ENRICO MENSA

DESIGN AND IMPLEMENTATION OF A METHODOLOGY FOR THE ALIGNMENT OF SEMANTIC RESOURCES AND THE AUTOMATIC POPULATION OF CONCEPTUAL SPACES

Università degli Studi di Torino

Polo delle Scienze della Natura

Corso di Laurea Magistrale in Informatica

Master's Degree Thesis

DESIGN AND IMPLEMENTATION OF A METHODOLOGY FOR THE ALIGNMENT OF SEMANTIC RESOURCES AND THE AUTOMATIC POPULATION OF CONCEPTUAL SPACES

ENRICO MENSA

ACADEMIC YEAR 2014/2015

Enrico Mensa: *Design and implementation of a methodology for the alignment of semantic resources and the automatic population of conceptual spaces*

SUPERVISORS:

Daniele Radicioni

Antonio Lieto

LOCATION:

Torino

A chiunque mi abbia mai insegnato qualcosa

ABSTRACT

In this thesis I introduce the TTCS[1] system, that exploits a resource-driven approach relying on BabelNet, NASARI and ConceptNet. TTCS takes in input a term and its context of usage and produces as output a novel type of vector-based semantic representation, where conceptual information is encoded through the conceptual spaces (a framework for common-sense knowledge representation and reasoning). The system has been evaluated through a twofold experimentation aimed at assessing i) the quality of the extracted common-sense conceptual information w.r.t. human judgments; ii) the usefulness of the obtained representations in a wider context. To serve this purpose the TTCS results have been employed in DUAL-PECCS[2], a previously existent categorization system based on both ontologies and conceptual spaces. In both cases the results are encouraging and provide precious insights to make substantial improvements.

1 Terms To Conceptual Spaces.

2 DUAL-Prototypes and Exemplars-based Conceptual Categorization System.

ACKNOWLEDGMENTS

Un grande ringraziamento al Prof. Daniele Radicioni e al Dott. Antonio Lieto per avermi guidato e sostenuto durante lo sviluppo del sistema e la scrittura di questa tesi. Lavorare con voi è stato un onore ed un piacere.

Ringrazio profondamente il coro *VoxViva* e la musica in generale, non per qualcosa di specifico, ma perché sono inevitabilmente parte di ogni cosa che faccio.

Ringrazio la mia famiglia che mi ha permesso di essere qui.

Ringrazio le persone che mi stanno vicino passo dopo passo e senza le quali fare il passo successivo sarebbe inutile.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

TTCS Terms To Conceptual Spaces.

DUAL-PECCS DUAL-Prototypes and Exemplars-based Conceptual Categorization System.

Part I

BASICS

1

INTRODUCTION

1.1 MOTIVATIONS

The development of reliable knowledge sources to use in different scenarios (such as automatic reasoning, recognition, categorization, *etc.*) represents an active area of research in the AI community. In this thesis I face the problem of automatically generating a conceptual space representation, starting from text and passing through a pipeline involving the integrated use of different linguistic resources: BabelNet (Navigli and Ponzetto, 2012), NASARI (Camacho-Collados, Pilehvar, and Navigli, 2015a) and ConceptNet (Speer and Havasi, 2012). The resulting representation enjoys the interesting property of being anchored to such resources, thus providing a uniform interface between the linguistic and the conceptual level.

Conceptual spaces can be thought of as a particular class of vector representations where knowledge is represented as a set of quality dimensions, and where a geometrical structure is associated to each quality dimension. In this setting, concepts correspond to convex regions, and regions with different geometrical properties correspond to different sorts of concepts (Gärdenfors, 2014). Here common-sense conceptual representation and reasoning have a natural geometrical interpretation, since prototypes (the most relevant representatives of a category from a cognitive point of view, see (Rosch, 1975)) correspond to the geometrical centre of a convex region, the centroid. Furthermore, exemplars-based representations can be mapped onto points in a multidimensional space, and their similarity can be computed as the intervening distance between each two points, based on

some suitable metrics (such as Euclidean and Manhattan distance, or standard cosine similarity).

It is widely accepted that knowledge acquisition is a severe and long-standing bottleneck in many applications (Lenat, Prakash, and Shepherd, 1985). However, while as a common reference it is possible to ingest existing formal ontologies such as OpenCyc or domain ontologies, unfortunately no broad coverage resources exist containing common-sense knowledge compliant to the conceptual spaces framework. Also, wide-coverage semantic resources such as DBPedia and ConceptNet, in fact, mostly fail to represent the sort of common-sense information based on prototypical and default information and usually required to perform forms of plausible reasoning.

Focus of this work is precisely how to extract common-sense information, suitable to be encoded through conceptual spaces. Automatically annotating information encoded through conceptual spaces is a process that goes all throughout from a *term* and a *context* of usage of the given term to individuating the underlying *concept*, to individuating a salient description for the concept at hand along the dimensions describing it within a conceptual space.

This work has the following main strengths: the TTCS can be used to build a broad-coverage, basically domain-independent knowledge base implementing the geometrical representational tenets of conceptual spaces; additionally, the TTCS can be used to integrate different recent, state-of-the-art lexical and semantic resources to operate them in a novel fashion (so to exploit both BabelNet concepts and ConceptNet relations). Finally, the TTCS builds on the relations harvested from ConceptNet to design a procedure to fill the appropriate dimensions in the conceptual space representation thus producing Lexicalized conceptual spaces (i.e., conceptual spaces whose representation is fully endowed with BabelSynsetIds). In other words, the TTCS system is able to perform two different kinds of operations: the first one consists in the extraction of semantics regarding a certain input term (*semantic extraction*), the second one is the the task of assigning the

correct semantic values to a certain set of quality dimension (*semantic matching*). In turn, the semantic extraction phase is composed by a *concept identification* step and a *extraction* step. The former detects the correct semantic identifier for an input term while the latter, exploiting the previously obtained semantic ID, extracts new semantic information regarding the input.

On the whole, the TTCS is part of a broader effort, aimed at collecting common-sense knowledge to overcome some of the limitations proper to most symbolic-oriented resources (like Cyc) in handling forms of non-monotonic reasoning (Frixione and Lieto, 2012).

This work is structured as follows: this Chapter introduces some theoretical notion about conceptual spaces required in order to understand the system and its processes. The second Chapter illustrates the state of art: it first surveys the related literature, elaborates on existing approaches and about the differences with current proposal (Section 2.1). Then I introduce the resources underlying the TTCS (Section 2.2). Chapter 3 explains in full detail the strategy implemented by the TTCS system and report some elements of the build of a resource where information is encoded based on the conceptual spaces framework. The experimentation to assess the obtained results is described in Chapter 4, where it is shown how the resource obtained through the TTCS has been evaluated through an on-line questionnaire, and used as the knowledge base used by a conceptual categorization system. The final remarks on future work conclude the thesis (Chapter 5).

1.2 CONCEPTUAL SPACES IN KNOWLEDGE REPRESENTATION

One of the main problems about the classical representation of knowledge stems from the fact that ontologies do not typically allow for any kind of prototypical information nor non-monotonic reasoning. Conversely, prototypical representation and typicality-based reason-

ing seem to be deeply employed in human cognition (Berlin and Kay, 1969; Rosch, 1975).

Conceptual spaces were originally proposed by Gärdenfors (Gärdenfors, 2004a; Gärdenfors, 2004b); they aim at providing a solid framework for non-monotonic reasoning and prototype representation. Currently, Cognitive Science approaches knowledge representation in basically two ways: symbolic representations and non-symbolic representations (connectionism). The former asserts that Turing machines should be used to model any cognitive system; the latter, conversely, states that the connection between artificial neurons (i.e., neural networks) should be employed.

According to their inventor, conceptual spaces can be placed in a third category, as they are a *geometrical representation*. This definition fits perfectly because a conceptual space is nothing but a multi-dimensional space in which each axis symbolizes a certain quality of an exemplar, that is, a point. Each coordinate of the space is then called *quality dimension*, so, a N-dimension conceptual space will have N quality dimensions, and each exemplar of the space will be a vector constituted by N values, one for each quality dimension. Relatedly, conceptual spaces strongly rely on the notion of *similarity* between exemplars, allowing the understanding and representation of a variegated set of cognitive phenomena.

1.2.1 *Quality Dimensions*

The definition of the set of quality dimensions for a certain conceptual space is the core task regarding this kind of framework. Indeed, a quality dimension is supposed to represent a "quality" of an object. Clearly the decision about which set of quality dimensions to employ must take in account the domain being represented. However, there is a feature that every proper quality dimension must have: each quality dimension is supposed to be endowed with certain geometrical

structure. Please note that such assumption does not imply that every quality dimension must have a continuous numerical domain; it can possibly be based on a topological or ordered structure.

The `weight` dimension is a first example of quality dimension: it is one-dimensional and bottom-limited to `0`. Another example of quality dimension could be `pitch`, a continuous one-dimensional structure going from low to high tones. An example of discrete dimension could be the `kinship` relation. In this kind of setting the distance between two values, let us say `mother` and `grandmother`, could be computed as the length of the path connecting them in the topological structure representing the kinship relationships: in our example, the distance would be 1.

1.2.1.1 *On the Origin of Quality Dimensions*

The origin of quality dimensions is a controversial topic, however different kind of dimensions can be pinpointed:

1. Innate or early developed quality dimensions: this kind of quality dimensions seem to be hardwired in our nervous system. Between these dimensions we can find `taste`, `color`, `smell`. According to Quine (Quine, 1969) innate quality dimensions are required to make learning possible.

2. When growing up, children experience the word and are able to combine the innate quality dimension in more complex representations, such as `volume` and `height`.

3. Other quality dimensions seem to be *culturally* dependent like for example `time`. In some cultures `time` is perceived like a circular phenomenon, while in other ones it is hardly considered as a dimension.

4. Finally, some quality dimensions have been introduced by *science*. For example the dimension of `mass`.

1.2.1.2 *Integral and Separable Dimensions*

Two important definitions that we need to introduce are the notions of *integral* and *separable* dimensions. Such notions are taken from cognitive psychology (Garner, 1974; Algom, 1992; Ashby, 2014) and are central to understand how quality dimensions can adequately fulfill their role i.e., to represent a certain *domain*.

Each quality dimension of a set $d_1, \ldots, d_n$ is defined to be *integral* if assigning a value to one of them necessarily requires the assignment of a value for all the other ones. For example, an object cannot have a value for `hue` without having a value for `brightness`. Dimensions that are not integral are said to be *separable*, as for example the `size` and the `hue` dimensions. Using this distinction the notion of a domain can now be defined as a set of integral dimensions that are separable from all the other dimensions. Ultimately, domains form the framework employed to assign properties to objects and to specify relations (via similarity) between them.

Thanks to the notions previously introduced we are ready to see conceptual spaces in a more formal way.

1.2.2 *Formal Definitions*

A conceptual space S is defined by a set of quality dimensions $d_1, \ldots, d_n$. A point in S is represented by a vector $v = \langle d_1, \ldots, d_n \rangle$ with one index for each dimension. Each quality dimension domain is defined upon a topological or metrical structure.

