{$key};
        #Searches for every couple key-value
        if ($text =~/$key\/$key\/S\s{0,1}(\.\*)\s{0,1},}[A-Za-z\u00C0-\u017F\
    u1e00-\u1ef9]{0,}\/considerare\/V\s{0,}[A-Za-z\u00C0-\u017F\u1e00-\
    u1ef9]{0,}\/in\/E\s{0,1}(\.\*)\s{0,1}$value\/$value\/S/is){
        #Regex which is used for counting the patterns made with the key, followed
        #by the pattern we want to analyze, followed then by the value
```

```
                $arrayris{ join '-', $key, $value }++;
                # increase for each key/value pair individually
                }
  }
}
while ( my ($k,$v) = each %arrayris ) {
 print CONTEGGIO "($k) => $v\n";
#This last loop allowed to print every extracted sentences inside our output text file
}
close INPUT;
close CONTEGGIO;
```

Script that counts the occurrences of every pattern with every couple of seeds

```
#!/usr/bin/perl
use strict;
use warnings;
open( INPUT, "<Input/Tutto.txt") or die "Can't open Pattern2.txt";
#Opens the text file containing our corpus  and put inside the file INPUT.
open( CONTEGGIO, ">Output/ConteggioA.txt") or die "Can't open Conteggio.txt
  ";
#Opens the output file
my $conto=0;
#The counting variable is initialized, and its value is set to 0

    while (my $text = <INPUT>){
    if ($text=~/.*\/considerare\/V\s{0,}[A-Za-z\u00C0-\u017F\u1e00-\u1ef9
  ]{0,} \/in\/E\.*/is){
    $conto++;
    #Scans all the text file for the given pattern (the regex) and adds 1 to the
  counting  #variable every time the value of the regex is found in the file
 }
}
 print CONTEGGIO "$conto";
#Prints the result into the output file
close INPUT;
close CONTEGGIO;
```

Script that counts all the occurrences of every extracted pattern in the corpus

```
#!/usr/bin/perl
use strict;
use warnings;
use PDL;  #implements the PDL module
        open my $meronimia,"<arrayMero.txt";
        open my $locazione,"<arrayLoca.txt";
        open my $coppia,"<arrayAcquedottoArcata.txt"
        open my $output, ">AlfabetoSegnoLMI.txt";

                        my $meronimiaLMI= piddle $meronimia;
                        my $locazioneLMI=piddle $locazione;
                        my $relazione= piddle $coppia;

                        my $n1 = norm $meronimiaLMI;
                        my $n2 = norm $locazioneLMI;
                            my $n4 = norm $relazione;



                            my $cos1 = inner( $n1, $n4 ); # inner product
                            my $cos2 = inner( $n3, $n4 );

                            my $v1=$cos1->sclr();# converts PDL object to Perl scalar
                            my $v2=$cos2->sclr();

print $output "Meronimia=".$v1."\nLocazione=".$v2;
```

Script that computes the cosine between the Vectors representing the relations and the vector representing the seeds

```perl
#!/usr/bin/perl
use strict;
use warnings;

open( INPUT, "<Wiki_Pulito/Materiale/L/Terzo.txt") or die "Con't find file";
open( OUTPUT, ">Wiki_Pulito/Risultati/Lista1.txt") or die "Con't find file";

my $conto=0;
my %arrayris;
while (my $text = <INPUT>){
  if ($text =~/([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/S)\s{0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A|E|P|PU|RIFS|RIMS|RDFP|PQNN|SPNN|B|RDMP|RDFS|RDFP|V|S|N|C|RDMS|SP]\s{0,}){0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/creare\/V\s{0,})([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A|E|P|PU|RIFS|RIMS|RDFP|PQNN|SPNN|B|RDMP|RDFS|RDFP|V|S|N|C|RDMS|SP]\s{0,}){0,}([A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/[A-Za-z\u00C0-\u017F\u1e00-\u1ef9]{0,}\/S)/is){

      my $arrkey=$conto;
            $arrayris{$arrkey}=$1."-".$5;
            $conto++;