1.2.2.1 *Properties and Concepts*

Studying how a knowledge framework defines properties is an important step for its understanding. In conceptual spaces property are defined as follows:

Definition 1.1 (Criterion P)**.** A *property* is a convex[1] region in some domain.

The reason for defining a property as a convex region is the fact that if two objects examples of a property C are respectively located at v_1 and v_2 for a set of quality dimensions defining a domain (i.e., integral dimensions separable from all the other ones), then any other object located between them will also be an example of C. Note that the Criterion P presumes that the notion of betweenness[2] is meaningful for the relevant quality dimensions.

Properties are a particular case of *concepts*. The fundamental distinction between the two it is the fact that properties are convex regions over a *single* domain, while a generic concept may be defined over *several* domains. Note that this kind of distinction is the same that we can make between adjectives and nouns. Adjectives like "red" and "tall" normally refer to a single domain and in fact they are properties; on the other hand, nouns like "house" and "cat" normally refer to a whole set of domains and thus represent concepts.

When we deal with concepts in order to compute their distances is useful to employ some kind of weighting of the different domains involved. The relative weights assigned to the domains depend on the *context* where concepts are used. This shrewdness has no counterpart in ontologies, and takes into account the *prominence* of the different domains involved. Such prominence can be inferred from the use of the concept that we are dealing with. For example if you are eating an orange its taste will be the prominent domain; on the other hand, if the orange is used as model for a still life painting its color or its shape will be weighted as prominent.

1 In a Euclidean space a convex region R is defined as a region such that, taken any couple of points $a \in R$ and $b \in R$, every point of the line segment connecting them is still in R.

2 Betwenness is the state of an object of being between two others in an ordered mathematical set (e.g., inside $\mathbb{N}$ 4 is between 3 and 5).

Another interesting feature of conceptual spaces is the possibility to take in account the *correlation* between regions from different domains associated to the concept. Thanks to this further variable, objects can be considered not only as a bunch of properties, but like a true semantic unit, provided with its own specificity. For example, the sour taste of an orange is strongly correlated with its PH value, while its shape and its color are poorly correlated.

Thanks to the introduction of the notions of correlation and prominence, we can define a concept in a conceptual space as follows:

Definition 1.2 (Criterion C)**.** A *concept* is represented as a set of convex regions in a number of domains together with a prominence assignment to the domains and information about how the regions in different domains are correlated.

1.2.2.2 *About the Prototypes*

It was mentioned before that conceptual spaces are suitable for prototype representation and non-monotonic reasoning. In this Section I will briefly illustrate how properties represented by the Criterion P and the prototype theory are deeply linked.

The main idea of the prototype theory is the fact that within a category of objects some of them are more representative than others. For example, robins are judged to be more representative of the category "bird" than penguins and hawks, and so, we can define a robin as the *prototypical* bird. This idea goes in collision with the Aristotelian theory of concepts (Smith and Medin, 1981) and hence with ontologies: in ontologies every member is defined in terms of necessary and sufficient properties, and all instances of a classical concept have same status.

The connection between conceptual spaces properties and prototype can be found in the definition of Criterion P. As a matter of fact, in a convex region each point is featured by a certain degree of centrality (or distance from the central element); it follows from the def-

inition of *metric space* that it is possible to compute its distance from the center of the considered region and such center can be considered as the prototype of the property.

Our dissertation on conceptual spaces ends up here, however, as mentioned before, conceptual spaces offer a very rich set of techniques to reason by exploiting the geometrical nature of the framework. These procedures are not explained here because they are not the focus of this work; more information on the topic can be found in (Gärdenfors, 2001; Dessalles, 2015).

1.3 DUAL-PECCS: A STARTING POINT

DUAL-PECCS is a categorization system that relies on a hybrid knowledge base coupling manually annotated conceptual spaces encoding common-sense information, and an external ontological component, represented by the OpenCyc ontology (Lieto et al., 2015; Lieto, Radicioni, and Rho, 2015). To date, DUAL-PECCS is focused on treating the *animal* domain.

Because the output of TTCS is a set of exemplars, DUAL-PECCS (and its built-in conceptual space) seemed to be a good starting point to test the TTCS results. Choosing DUAL-PECCS to evaluate the TTCS exemplars has a direct consequence: the conceptual space adopted by DUAL-PECCS and TTCS must be the same. Such limitation can be beneficial and a source of issues at the same time:

- Designing a conceptual space (i.e., choosing its quality dimensions) is a very delicate task that requires attention, knowledge of the domain in question and a lot of time for its fine tuning. Instead, using a previously well-conceived conceptual space as a solid start point gave me the opportunity to focus on the accurate development of TTCS processes, rather than on the conceptual space definition.

- Exploiting a conceptual space that is oriented to the representation of a certain domain (the animal domain in the DUAL-PECCS case) naturally brings a set of limitations regarding the type of information that can be exploited from TTCS output. Also, the format of the output produced by TTCS had to be consistent with that used as input for DUAL-PECCS. We can consider this group of constraints as the DUAL-PECCS heritage.

After a very brief description on how DUAL-PECCS works, the following Section will give an outline of the DUAL-PECCS heritage.

SYSTEM DESCRIPTION. In essence, the DUAL-PECCS is a system that executes a two-steps categorization process: it first computes a result based on conceptual spaces, and it then checks the validity of the obtained result against an ontological knowledge base. In order to do so, the system relies on a hybrid knowledge base that includes prototypes and exemplars (required to accomplish the first categorization) and the formal ontology OpenCyc (required to verify the validity of the first computation). In particular, prototype and exemplar-based retrieval is based on a fast and approximate kind of categorization, and benefits from common-sense information associated to concepts. On the other hand, the retrieval of the classical representation of concepts is featured by explicit rule following, and makes no use of common-sense information. The full description of DUAL-PECCS can be found at (Lieto, Radicioni, and Rho, 2015).

To date, DUAL-PECCS relies on a manually populated set of around 300 exemplars; as earlier mentioned the TTCS system should provide the DUAL-PECCS with a larger set of automatically extracted exemplars.

1.3.1 *Heritage of the System*

We defined as heritage of DUAL-PECCS the set of constraints associated to the conceptual space representation adopted by the system. It is important to point out these constraints because they guided the design of the output of TTCS. Some observations regarding DUAL-PECCS heritage:

- DUAL-PECCS obtains the set of exemplars employed during the categorization by parsing a single XML file containing all of them. A complete description of the structure and keywords of this XML is provided in Section 3.1. Please note that DUAL-PECCS also exploits a couple of special "grouping" tags that for the sake of simplicity and practicality will not be mentioned in this work, as their presence is merely dictated by compatibility needs.

- Some of the quality dimensions employed by DUAL-PECCS were strongly focused on the animal domain, and turned out to be impossible to populate in a general purpose setting.

- Some of the concepts in OpenCyc are enriched by WordNetSynsetIds. As we will see, the TTCS fully relies on BabelSynsetIds, so, during the exporting of the built exemplars a translation of the semantic IDs has been required.

- Because the DUAL-PECCS converts the input textual description into a vector, some of the distances computed by the system are in fact based upon string matching rather than on semantic IDs. Consequently, in order to make the stored exemplars comparable with the lexicon-based query vector, TTCS provides as quality dimension value not only a semantic ID but also the list of synset terms corresponding to the assumed value.

2

STATE OF THE ART

2.1 RELATED WORK

Automatically extracting semantic information and annotating texts is an open problem in various fields of AI, and especially for the NLP community (Reeve and Han, 2005). In the last few years many different methodologies and systems for the construction of unified lexical and semantic resources have been proposed.

Some of them are directly referred to the extraction of conceptual spaces representations. Existing approaches, for example, try to induce conceptual spaces based on distributional semantics by directly accessing huge amounts of textual documents to extract the multi-dimensional feature vectors that describe the conceptual spaces. In particular, (Derrac and Schockaert, 2015) try to learn a different vector space representation for each semantic type (e.g., movies), given a textual description of the entities in that domain (e.g., movie reviews). Specifically, they use multi-dimensional scaling (MDS) to construct the space and identify directions corresponding to salient properties of the considered domain in a *post-hoc* analysis.

Other approaches that show some similarities with this proposal aim at learning word embeddings from text corpora. Word embeddings (Turney, Pantel, et al., 2010; Mikolov et al., 2013; Pennington, Socher, and Manning, 2014) represent the meaning of words as points in a high-dimensional Euclidean space, and are in this sense reminding of conceptual spaces. However, they differ from conceptual spaces in at least two crucial ways (and this limits their usefulness for applications in knowledge representation: e.g., in automatically dealing with the inconsistencies). First, word embedding models are mainly

aimed at modelling similarity (and notions such as analogy) and are not aimed at providing a geometric representation of conceptual information (e.g., by representing concepts as convex regions where prototypical effects are naturally modelled). Additionally, the dimensions of a word embedding space are essentially meaningless, while quality dimensions in conceptual spaces directly reflect the salient cognitive properties of the underlying domain.

Differently from these approaches, aiming at extracting conceptual spaces (or similar multidimensional vector representations) from text or textual corpora, I use an approach that explicitly relies on existing linguistic resources. It is assumed that such resources already represent an intermediate step between the lexical level and the conceptual level of the representation, which is the target.

Existing resources, in general, can be arranged into two main classes: hand-crafted resources –created either by expert annotators, such as WordNet (Miller, 1995), FrameNet (Baker, Fillmore, and Lowe, 1998) and VerbNet (Levin, 1993), or through collaborative initiatives, such as ConceptNet (Havasi, Speer, and Alonso, 2007)–; and resources built by automatically combining the above ones, like in the case of BabelNet (Navigli and Ponzetto, 2012).

Although the reader is supposed to be familiar with resources such as WordNet and BabelNet, I provide a quick overview for the sake of the self-containedness.

WordNet (WN) is a lexical database for the English language. Different from traditional dictionaries –organizing terms alphabetically, thus possibly scattering senses– WN relies on the idea of grouping terms into synonyms sets (called *synsets*), that are equipped with short definitions and usage examples. Such sets are represented as nodes of a large semantic network, where the edges represent semantic relations among synset elements (such as hyponymy, hypernymy, antonymy, meronymy, holonymy). BabelNet is a wide-coverage multilingual semantic network resulting from the integration of lexicographic and encyclopedic knowledge from WordNet and Wikipedia (Nav-

igli and Ponzetto, 2010); it extends the constructive rationale of WN –based on sets of synonyms– through the structure of Wikipedia composed of redirect pages, disambiguation pages, internal links, interlanguage links, and categorical information. More on the algorithm used to build BabelNet can be found in (Navigli and Ponzetto, 2012).

Recently, great efforts have been spent to make such resources interoperable, such as the UBY platform (Eckle-Kohler et al., 2012; Gurevych et al., 2012) and the LEMON model (McCrae, Montiel-Ponsoda, and Cimiano, 2012). Specifically, UBY is a lexical resource that combines a wide range of lexica (*WordNet*, *Wiktionary*, *Wikipedia*, FrameNet, VerbNet and *OmegaWiki*),[1] by converting them into lexica compliant to the ISO standard Lexical Markup Framework (LMF). Similar to the UBY-LMF project, also the LEMON project relies on the adoption of the LMF for standardisation purposes: LEMON builds on the LexInfo project (Buitelaar et al., 2009), and it has the purpose of mapping lexical information onto symbolic ontologies. Ontologies, in turn, record the linguistic realizations for classes, properties and individuals. The TTCS system does not directly compare with approaches based on formal ontologies (and standard logic-oriented symbolic representations in general), since the notion of meaning currently considered is complementary to ontological information that, on the other hand, is not explicitly committed to represent and reason on common-sense information. In our case, meaning is associated to terms that are mapped onto conceptual spaces via the IDs provided by BabelNet, which is used as a reference framework for concept identifiers. Additionally, our representation contains a fixed (though not immutable) set of concise relations that are considered useful to grasp the common-sense knowledge component of a concept.

Some similarities can be drawn with works aiming at aligning WordNet (WN) and FrameNet (FN) (Tonelli and Pianta, 2009; Ferrández et al., 2010). The TTCS system shares some traits with the latter

1 Resources marked with emphasized fonts are harmonized in UBY in both English and German versions.

approaches, in that I provide a method to put together different linguistic resources. However, at the current stage of development, the exploited resources are not aligned, but I rather provide a method for the intelligent multi-resource integration and exploitation, aimed at extracting relevant common-sense information that can be useful to fill conceptual spaces dimensions.

None of the mentioned proposals addresses the issue of integrating resources and extracting information to the ends of providing common-sense conceptual representations. The rationale underlying TTCS is to extract the conceptual information hosted in BabelNet (and its vectorial counterpart, NASARI) and to exploit the relations in ConceptNet so to rearrange BabelNet concepts in a semantic network enriched with ConceptNet relations. Differently from the surveyed works, this is done by leveraging the lexical-semantic interface provided by such resources. In the next Section I illustrate my strategy.

2.2 RESOURCES DESCRIPTION

The extraction of semantic information regarding a certain concept is done relying on external resources that can provide such type of data. I have taken into account a set of different resources but only two of them resulted in being appropriate for our purporses: ConceptNet and NASARI. In this Section I will briefly explore those resources.

2.2.1 *NASARI*

NASARI is a set of semantic vectors built on BabelNet synsets and Wikipedia pages. Within TTCS, NASARI is seen as the "database" of all the meanings that can be associated to a given term. More precisely, NASARI is used both to identify the meaning of an input term and to identify all the meanings of all the extracted terms obtained.

The resource is distributed in two different versions:

- *Lexical*: A NASARI lexical vector is composed by a head concept (represented by its *BabelSynsetId* in the first position) and a body, that is a list of terms that are related to the head concept. Each terms is followed by a number that quantifies its correlation towards the head concept. In the second position of the vector we have the Wikipedia page title of the head concept.

- *Unified*: A NASARI unified vector is composed by a head concept (represented by its *BabelSynsetId* in the first position) and a body, that is a list of synsets that are related to the head concept. A synset is represented by its BabelSynsetId (simply ID in the following) and each ID is followed by a number that quantifies its correlation towards the head concept. In the second position of the vector we have the Wikipedia page title of the head concept.

Since the TTCS system works at the semantic level, the unified version of NASARI was adopted. Additionally, in order to avoid continuous accesses to BabelNet, I built an all-in-one resource that maps each ID referred in NASARI vectors onto its synset terms (more details can be found in Section 3.5.0.2).

VECTOR GENERATION. According to the autors (Camacho-Collados, Pilehvar, and Navigli, 2015b), NASARI vectors are obtained by exploiting two different language resources: the expert-based lexicographic WordNet (Miller, 1995) and the collaboratively-constructed encyclopedic Wikipedia. More precisely, given a certain concept represented as tuple $b = (p, s)$, where p is a Wikipedia page and s is a WordNet synset, the vector is built from a set of contextual information obtained by: b, all the pages with an outgoing link to b, s and all the synsets in its direct neighborhood. After all such information is gathered, the most relevant words (for NASARI lexical vectors) and synsets (for NASARI unified vectors) are identified through the use of lexical specificity. This statistical measure, based

Vectors count	2,867,355
Max vector length	565
Min vector length	1
Average vector length	18
Cited IDs	2,868,176
Null Wikipedia page titles	20,867

Table 1: Statistics about NASARI unified.

on a hypergeometric distribution over word frequency, is similar to the standard term frequency-inverse document frequency weighting scheme (Jones, 1972) but is not especially sensitive to different text length.

NASARI STATISTICS. For the sake of completeness, in Table 1 illustrates some statistics about the NASARI unified version.

Note an interesting fact: there are 821 IDs that are cited (i.e., contained in a vector) but that have no vector themselves. Those IDs appear a total of 2,418,369 times.

2.2.2 *ConceptNet*

ConceptNet is a lexical network based on common-sense knowledge. Unfortunately, ConceptNet is only midly semantic: a ConceptNet node is only identified by its URI (e.g., `/c/en/dog`) so there is no immediate way to understand if a relation that starts from a certain node is referring to a certain meaning of that node. For example, the nodes associated to `/c/en/board` could refer both to the administration board and to the wood board. If NASARI can be seen as the semantic center of TTCS, ConceptNet is then the information pool from which we can extract new concepts related to the term in input.

RelatedTo	IsA	PartOf
MemberOf	HasA	UsedFor
CapableOf	AtLocation	Causes
HasSubevent	HasFirstSubevent	HasLastSubevent
HasPrerequisite	HasProperty	MotivatedByGoal
ObstructedBy	Desires	CreatedBy
TranslationOf	DefinedAs	Synonym
Antonym	Attribute	DerivedFrom
CompoundDerivedFrom	DesireOf	CausesDesire
EtymologicallyDerivedFrom	HasContext	Entails
InstanceOf	LocatedNear	LocationOfAction
MadeOf	ReceivesAction	SimilarTo
SymbolOf	SimilarSize	dbpedia/spokenIn
dbpedia/field	dbpedia/genre	dbpedia/knownFor
dbpedia/influenced	dbpedia/influencedBy	dbpedia/languageFamily
dbpedia/mainInterest	dbpedia/notableIdea	wordnet/participleOf
wordnet/adjectivePertainsTo	wordnet/adverbPertainsTo	NotMadeOf
NotDesires	NotHasProperty	NotCauses
NotCapableOf	NotHasA	NotIsA

Table 2: The complete list of ConceptNet relationships.

ConceptNet seems to store information that is valuable to our purpose of populating quality dimensions. In fact, ConceptNet relies on relations such as /r/IsA, /r/AtLocation or /r/CapableOf that could very likely bring us what we need to built a proper exemplar for a given term. Table 2 shows the complete list of relationships used in ConceptNet; however, only a subset of such relations is actually considered by the TTCS system.

CONCEPTNET STATISTICS. Considering only a small subset of the ConceptNet relationships (reported in Table 5) and only the en-

glish nodes, the total of assertion being loaded by TTCS amounts to 9,309,259, overall linking 2,954,401 concepts.

Part II

THE SYSTEM

3

TTCS: TERMS TO CONCEPTUAL SPACES

In this Chapter the two-phases process underlying the TTCS computation is illustrated in full detail. Before providing the description of the system, the quality dimensions that are the desired output of the system are introduced.

3.1 CONCEPTUAL SPACES FORMAT

3.1.1 *Adopted Quality Dimensions*

One of the most relevant design choices is about the definition of quality dimensions. As we know a fraction of this work was already done thanks to the heritage of DUAL-PECCS (Section 1.3.1); however, some considerations about the dimensions at hand has to be made. In my representation every quality dimension can be categorized based on two definitions:

Definition 3.1 (Metric quality dimension)**.** A quality dimension can be defined as *metric*, meaning that its values are points in a metric space. When a quality dimension is metric, it has one or more *fields*. These fields are in fact integral dimensions as defined in Section 1.2.1.2.

For example, the color dimension has 3 fields representing the L*a*b* color space.

Definition 3.2 (ConceptNet and dictionary driven approaches)**.** A quality dimension can be filled filled in through the *ConceptNet-driven* process, meaning that it is directly mapped from one or more ConceptNet relationships. If a quality dimension is not filled in through

Name	BabelSynsetId	Metric	DD	CND
class	00016733n	no	-	IsA
family	00032896n	no	✓	-
shape	00021751n	no	✓	-
color	00020726n	yes	✓	-
locationEnv	00057017n	yes	✓	-
atLocation	00051760n	no	-	AtLocation
feeding	00029546n	yes	✓	-
hasPart	00021395n	no	-	HasA
partOf	00021394n	no	-	PartOf
locomotion	00051798n	yes	✓	-
symbol	00075653n	no	-	SymbolOf
function	00036822n	no	-	UsedFor

Table 3: List of considered quality dimensions; the last two columns indicate respectively whether each dimension is filled via the dictionary-driven (DD) or through the ConceptNet-driven (CND) approach.

the ConceptNet-driven approach, it must be filled via the *dictionary-driven* process, meaning that its values are obtained from an *ad-hoc* dictionary.

Even though at this point these definitions may seem unclear, they will come clear later.

Examples of quality dimensions translation maps and dictionaries are reported in Appendix A.

3.1.1.1 *Quality Dimension List*

The set of the adopted quality dimensions is reported in Table 3. This set of quality dimension is quite oriented to the representation of animals because, as earlier mentioned, the result of the TTCS compu-

tation is built in order to be read by DUAL-PECCS. In the following paragraphs each quality dimension is briefly illustrated, along with some examples.

3.1.1.2 *The class Quality Dimension*

The class dimension is non metric, populated via the ConceptNet-driven approach, and it is used to represent any kind of general classification that may result appropriate for the given object. The closer ConceptNet relationship that fits this kind of description is the IsA edge; so, this dimension is filled thanks to that relationship. Listing 1 shows a few examples of value assignment for the class quality dimension.

```
<!-- From the object 'bus (public transportation)'  -->
<class id="03100490" name="transporting, transportation infrastructure,
     transport policy, traversing, transport, transport development
    manager, transportation of goods, transportation, transport
    infrastructure, conveyance, transpot, transportation industry,
    passenger transport, means of transport, passenger transportation,
     transport industry"></class>

<!-- From the object 'bed' -->
<class id="03405725" name="oak furniture, home furnishing, furniture,
    furniture industry, home furnishings, forniture, bedroom set (
    group), household furniture, piece of furniture, furniture design,
     article of furniture"></class>

<!-- From the object 'banana' -->
<class id="13134947" name="seed pod, fruits, pod types, fruit, fruiting
     shrubs, fleshy fruit, fruiting, culture of fruits, fruity, friut,
     prutas"></class>
```

Listing 1: Three examples of value assignment for the class quality dimension.

3.1.1.3 *The family Quality Dimension*

The family dimension is non metric and populated through the dictionary-driven approach. This quality dimension is similar to class, but it is more specific and its values are obtained from a dictionary, so, they are supposed to be less noisy. The complete dictionary for this quality

dimension is provided in Appendix A. Listing 2 shows a few examples of value assignment for the family quality dimension.

```
<!-- From the object 'hammer'  -->
<family id="04451818" name="indoor equipment, equipment., tools and
    equipment, animal tools, aparejo, tools, tool, tool user, toolcase
    , monkey tools"></family>

<!-- From the object 'whale' -->
<family id="02062430" name="cytaceans, cetacea, cetacian, cetacean
    mammal, whales and dolphins, cetecea, cetecean, cetaceans,
    catecean, blower, cetacean"></family>

<!-- From the object 'ant' -->
<family id="02159955" name="insecto, insect orders, bicho, insect life
    cycle, insect viruses, insect, insect hormones, orders of insects,
     dicondylia, insects, entomofauna"></family>
```

Listing 2: Three examples of value assignment for the family quality dimension.