   }
 }

while ( my ($k,$v) = each %arrayris ) {
 print OUTPUT "($k) => $v\n";
}
```

Script that extracts automatically the seeds to be used for our experiment

Bibliography

[1] D. Jurafsky and J.H. Martin. *Speech and Language Processing - An introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, 2002.

[2] P. Buitelaar et al. (Eds.). *Ontology Learning from Text.* IOS Press, 2005.

[3] D.D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, 1992.

[4] C.L.A. Clarke, G.V Cormack, and C.R. Palmer. An overview of multitext. *ACM SIGIR Forum*, 1998.

[5] Sharon Caraballo. Automatic acquisition of a hypernym-labeled noun hierarchy from text. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, June 1999.

[6] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. *Proceedings of the 14th conference on Computational linguistics*, 1992.

[7] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, pages 83–135, 2006.

[8] P. Pantel and D. Ravichandran. Automatical ly labeling semantic classes. *Proceedings of HLT/NAACL*, 2004.

[9] Patrick Pantel and Ravichandran Deepak. Automatically labeling semantic classes. *HLT-NAACL 2004: Workshop on Computational Lexical Semantic*, May 2004.

[10] E. Riloff. Automatically labeling semantic classesly generating extraction patterns from untagged text. *AAAI/IAAI*, 2:1044–1049, 1996.

[11] E. Riloff, R. Jones, A. McCallum, and K. Nigam. Bootstrapping for text learning tasks. *JCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications Finite-State Transducers for Semi-Structured Text Mining (AAAI-99)*, 1999.

[12] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. *Proceedings of Conference on Computational Linguistics/Association for Computational Linguistics (COLING/ACL-06)*, pages 113–120, 2006.

[13] V. Nastase and S. Szpakowicz. Exploring noun-modifier sematic relations. *5th International Workshop on Computational Semantics*, 2003.

[14] P.D. Turney. Expressing implicit semantic relations without supervision. *Annual Meeting-Association for computational linguistics*, 2006.

[15] S. Brin. Extracting patterns and relations from the world wide web. *Lecture Notes in Computer Science*, pages 172–183, 1999.

[16] M Berland and E Charniak. Finding parts in very large corpora. *Proceedings of the 37th annual meeting of the ACL*, 1999.

[17] E.Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[18] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 1988.

[19] D. Ravichandran and E. Hovy. Learning surface patterns for a question answering system. *Proceedings of the ACL Conference*, 2002.

[20] Jane Morris and Graeme Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17 (1), March 1991.

[21] D.A. Cruse. Lexical semantics. *Cambridge Textbook in Linguistics*, page 159, 1989.

[22] P. D. Turney. Measuring semantic similarity by latent relational analysis. *International joint conference on artificial intelligence*, 2005.

[23] D. Moldovan, A. Badulescu, M. Tatu, D. Antohe, and R. Girju. Models for the semantic classification of noun phrases. *HLT/NAACL*, 2004.

[24] Dan Moldovan, Adriana Badulescu, Roxana Girju, Marta Tatu, and Daniel Antohe. Models for the semantic classification of noun phrases. *HLT-NAACL 2004: Workshop on Computational Lexical Semantic*, May 2004.

[25] Marti A. Hearst. Noun homograph disambiguation using local context in large text corpora. *Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New OED and Text Research*, October 1991.

[26] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recognition Letters*, pages 105–111, 1995.

[27] R. Girju, D. Moldovan, M. Tatu, and D.Antohe. On the semantics of noun compounds. *Computer Speech Language*, 2005.

[28] D. Roth and W. Yih. Probabilistic reasoning for entity relation recognition. *Proceedings of COLING*, 2002.

[29] J.Pearl. *Probabilistic reasoning in intelligent systems.* Morgan Kaufmann Publications, 1988.

[30] E.Terra and C.L.A. Clarke. Scoring missing terms in information retrieval task. *Proceedings of the thirteenth ACM international conference*, 2004.

[31] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

[32] J. Justeson. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, pages 9–27, 1995.

[33] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. *Arxiv preprint cs.LG/0111003*, 2001.

[34] M.Collins and Y. Singer. Unsupervised models for named entity classification. *Proceedings of the joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[35] G.H. Golub and C.F. Van Loan. *Matrix Computations*, 1996.

[36] T. M. Cover and J. A. Thomas. *Elements of Information Theory*, 1991.

[37] T.K. Landauer and S.T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 1997.