3.1.1.4 *The shape Quality Dimension*

The shape dimension is non metric and populated via the dictionary-driven process. The purpose of this dimension is indeed the representation of the shape of an object. Listing 3 shows an example of value assignment for the shape quality dimension.

```
<!-- From the object 'wheel'  -->
<shape id="00019193" name="circularity"></shape>
```

Listing 3: An example of value assignment for the shape quality dimension.

3.1.1.5 *The color Quality Dimension*

The color dimension is metric and populated though the dictionary-driven approach. The L*a*b* color space was choose to translate each color semantic ID to a set of numerical values because, unlike others colors spaces, L*a*b* is designed to approximate human vision. More precisely, a color in the L*a*b* color space is defined by three values:

- l: luminosity (integer ranging over the interval $[0, 100]$).

- a: green to red chromaticity (integer ranging over the interval $[-128, 127]$).
- b: blue to yellow chromaticity (integer ranging over the interval $[-128, 127]$).

The complete translation map for this quality dimension can be found as example in Appendix A. Listing 4 shows a couple of examples of value assignment for the color quality dimension.

```
<!-- From the object 'chalk'  -->
<color id="04960729" name="whitenesses, whiteishness, white-finn,
    whiteish, whitishly, white (colour), whiteness, whitishness,
    antique white, whitest, whiter, white, white (political adjective)
    , man on a white horse, whiteishly, whitish, ffffff, color/white,
    white-">
  <l>100</l>
  <a>-0</a>
  <b>-0</b>
</color>

<!-- From the object 'cheese'  -->
<color id="04965661" name="yellow color, royal yellow, yellowish,
    yellowishly, yellowwhite, yellow, yellowness, yellow-white,
    yellowishness, whiteyellow, yellow white, yelow, yellow (colour),
    white-yellow, white yellow, yellower, symbolism of yellow, yellowy
    , yellow (color), dark yellow, yellowest">
  <l>97</l>
  <a>-21</a>
  <b>94</b>
</color>
```

Listing 4: A couple of examples of value assignment for the color quality dimension.

3.1.1.6 *The locationEnv Quality Dimension*

The locationEnv dimension is metric and populated via the dictionary-driven approach. It is used to indicate the location where we can find the object. Note that this dimension was conceived for the animal domain (again, a DUAL-PECCS heritage), and so it is designed for the representation of environments such as `savanna`, `tundra`, `desert`,

etc.. The set of integral dimensions in which a semantic value for this quality dimension is translated is the following:

- humidity: indicates the humidity of the environment (integer ranging over the interval $[0, 100]$).
- temperature: indicates the temperature of the environment (integer ranging over the interval $[-40, 50]$).
- altitude: indicates the altitude of the environment (integer ranging over the interval $[-11000, 8848]$).
- vegetation: indicates the vegetation intensity of the environment (integer ranging over the interval $[0, 100]$).

Listing 5 shows a couple of examples of value assignments for the locationEnv quality dimension.

```
<!-- From the object 'camel'  -->
<locationEnv id="08505573" name="desertic, desert basin, desert flora,
    temperate desert, deserts, desert region, hot desert, the desert,
    sunny country, desert fauna, desert environment, arid region,
    desert, animals in deserts, hot deserts, desert flowers, evolution
     of deserts">
  <humidity>5</humidity>
  <temperature>45</temperature>
  <altitude>0</altitude>
  <vegetation>5</vegetation>
</locationEnv>

<!-- From the object 'monkey'  -->
<locationEnv id="08439126" name="rain-forest, lower canopy, rain
    forests, subtropical rainforests, subtropical rainforest,
    rainforests of the world, strata of the tropical rainforest,
    brazils rainforest, rain forest, rainforest animals, the
    rainforest, the layers of the rainforest, vine thicket, rainforest
     layer, layers of the rainforest, rainforest canopys, emergent
    layer, rainforest destruction and degradation, rainforest,
    rainforests, rainforest layers, primary rainforest, tropcial
    rainforests">
  <humidity>80</humidity>
  <temperature>20</temperature>
  <altitude>0</altitude>
  <vegetation>90</vegetation>
</locationEnv>
```

Listing 5: A couple of examples of value assignment for the locationEnv quality dimension.

3.1.1.7 *The atLocation Quality Dimension*

The atLocation dimension is non metric and populated via the ConceptNet-driven approach. It is a more general version of the locationEnv dimension. In particular, this dimension is mapped from the edge AtLocation of ConceptNet and so it expands the previous idea of environment to a more general interpretation of "location" (e.g., the heart is located at the chest). Listing 6 shows a few examples of value assignment for the atLocation quality dimension.

```
<!-- From the object 'mouse'  -->
<atLocation id="07850329" name="chese, mouldy cheese, cheeses, cheese
    wheel, cheeze, home cheesemaking, cheese powder, chees, coagulated
     milk curd, cheese"></atLocation>

<!-- From the object 'tea'  -->
<atLocation id="03147509" name="cup"></atLocation>

<!-- From the object 'taxicab'  -->
<atLocation id="02692232" name="airstrips, water aerodrome, airport hot
     spots, airpot, auxiliary landing field, civilian airport, land-
    side, airside (airport), drome, flugplatz, airdrome, airporgt,
    aerodrome, public airport, airline destinations, aerfort, air
    sides, airfields, air port, land side, water airport, commercial
    airport, airport infrastructure, air field, jetport, aeropuerto,
    civil airport, landside, air-sides, airport lighting, airport fees
    , airfeild, airpoprt, airport, airports, land-sides, airport
    transfers, airpoprts, airport infrastucture, aerodromes, air strip
    , landsides, small airport, airsides, land sides"></atLocation>

<!-- From the object 'rock'  -->
<atLocation id="03768346" name="mine, mine (industry)"></atLocation>
```

Listing 6: A few examples of value assignment for the atLocation quality dimension.

3.1.1.8 *The feeding Quality Dimension*

The feeding dimension is metric, populated through the dictionary-driven approach and it is used to indicate what kind of alimentation the object has and, indeed, this is another quality dimension inspired by the animal domain. The only integral dimension translated starting from feeding is the foodType field. Such field is an integer between

1 and 5, and its values have been defined based on the work by Getz (2011) in order to keep a coherent distance between the different alimentation types. Listing 7 shows an example of value assignment for the feeding quality dimension.

```
<!-- From the object 'bear'  -->
<feeding id="02075296" name="carnivorism, faunivore, meat-eating,
    obligate carnivores, carnivorous animal, obligate carnivore,
    secondary consumer, carnivory, facultative carnivore, carnivores,
    carnivor, carnivore, carnivorous, beast of prey">
  <foodType>5</foodType>
</feeding>
```

Listing 7: An example of value assignment for the feeding quality dimension.

3.1.1.9 *The hasPart and partOf Quality Dimension*

Both the hasPart and the partOf dimension are non metric and populated via the ConceptNet-driven approach. I treat this dimensions together because they are semantically close even if clearly their meaning is different. The hasPart dimension can be used to represent any part of the object, while the partOf dimension can be used to represent the entities to which the object belongs. The two dimensions are filled from the corresponding ConceptNet edges HasA and PartOf. Listing 8 shows a few examples of value assignment for the hasPart and the partOf quality dimensions.

```
<!-- From the object 'cat'  -->
<hasPart id="02157557" name="human tail, vestigial tail, tail, human
    tails, tail (anatomy)"></hasPart>
<hasPart id="14764061" name="furlike, fur, fur bearers, fur bearing,
    dog hair, fur-like, furbearer, pelt, dog fur, fur bearer,
    furbearing, furbearers, furrily, furs, fur-bearing, animal hair,
    furriness, fur-bearer, guard hairs, down hair, fur-bearers, animal
     fur, fur like, pet hair, furriest"></hasPart>

<!-- From the object 'cake'  -->
<hasPart id="01460457" name="egg, egg (biology), egg (bird)"></hasPart>

<!-- From the object 'lion'  -->
<partOf id="07508486" name="proud (emotion), hubris (ancient greece),
    fake pride, secondary pride, hubris (literature), group pride,
    pridefulness, pride, unhumbleness, arrogantly"></partOf>
```

```
<!-- From the object 'bed'  -->
<partOf id="02821627" name="bedchamber, bedrooms, guest room, chamber,
    sleeping room, sleeping accommodation, bedroom, guestroom"></
    partOf>

<!-- From the object 'cable'  -->
<partOf id="02934168" name="telegraphy equations, line, star quad,
    forex cable, communications cable, balanced transmission line,
    transmission-line, telegraphy equation, transmission lines, cable,
     transmission line"></partOf>
```

Listing 8: A few examples of value assignment for the the hasPart and the partOf quality dimensions.

3.1.1.10 *The locomotion Quality Dimension*

The locomotion dimension is metric, populated via the ConceptNet-driven approach and it is used to indicate how a certain object moves: we are again in presence of a quality dimension inspired by the animal domain. The idea behind this dimension is fairly similar to the one behind the feeding dimension, in fact, we have again a translation into a single integral dimension, corresponding to the name of movement. The movement field is an integer between 1 and 8 and its values, defined following Bejan and Marden theories about locomotion (Bejan and Marden, 2006), are designed in order to have meaningful distances between the different locomotions. Listing 9 shows two examples of value assignment for the locomotion quality dimension.

```
<!-- From the object 'whale'  -->
<locomotion id="00442115" name="swimming, competitive swimming,
    swimming (sport), swimmers, swim">
  <movement>1</movement>
</locomotion>

<!-- From the object 'kangaroo'  -->
<locomotion id="00119568" name="jumping, jump">
  <movement>7</movement>
</locomotion>
```

Listing 9: Two examples of value assignment for the locomotion quality dimension.

3.1.1.11 *The symbol Quality Dimension*

The symbol dimension is non metric and populated through the ConceptNet-driven approach. It is used to represent any metaphorical symbol that the object may represent (e.g., the lion may be symbol of strength or a sword symbol of war) and the ConceptNet relationship from which the population of this dimension is driven is the edge SymbolOf (Appendix A shows an example of the driver file). This dimension was introduced mostly as an experiment to test the adaptability of conceptual spaces. Listing 10 shows two examples of value assignment for the symbol quality dimension.

```
<!-- From the object 'kiss'  -->
<symbol id="07543288" name="love, sacrificial love, loving relationship
    , loves, love interest, loveliest, lovelier, redamancy, true love,
     lovingly, prem (hinduism), chemistry of love, true love (the
    feeling), loveworthy, lovable, latin words for love, idealised
    love, idealized love, l o v e, romantic interest"></symbol>

<!-- From the object 'sword'  -->
<symbol id="00973077" name="war-like, warring nations, weapons and
    warfare, war-likeness, military combat, causes of war, conflict
    zone, warfare, war and militarism, warring, war likeness, military
     conflict, combat operation, cause of war, war like, hot warfare,
    war, resource war, the causes of war, war and military science,
    warred, warlikeness, taisen, warlike, armed conflict, wars, armed
    conflicts"></symbol>
```

Listing 10: Two examples of value assignment for the symbol quality dimension.

3.1.1.12 *The function Quality Dimension*

The function dimension is non metric and populated via the ConceptNet-driven approach. It represents how the object could be used in various environments. The dimension is filled by the corresponding ConceptNet edge UsedFor. Listing 11 shows a few examples of value assignment for the function quality dimension.

```
<!-- From the object 'train'  -->
<function id="00283127" name="locomotion, travel"></function>
```

```
<!-- From the object 'horse'  -->
<function id="04088797" name="ride"></function>

<!-- From the object 'bed'  -->
<function id="01064148" name="rest, ease, relaxation, repose"></
    function>
```

Listing 11: A few examples of value assignment for the the function quality dimension.

3.1.2 *Exemplar Structure*

According to the conceptual spaces theory, an exemplar can be defined as a vector with the same dimensionality of the space, expressing a numerical value for each quality dimension. The set of values results in a point in the conceptual space that, combined with the others, allows region reasoning. This kind of definition is very powerful, but it is hardly reconcilable with the linguistic information treated by TTCS. In other words, while some quality dimensions perfectly fit the theoretic definition (e.g., weight, volume, size), other quality dimensions are more difficult to be translated in an ordered or topological structure (e.g., location, preferred food). This kind of distinction is the reason behind the previously introduced definition of metric dimension (Definition 3.1). However, TTCS treats metric and not metric quality dimension equally: as a matter of fact, every dimension assumes as value one or more IDs and, if metric, will have its values translated by means of a translation map. Once the generated exemplars are processed by DUAL-PECCS, the system treats differently metric and not metric quality dimensions: the metric dimensions are processed accordingly to the conceptual spaces theory, the non metric dimensions are processed by means of string matching and other related metrics.

MULTI-VALUE DIMENSIONS. As it is well known, a vector in a geometrical space has exactly one value for each coordinate of the

space; yet, some of the dimensions previously presented could possibly assume more that one value for a single object. For example, an object can be located (the atLocation dimension) in different places, or it can belong to different classes (the IsA dimension). In order to properly capture all the features of an object, each quality dimension of a single exemplar can have more then one value. From the reasoning point of view (that in this setting is up to DUAL-PECCS) this multi-value feature is accounted for by making a number of copies of the exemplar vector, each of those will have one and only one of the assigned values of the multi-value dimension. Thanks to this replication mechanism, the distance between a query vector and the different "versions" of the examplar vector can be properly calculated. More about this can be found in (Gärdenfors, 2014).

In conclusion, TTCS sees an exemplar as an N-dimensional vector, where N is the dimension of the conceptual space, and where each quality dimension has a set of IDs that work has values.

3.2 THE SYSTEM

The TTCS system takes in input a pair $\langle t, ctx_t \rangle$ where t is a term and ctx_t is the context in which t occurs,[1] and produces as output a set of attribute-value pairs:

$$\bigcup_{d \in \mathcal{D}} \{\langle ID_d, \{v_1, \cdots, v_n\}\rangle\}, \tag{1}$$

where ID_d is the identifier of the d-th quality dimension, and $\{v_1, \cdots, v_n\}$ is the set of values chosen for d. Such values will be used as fillers for conceptual space dimensions $d \in \mathcal{D}$. The output of the system is then a conceptual space representation of the input term.

The control strategy implemented by the TTCS is described in Algorithm 1, and it includes two main steps:

1 Typically, one or more sentences.

- **semantic extraction** phase (lines 1-8): starting from the input term and its context, NASARI is accessed in order to provide that term with a ID so to identify the correct concept. This step corresponds to a simple though effective form of word-sense disambiguation, which relies on NASARI vectors. Once the concept underlying t has been identified, its ConceptNet connections are explored and a bag-of-concepts semantically related to the seed term is extracted;

- **semantic matching** phase (lines 9-17): a new empty exemplar, corresponding to an empty vector in the conceptual space, is created; we then use the bag-of-concepts extracted in the previous phase to identify the values suitable as fillers for the conceptual space quality dimensions.

Algorithm 1 The control strategy of the TTCS system.

input : the pair $\langle t, ctx_t \rangle$

output : $\bigcup_{d \in \mathcal{D}} \{\langle ID_d, \{v_1, \cdots, v_n\}\rangle\}$

1: /* Associate t to a NASARI vector $\vec{v}_i$ and yield the lexicalized concept c^t */
2: $c^t \leftarrow \arg\max_i (WO(ctx_t, \vec{v}_i))$
3: **for each** edge $e_i \in E$ such that $t \xrightarrow{e_i} t'_i$ **do**
4: retrieve the terms t'_i related to t
5: **if** t'_i isRelevant to t for the meaning conveyed by c^t **then**
6: $C \leftarrow C \cup c^t_i$ /* C set of concepts related to c^t*/
7: **end if**
8: **end for**
9: **for each** pair $c^t_i \in C, d \in \mathcal{D}$ **do**
10: **if** d is filled by ConceptNet edges E^d and $c^t \xrightarrow{e_i} c^t_i \mid e_i \in E^d$ **then**
11: /* ConceptNet-driven mapping */
12: fill the dimension d identified by ID_d with c^t_i as value
13: **else if** d is filled by dictionary Dic and $c^t_i \in Dic_d$ **then**
14: /* Dictionary-driven mapping */
15: fill the dimension d identified by ID_d with c^t_i as value
16: **end if**
17: **end for**
18: **return** $\bigcup_{d \in \mathcal{D}} \{\langle ID_d, \{v_1, \cdots, v_n\}\rangle\}$

3.2.1 *Input Format and Generation*

It has been mentioned before that the input of the system consists in a pair $\langle t, ctx_t \rangle$, where t is a term and ctx_t is a bag-of-words representing its context of use. Clearly, it is expected that the given context is pruned of stop words and punctuation. The context of a term is a indispensable element because without it TTCS would not be able to identify the correct meaning of the input. Listing 12 illustrates a small example of a dataset that TTCS takes in input.

```
[
  {
    "term": "lion",
    "referenceText": "The lion is a fast animal.",
    "context": ["lion", "fast", "animal"]
  },
  { "term": "antelope",
    "referenceText": "An antelope can run very fast.",
    "context": ["antelope", "run", "fast"]
  },
  { "term": "dog",
    "referenceText": "Dog are smart animals.",
    "context": ["dog", "smart", "animal"]
  }
]
```

Listing 12: Example of dataset (JSON).

Note the presence of the field `referenceText`, that can be used to illustrate the text from which the context has been extracted.

The input that has been employed during the evaluation is composed by 593 terms and, without loss of generality, each context has been retrieved by accessing the DBPedia page associated to the corresponding term. Every context has been normalized and pruned of stop words and punctuation.

3.2.2 *Semantic Extraction*

The semantic extraction phase creates a bag-of-concepts C containing a set of values for the conceptual representation of a given concept. Two steps can be distinguished: the *concept identification* and the *extraction*.

3.2.2.1 *Concept Identification*

As regards as the concept identification (Algorithm 1, line 3), given the pair $\langle t, ctx_t \rangle$, NASARI is employed to assign a ID id_t. At the end of this step we obtain a lexicalized concept c^t, which is referred to as *seed concept*.

The problem of assigning a ID to the term t can be cast to the problem of associating a NASARI vector to t. A set of *candidate vectors* V is individuated; a NASARI vector $\vec{v}$ is a candidate for the meaning of t iff t is contained in the synset (the set of synonyms) associated to the head of the vector $\vec{v}$.[2] We distinguish three cases:

- V is empty: we cannot identify any concept for t, and the process stops;
- V contains exactly one element $\vec{v}_1$: t is identified by the ID of $\vec{v}_1$;
- V has more than one element: for each $\vec{v}_i \in V$ we compute the weighted overlap between ctx_t and all the synsets that appear as body of $\vec{v}_i$. The vector $\vec{v}_k$ that has maximum weighted overlap (Pilehvar, Jurgens, and Navigli, 2013) is then chosen as the best candidate, so t is identified by the ID of $\vec{v}_k$. If the weight of all the candidates is zero, we cannot identify any concept for t and the process stops.

At the end of the concept identification task a lexicalized concept c^t has been computed, that represents the correct semantics of t by means of a NASARI vector. Table 4 shows the scoreboard table[3] obtained during an execution that had `bus` as input term.

3.2.2.2 *Extraction*

In the extraction step, we access the ConceptNet node associated with t, scan t's outgoing edges $e_i \in E$, and retrieve the related terms $\{t'_1, \ldots, t'_n\}$, such that $t \xrightarrow{e_1} t'_1, \ldots, t \xrightarrow{e_n} t'_n$. The list of considered ConceptNet relations is provided in Table 5.[4] Since ConceptNet does not provide any anchoring mechanism to associate its terms to meaning identifiers (the BabelSynsetIds, in the present setting), it is necessary to determine which edges are relevant for the concept associated to

2 See Section 2.2.1 for a detailed description of the NASARI vectors.
3 A scoreboard table is a textual feedback regarding the concept identification step.
4 The complete list of ConceptNet relationships can be found in Table 2.

BabelSynsetId	Score	Sense
00014082	271	WT: 'bus (computing)', 'bus', 'external data bus', 'computer buses', 'computer bus'
00014081	115	WT: 'bus network', 'bus', 'bus topology', 'linear bus topology', 'bus network'
03145462	15	WT: 'bus (ratp)', 'ratp bus line 119', 'bus', 'ratp bus line 496'
00007329	6,726	WT: 'coach (bus)', 'public service vehicle', 'busses', 'bus'
00014083	230	WT: 'decrepit car', 'bus', 'hoopty', 'old banger', 'jalopy', 'gelopie'

Table 4: The scoreboard table computed during the execution of TTCS on term bus. Each row represents a candidate vector and in this example the penultimate was ultimately selected as meaning for the input term. WT stands for Wikipedia Title.

t, that is in the meaning conveyed by c^t. In particular, when we access the ConceptNet page for t, we find not only the edges regarding t intended as c^t, but also all the edges regarding t in any possible meaning. Ultimately, in this phase we look for the set of concepts related to c^t, that is the set $C = \{c^t_1, \cdots, c^t_k\}$, with $k \leqslant n$.

To select only the edges that concern c^t I introduce the notion of *relevance*. The algorithm is devised as follows:

1. Access the ConceptNet node regarding t and consider its set E of edges.

2. For each $e_i \in E$, we call t'_i the term linked to t via e_i, and we verify that t'_i is relevant to t in the meaning intended by c^t. The term t'_i is relevant if either it appears within the first (highest

IsA	PartOf	MemberOf	HasA
CapableOf	AtLocation	HasProperty	Attribute
MadeOf	SymbolOf	UsedFor	InstanceOf

Table 5: The list of the considered ConceptNet relations.

weighted) α synsets[5] in the NASARI vector of c^t, or if the set of nodes directly linked to the node of t'_i in ConceptNet shares at least β terms[6] with the NASARI vector of c^t.

3. If t'_i is relevant, we then instantiate a concept c^t_i, that we identify through the ID of the first synset in the c^t NASARI vector. Finally, c^t_i is added to the result set C.[7]

For example, given in input the term "bank" and a usage context such as "A bank is a financial institution that creates credit by lending money to a borrower" we first disambiguate the term to identify the concept c^{bank}. Then, we inspect the edges of the ConceptNet node "bank" and thanks to the relevance notion we get rid of sentences such as "bank **isA** flight maneuver" since the term "flight manoeuvre" is not present in the vector associated to concept c^{bank}; conversely, we'll accept sentences such as "bank **HasA** branch". Finally "branch" is identified as a concept and then added to C.

Table 6 shows a portion of the extraction table[8] obtained during an execution that had `banana` as input term.

5 α is presently set to 100.

6 β is presently set to 3.

7 Note that the presence of t'_i in the vector of c is guaranteed only if t'_i was detected as relevant through the first relevance condition. So, if t'_i does not appear in the vector of c^t, the identification process fails, and the term is not added to the result set C.

8 An extraction table is a textual feedback reporting all the details about the extraction step.

Extracted term	Rel. Type	Relevant	Concept Identification
atop refrigerator	AtLocation	no	-
* yellow	HasProperty	yes	WT: 'yellow', 'yellow color', ...
film	InstanceOf	no	-
television show	InstanceOf	no	-
* fruit	IsA	yes	WT: 'fruit', 'seed pod', 'fruits', ...
live thing	IsA	no	-
yellow fruit	IsA	no	-
bunch	IsA	no	-
* herb	IsA	yes	WT: 'herbaceous plant', 'herbaceous', ...
album	IsA	no	-
movie	IsA	no	-
orchard	AtLocation	no	-
store	AtLocation	no	-
fry	HasProperty	no	-
...	...	...	...

Table 6: Portion of the extraction table obtained during the execution of TTCS on term banana. As we can see the identification process is performed only on relevant terms. WT stands for Wikipedia Title.

3.2.3 *Semantic Matching*

The second and final step consists in generating a new exemplar *ex* in the conceptual space representation, and in filling it with the information previously extracted. An exemplar is a list of sets of IDs, where each set corresponds to a quality dimension; it is named and identified in accordance with the seed term t and its meaning c^t.

However, only in some cases C will be rich enough to completely fill the exemplar, which thus can be partially empty; interestingly, conceptual spaces are robust to lacking and/or noisy information.

3.2.3.1 *Dimension Anchoring*

The process of assigning a certain value to a quality dimension is called *dimension anchoring*, and it is carried out for every pair $c_i^t \in C$

and $d \in \mathcal{D}$, where $\mathcal{D}$ is the set of quality dimensions of the conceptual space. The implementation of this process differs according to the way quality dimensions are filled: every quality dimension can be filled either based on ConceptNet or on a dictionary. In the former case (ConceptNet-driven approach) the process of extracting values to fill d leverages the set of edges E^d: if c_i^t is related to c^t by an edge included in E^d (the set of edges that are relevant to dimension d), then c_i^t is a valid value and it is added to ex as value for the d quality dimension, identified by ID_d like indicated in Equation 1. In the latter case (dictionary-driven approach) we exploit the dictionary Dic associated with d: if c_i^t is included in Dic, then it is a valid value and it is added to ex as value for the d quality dimension, identified by ID_d like indicated in Equation 1 (please also refer to Algorithm 1, lines 12 and 15).

3.2.3.2 *Export of the Exemplar*

The last operation performed on the obtained exemplar ex is its export in XML. Such operation is performed one quality dimension at a time and if the dimension that we are exporting is metric, its values are translated. In any case the XML representation of ex contains, for each of its values (i.e., BabelSynsetIds), the WordNetSynsetId corresponding to the value and the list of terms contained in the BabelSynset that has the value as identifier. We note that the exported XML had to satisfy certain constrains in order to produce a set of exemplars readable and understandable by DUAL-PECCS (more details about this point in Section 1.3.1).

TRANSLATION OF METRIC QUALITY DIMENSIONS. As we already know, in the conceptual spaces theory metrical values are fundamental to enable the computation of common-sense reasoning by exploiting the distances between exemplars in the resulting geometrical framework, so, in order to translate metric quality dimensions a set of translation maps was devised (e.g., in the case of *color*, the

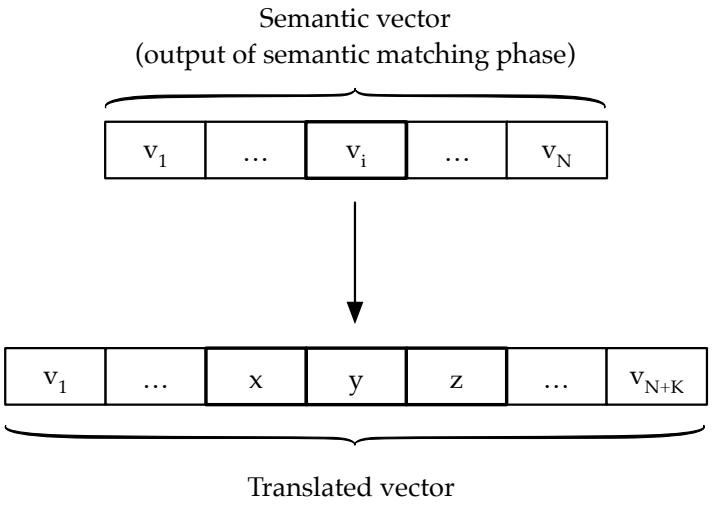

Figure 1: Translation of the value v_i, assigned to a metric quality dimension.

red color is directly translated into its L*a*b color space: $\langle 53, 80, 67 \rangle$). After the new exemplar *ex* is filled with the values extracted through the above mentioned procedure, we translate the values of the metric quality dimensions by exploiting the related translation maps. As shown in Figure 1, the value v_i (that is a BabelSynsetId) is translated in the numeric triple (x, y, z). The length of the vector changes from N to $N + K$ where K is the sum for each translated dimension of the cardinality of the values translated from those dimensions. Listing 13 shows a complete exemplar obtained at the end of the TTCS execution. In comments we have the Wikipedia page titles associated to each value: such pieces of information are reported to make the file more easily readable, but they are not interpreted by DUAL-PECCS. Each tag is a quality dimension, the `id` attribute contains the WordNetSynsetId associated to the value and the `name` attribute contains the BabelSynset senses of the synset having the value as identifier. Particular attention should be paid to the color dimension: its translated values are contained inside the corresponding tag.

```
<!-- Object seed term: taxi, WikiTitle: taxicab-->
<object id="02930766" name="taksio, taxicab driver, taxicab, cab,
    metered taxi, taxi (movie), taxidriver, tacksey, taxicabs, cabbie,
     medallion system, hack, taximeter cabriolet, taxi cab, taxi
    driver, cabman, cabdriver, taxi, taxicab commission, cab driver,
```

```
    taxi cabs, mini-cab, meter taxi, taxi (transport), cabby, taxi
    service">
  <!-- class: "" -->
  <class id="14008806" name="operation"></class>
  <!-- class: "" -->
  <class id="08240169" name="lot, set, band, circle"></class>
  <!-- class: "car" -->
  <class id="02958343" name="car automobile, motor-car, ottomobile,
      auto, cardoor, automobiles, the automobile, motorcar, automobles
      , passenger vehicle, autos, motor car, automobile, self-
      propelling carriage, cars, a car, passenger vehicles, car,
      machine, m1 vehicle, self-rolling carriage, automobil,
      automobilism"></class>
  <!-- class: "" -->
  <class id="04088797" name="ride"></class>
  <!-- color: "yellow" -->
  <color id="04965661" name="yellow color, royal yellow, yellowish,
      yellowishly, yellowwhite, yellow, yellowness, yellow-white,
      yellowishness, whiteyellow, yellow white, yelow, yellow (colour)
      , white-yellow, white yellow, yellower, symbolism of yellow,
      yellowy, yellow (color), dark yellow, yellowest">
    <l>97</l>
    <a>-21</a>
    <b>94</b>
  </color>
  <!-- atLocation: "road" -->
  <atLocation id="04096066" name="road maintenance, causway, highway
      construction, unsealed road, causeways, road building, stone
      chippings, approach road, route, slab stabilization, road, joint
       sealing, roads and highways, road machinery, road transport
      policy, unpaved road, road system, route (highway), roadbuilding
      , cswy, roadways, maintaining road, all-weather road"></
      atLocation>
  <!-- atLocation: "airport" -->
  <atLocation id="02692232" name="airstrips, water aerodrome, airport
      hot spots, airpot, auxiliary landing field, civilian airport,
      land-side, airside (airport), drome, flugplatz, airdrome,
      airporgt, aerodrome, public airport, airline destinations,
      aerfort, air sides, airfields, air port, land side, water
      airport, commercial airport, airport infrastructure, air field,
      jetport, aeropuerto, civil airport, landside, air-sides, airport
       lighting, airport fees, airfeild, airpoprt, airport, airports,
      land-sides, airport transfers, airpoprts, airport infrastucture,
       aerodromes, air strip, landsides, small airport, airsides, land
       sides"></atLocation>
</object>
```

Listing 13: The exemplar obtained by TTCS for the input term “taxi”. Note the presence of multiple values.

3.3 IMPLEMENTATION DETAILS

Purpose of this this Section is the exploration of some of the implementation details that are behind TTCS. The whole project was developed in Java and the most important classes of the system are the following:

- `LConcept.java`, that stores the input term, its ID and its WordNetSynsetId. It is defined during the concept identification step.
- `ExtractedLConcept.java`, that is an extension of `LConcept` and represents a concept that has been extracted from ConceptNet during the semantic extraction phase. This class adds two properties to its superclass: the `LConcept` from which the object has been extracted and the set of edges (class `EdgeType` later described) linking the input `LConcept` and the `ExtractedLConcept`.
- `Exemplar.java`, that represents an exemplar and stores its list of set of values (one for each quality dimension), its ID, its WordNetSynsetId and its label. It represents an exemplar of the conceptual space.
- `TTCS.java`, that contains the `main` method and regulates the whole execution of the system. More precisely, its job is to collect each input term, launch the `semanticExtraction()` and `semanticMatching()` methods and call the export of the potential exemplar.
- `EdgeType.java`, that is an enumeration class and stores all the ConceptNet edges that can be found during the parsing. The class offers some useful methods such as:
  - `parseEdgeType()` that, given a `String`, tries to retrieve and return the corresponding `EdgeType`. This method is massively used during the parsing of ConceptNet.

  - `isOfInterest()` that tells if an `EdgeType` is to be stored and used during the extraction process (i.e., is one of the edges in Table 5).

- `Dataset.java`, that loads the JSON file containing the dataset and offers some utility methods to collect the terms one by one. It is a singleton class preloaded at the start of the system execution.

- `NASARI.java`, that loads the NASARI resource and offers a set of fundamental methods such as:

  - `LConceptIdentification()` that, given a term an its context, performs the concept identification step and thus builds and returns a `LConcept` object. The method is called by the `semanticExtraction()` method.

  - `isRelevantTermForLConcept()` that, given a term t_i' and a `LConcept` c^t, verifies if t_i' is relevant to c^t.
    The method is called during the extraction step (i.e., by the method `extractByLConcept()` later described).

  - `ExtractedLConceptIdentification()` that, given a term t_i', a `LConcept` c^t and a `EdgeType` e_i identifies the meaning of t_i' and thus builds and returns an `ExtractedLConcept` object, which internally refers to c^t and e.
    The method is called during the extraction step (i.e., by the method `extractByLConcept()` later described).

  It is a singleton class preloaded at the start of the system execution.

- `ConceptNet.java`, that loads the ConceptNet resource and offers a set of fundamental methods such as:

  - `extractByLConcept()` that performs the extraction step on a given concept. To accomplish its purpose this method relies on two procedures: `isRelevantTermForLConcept()` and `ExtractedLConceptIdentification()`. In particular, the

method produces as result the `ExtractedLConcept` returned by the `ExtractedLConceptIdentification()` procedure.

- `getTermNeighbourhood()` that, given a term `t`, computes and returns its direct neighbourhood in ConceptNet.

It is a singleton class preloaded at the start of the system execution.

- `ConceptualSpace.java`, that stores the conceptual space schema. All the quality dimensions are instantiated by this class together with their parsed auxiliary data (i.e., translation maps, ConceptNet drivers and dictionaries). The class offers a set of important methods such as:
  - `dimensionAnchoring()` that, given an `Exemplar` ex, an index of a quality dimension `i` and a `ExtractedLConcept` c_i^t, verifies if c_i^t is a suitable value for the i-th quality dimension of the conceptual space. If it is, c_i^t is added in the i-th values set of ex (i.e., as value for the i-th quality dimension in ex). In order to perform its task, this method verifies if the i-th quality dimension needs to be populated via dictionary or ConceptNet drivers. Such distinction is made by checking the Java type of the i-th quality dimension: it can be either a `CNDQualityDimension` or a `DDQualityDimension`, both later described.
  - `exportExemplar()` that, given an `Exemplar`, performs its export, operation that involves both the translation of the metric quality dimensions and the construction of the XML file representing the input exemplar.

  It is a singleton class preloaded at the start of the system execution.

- `QualityDimension.java`, that is an abstract class representing a quality dimension. Its subclasses are `CNDQualityDimension.java` and `DDQualityDimension.java` and they are both instantiated

by the class `ConceptualSpace` during its loading. Each quality dimension explicits if its metrical or not[9] and stores coherently with its features its eventual ConceptNet drivers, dictionary or translation map.

Together with the upon described classes I devised a set of ancillary classes that perform different utility tasks. The following are the most remarkable:

- `Statistics.java`, that during the execution of the system collects various statistical data (it stores about 20 different counters). It is a singleton class.
- `LogUtil.java`, that is responsible of all the logging operations. The system produces two logs: the first is a trace of all the textual output of the system, the second is a statistical report (more information about this topic in Section 3.4) and both are handled by the `LogUtil` object. It is a singleton class.
- `PropertiesUtil.java`, that handles the properties files containing the configuration of the system (more about this in the next Section).

A conclusive schema about the system and its implementation is reported in Figure 2.

3.3.0.1 *Configuration of the System*

One of the typical problems encountered during the developing of a complex system is the tuning of its variables. In order to have a easily and sufficiently configurable software, I devised two different configuration files: `run_config.properties` and `config.properties`.

The latter, `config.properties`, is the list of paths that the system considers for the retrieving of the various resources that it parses (e.g., NASARI, ConceptNet, the dataset).

9 As it is not very informative, I omit how the representation of fields in metric quality dimension is implemented.

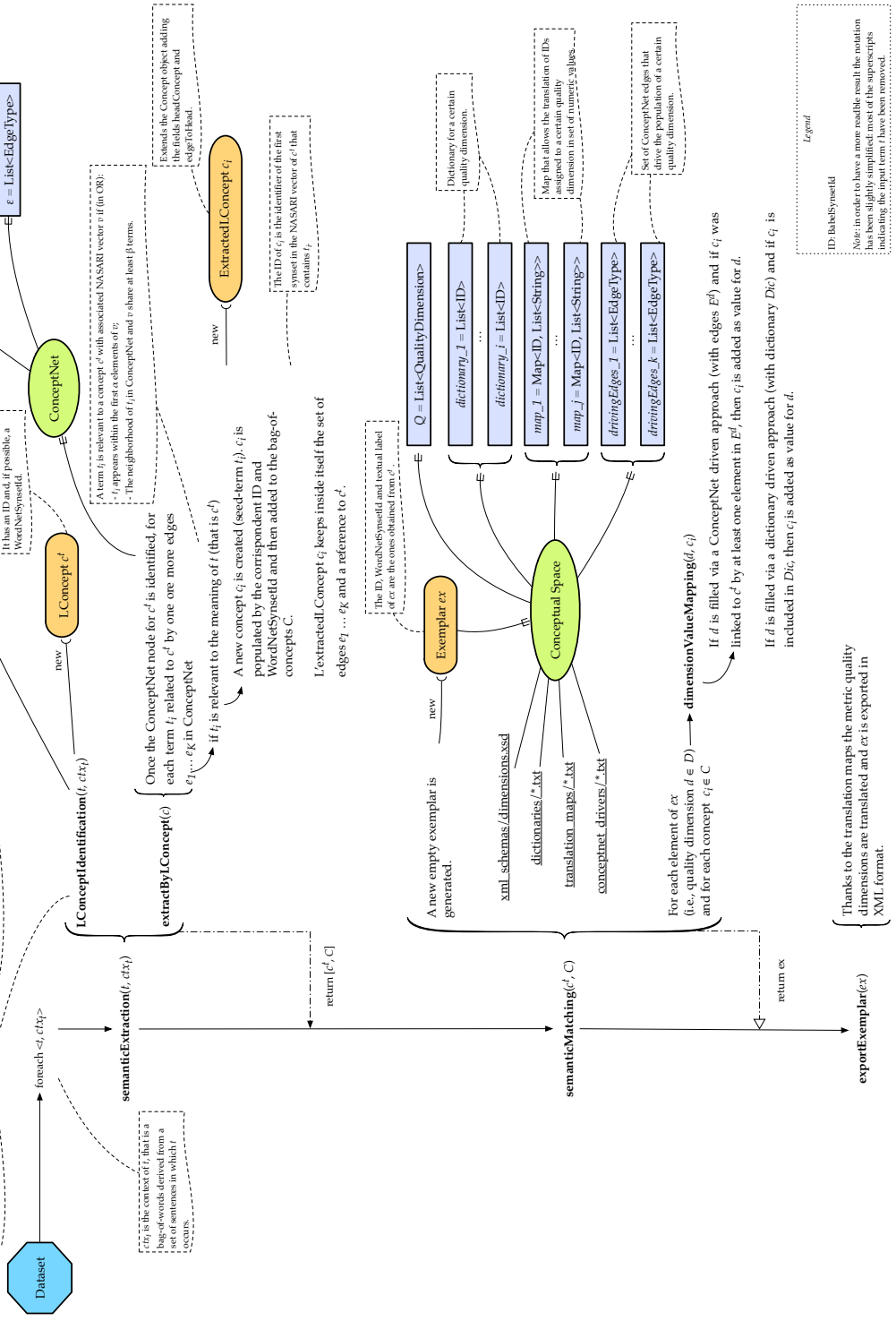

Figure 2: Schema of the TTCS system.

On the other hand, `run_config.properties` stores all the configuration variables that can actually change the performances of the system. In particular, the file allows the configuration of the variables:

- `is_logging`, that enables or disables the logging features.
- `of_interest_edgetypes`, that lists all the ConceptNet edges considered of interest (Table 5).
- `reversed_assertions`, that enables or disables the loading of inversed assertions. More precisely, ConceptNet relationships can be considered in a classical manner (right to left) or in a reversed manner also (left to right). For example, if A IsA B, then B appears in the associations of A. If the reversed assertions flag is active, A appears in the associations of B.
- `alpha` and `beta`, corresponding to α and β, the two constant employed respectively during the concept identification and during the relevance evaluation (Section 3.2.2).
- `printout_modules`, that lists the active printout modules on the execution. For example, the module `extraction_details` enables the printing of the extraction tables, of which an example has been reported before in this Chapter (Table 6).

Thanks to these external configuration files, TTCS can be easily executed on various environments and with different settings.

Please note that TTCS works with local files and so does not require any Internet connection.

THE SERVER. The machine that I used to test and develop TTCS is equipped by 96GB of RAM memory and a Intel(R) Xeon(R) CPU (E5-4610 v2 @ 2.30GHz). The operative system installed on the server is Scientific Linux 7.2 (Nitrogen).

EMPLOYED EXTERNAL SOFTWARES. The system was developed on the IDE IntelliJIDEA,[10] which supports an extended set of convenient tools for the management of the local and server development. As build system I employed Gradle,[11] which is able to automatically deal with dependencies, publish the project, *etc.*.

3.4 OUTPUT OF THE SYSTEM

Clearly, the most important output of the system is the XML file containing the new generated exemplars. But, as mentioned before, the system also produces two logs in order to give some feedback about the execution and to inspect the intermediate steps of the computation.

The first log contains all the textual output of the system such as scoreboard tables and extraction tables (we have seen a couple of examples in Table 4 and Table 6). The data contained in this log allows the understanding of the choices made by the system.

The second log contains a statistical report about the execution and it is very useful to get a first insight of the performances obtained by the system. In Listing 20 we have the full report concerning the execution that produced the exemplars then evaluated in Chapter 4, however, in the next Section I briefly describe the content of the report and therefore have a first impression on how the system behaved.

3.4.1 *Execution Analysis*

In order to build an actual conceptual space, the TTCS took in input a set of 593 cross-domain pairs term-context; such contexts have been obtained by crawling DBPedia, and by extracting the abstracts therein.

10 https://www.jetbrains.com/idea/.

11 http://gradle.org/.

To briefly account for the computational effort required in the concept identification step, the TTCS handled over 2.8M NASARI vectors (by restricting to the first 100 features). By and large, 592 terms out of the initial 593 were disambiguated: the failure was due to the fact that no NASARI vector was found containing the given term in the synset associated to its head. In order to test the quality of the identification, 163 randomly selected exemplars were taken from the output and the association made during this step were manually verified: the identification resulted to be correct in the 95% of cases (155/163). The evaluation was not performed on all the results because verifying the assigned vector of each built exemplar is an operation that can be done only manually and it is very time consuming.

In the extraction step, the TTCS accessed around 10M ConceptNet assertions, linking about 3M concepts. In this step 28K ConceptNet nodes (on average 47.6 per concept) were extracted; 2.3K out of 28K concepts were selected, by finally retaining the relevant and correctly identified ones.

The semantic extraction phase ended up with 516 success cases (the 76 failures were caused by the lack of the ConceptNet node, and not by the extraction of not relevant concepts); these are cases where the bag-of-concepts C contains at least one extracted concept. For 30 lexicalized concepts the resulting bag-of-concept did not contain any suitable value to fill the exemplars. This led to a total of 486 correctly extracted exemplars, and to filling overall 2,388 dimensions (on average 4.9 per exemplar). Whether and to what extent the information extracted by the TTCS approaches human common-sense knowledge is the object of the next Chapter.

EXECUTION TIME. The report (Listing 20) also collects some data about the execution time. The whole execution took about 14 minutes, but roughly half of this time is spent loading NASARI and ConceptNet (that, together, take more than 10GB of data space). Once the resources are loaded, the most time-consuming task seems to be the

identification of a seed concept (about 0.75 seconds each, for a total of 7.2 minutes): this kind of performance is possible thanks to the fact that TTCS computation strongly relies on hash maps.

In conclusion, aside from the constant 7 minutes required to load the resources, we have a linear growth of the computational time: each term takes about 1 second to be processed, so, TTCS seems to be quite scalable (for instance, processing 10,000 terms would take less than three hours).

3.5 ANCILLARY SOFTWARE

In order to prepare all the resources that TTCS needs during its execution I employed a few ancillary software.

3.5.0.1 *HyponymGenerator System*

The HyponimGenerator system its a simple one-class Java software that takes in input a certain BabelSynsetId and retrieves the set of its hyponims (i.e., BabelSynsetIds). The search is performed on the BabelNet graph and so the software requires Internet access and a BabelNet API key.[12]

In order to avoid circular references and hence infinite loops, the search on BabelNet can be limited by setting a maximum search depth.

I exploited the HyponimGenerator during the compilation of the dictionaries for the various quality dimensions: the intent was to speed up the process and to produce the richest possible set of values. However, the result of the HyponymGenerator was manually integrated in each dictionary in order to provide TTCS with a trustworthy source for the dimension anchoring.

12 http://babelnet.org/guide.

3.5.0.2 *NASARISynset System*

The NASARISynset system is a more complex utility and it is composed by three main modules: NSVectorsGen, NSIntegrativeGen and NSAnalyzer.[13]

I introduced NASARI in Section 2.2.1, pointing out that NASARI unified was the most suitable resource for TTCS purposes; I also mentioned that because NASARI unified vectors are fundamentally a set of weighted BabelSynsetIds and because TTCS relies not only on the BabelSynsets identifiers but also on the terms within them, I had to build an all-in-one resource that had to store both this kind of information. I called this resource *NASARISynset* and the accessory software that took care of its generation and analysis (indeed called NASARISynset system) is the topic of this Section.

NSVectorsGen. *NSVectorsGen* is the first component of the system and its purpose is to create the NASARISynset vectors. Such operation can be cast to the problem of enriching every NASARI unified vector by retrieving and appending it to its corresponding synset terms, that is the list of senses contained in the synset that has for ID the ID of the head of the vector. Actually TTCS expects to have access to the synset terms of any of the BabelSynsetIds appearing in the NASARI resource and not only the ones that appear as vector head. Indeed, appending every set of synset terms to each ID occurrence would have resulted in a massively redundant file, so, I decided to include the set of synset terms only for those IDs that appear as head of a vector. Then, while parsing NASARISynset, TTCS can collect each set and link it to the proper ID. However, this implementation choice makes NSVectorsGen prone to one risk: what happens if an ID appears inside a vector but not as a vector itself? The next paragraph shows how this problem was dealt with.

13 NS stands for NASARISynset.

In the end, together with the synset terms, NSVectorsGen also retrieves the WordNetSynsetId of each vector.

Since NASARI unified stores about 3M vectors, their enrichment required a massive amount of accesses to the BabelNet KB. I esteemed that a sequential version of the software would have required about 20 days to do the job, so, I developed a multi-threading version that took only 28 hours to build the final resource.

Listing 14 shows an example of a NASARI unified vector and Listing 15 shows the corresponding enriched vector in the NASARISynset resource.

```
bn:00006807n
Atmosphere
bn:00052888n_3433.51
bn:00017004n_2487.95
bn:00037383n_2044.59
bn:00037400n_1826.61
[...]
```

Listing 14: The NASARI unified vector for the `atmosphere` concept.

```
bn:00006807n
{atmosphere|Celestial body's atmosphere|Planetary atmospheres|The
    atmosphere|Atmsphere|Aerosphere|Celestial body atmosphere|Atmosfer|
    Atmosphere composition|Atmospheric|Planetary atmosphere|Atmospheric
    environment}
wn:09210604n
Atmosphere
bn:00052888n_3433.51
bn:00017004n_2487.95
bn:00037383n_2044.59
bn:00037400n_1826.61
[...]
```

Listing 15: The NASARISynset vector for the `atmosphere` concept. In second position we find all the senses of the `atmosphere` BabelSynset and in third position we find the WordNetSynsetId of the `atmosphere` concept.

NSIntegrativeGen. Some of the IDs that appear inside the body of NASARI unified vectors do not have a vector themselves.

Because NSVectorsGen only retrieves synset terms for IDs appearing as head of vectors, TTCS would certainly not have accessed the synset terms of this "not-having-a-vector" IDs.

The role of the *NSIntegrativeGen* module is then to build a list of *integrative synsets*, namely the set of synset terms of those IDs that do not have a vector in NASARI unified. These synsets are stored in a complementary file that is parsed by TTCS: thanks to it the access to the synset terms of every ID appearing in any NASARI vector is guaranteed.

Listing 16 shows a brief extract of the file containing the integrative synsets.

```
[...]
bn:00008284n_{Band (electronics)|Band (radio)|band}
bn:00008365n_{bank}
bn:00007912n_{severity|badness|severeness}
bn:00007747n_{rear|backside|back end}
bn:00007882n_{troublemaker|trouble maker|troubler|mischief-maker|bad hat}
bn:00007884n_{misfortune|bad luck|tough luck|ill luck}
bn:00007889n_{bad temper|ill temper}
bn:00007652n_{nursery|baby's room}
bn:00007322n_{automobile race|auto race|car race}
[...]
```

Listing 16: An extact of the file containing the integrative synsets. They appear like body-less NASARI vectors.

NSAnalyzer. *NSAnalyzer* is the third and last component of the NASARISynset system. As the name suggests, the purpose of this module is to analyze the NASARISynset file and return some interesting statistics about it; for instance I employed this software to discover the list of IDs not having a corresponding vector in NASARI (the list was then given as input to the NSIntegrativeGen in order to build the integrative synsets). Other statistics calculated by this software include: number of vectors, number of missing Wikipedia page titles, maximum/minimum/average vector length, the number of mentioned IDs, the number of IDs not having a corresponding

vector, how many times they appear and their total average weight. Some of this statistics are reported in Table 1.

4

EXPERIMENTATION AND EVALUATION

A twofold experimentation has been devised, aimed at assessing the quality of the extracted common-sense conceptual information w.r.t. human expectations, and the usefulness of the obtained representations in the context of a specific conceptual categorization task, by using the aforementioned DUAL-PECCS system.

4.1 HUMAN EVALUATION OF THE EXTRACTED CONCEPTUAL INFORMATION

The first evaluation regards the assessment of the quality of the extracted conceptual space representations through the TTCS, based on human common-sense judgement.

EXPERIMENTAL DESIGN For this task 27 persons have been recruited, 17-52 years of age (average 37, 11 females and 16 males), mostly from the Department of Computer Science of the University of Turin, all naïve to the experiment. The human subjects have been provided with a concept (e.g., *dog*) and some related common-sense statements obtained from the representations extracted by the TTCS (e.g., "Dogs have fur", "Dogs are animals", *etc.*). In this setting, participants had to assess each statement by indicating *i*) whether it was appropriate or not for the concept at hand; and *ii*) any further statement reputed essential in order to complete the common-sense description of the considered concept. Participants were randomly split into 2 groups (respectively composed by 12 and 15 participants); subjects had to provide their assessment through an on-line questionnaire containing statements about 15 concepts randomly extracted from the

obtained conceptual space resource. Each piece of information available in the conceptual space has been used to generate a statement, in such a way that all information collected by the TTCS has undergone this experimentation. Overall 173 statements have been proposed to human subjects for evaluation.

EVALUATION METRICS According to the experimental design, for each concept we recorded the number of statements that were found inappropriate (*deletions*), and the number of added statements (*insertions*) were recorded. In particular, two distinct metrics have been adopted: one considering *all* answers, and one examining *relevant* answers. As "*all* answers" metrics all of the responses were recorded, whilst the second metrics has been designed to tame the sparsity of human answers. As regards as the "*relevant* answers" metrics, a *relevant* deletion (or insertion) has been defined as a deletion (or insertion) that occurs when a statement about a given concept is not accepted (or felt to be lacking, but necessary) by at least 3 participants.

For example, given the concept *mouse* and the statements "Mouse lives in desert" and "Mouse is snake food", a relevant deletion was recorded for the former statement which was refused by 3 participants, but not for the latter one, which was refused by only 1 participant. The same threshold has been applied to insertions (to be relevant, a given insertion had to be required by at least 3 participants).

RESULTS The final figures of the results are reported in Table 7. Let us start by considering the result on deletions (Table 7-a). The participants produced on the whole 2,340 judgements; the *all answers* raw datum is that in 1,945 cases they accepted the considered statement (thereby determining a 83.12% of correct results). As regards as *relevant deletions*, 55 statements were refused by at least 3 participants, thus leading to 32% *relevant deletions* (and to its complement, 68% of accuracy). For example, 7 statements were built for the con-

Table 7: The accuracy results on the deletions (Table 7-a) and insertions (Table 7-b).

a. Analysis of the deletions suggested by human subjects.

	accuracy	agreement
all deletions	83.12%	83%
relevant deletions	68.21%	57%

b. Analysis of the insertions suggested by human subjects.

	# insertions	agreement
all insertions	149	3%
relevant insertions	7	58%

cept *cat*, 2 of which have been rejected by at least 3 participants: this ratio (2/7) has then been averaged with those computed for the other 29 concepts. The figure presented in Table 7-a reports such rates. The relevant deletions occurred mainly for statements rather obscure or incorrect, such as "A bullet is spherical", "Banana is a dessert", or "Singer is at location show".

Besides, in order to evaluate how reliable are the collected judgements, the agreement in participants' answers has been also measured. Agreement values were measured about the *all deletions* metrics.[1] The average agreement concerning the *deletions* amounts to 83%; this datum grasps that there is a neat consensus on which statements are acceptable and which ones are not (be them counter-intuitive, or explicitly incorrect, and irrespective of individual preferences and experiences).

1 As the ratio between the number of deletions expressed for a given statement and the number of assessments obtained by that statement: e.g., the statement "Soap has function of scent" has been questioned by 2 participants out of 12. The agreement on such deletion was computed as $2/12 = 16.7\%$.

On the other hand, as regards as insertions (please refer to Table 7-b), a clear data sparsity has been registered which resulted in a low agreement (3%). The few cases where the participants provided *relevant insertions* (that is, proposing at least 3 times the same statement for insertion), point out an information lacking from the set of statements extracted for that concept: e.g., the TTCS missed to extract that"Airplanes fly", that "Camels have two humps" and that "Chameleon changes color". Despite there is a consistent difference in the agreement between the *all insertions* and *all deletions* metrics, there is on the other hand a similar agreement rate on relevant deletions and insertions. Or, equivalently, there is a prominent analogous accord on both mistaken and missing information.

A general insight emerging from the collected data is that the output obtained by the TTCS system mostly corresponds to the characterization of the conceptual information in terms of prototypical, common-sense knowledge.

This element is crucial since most of the available KBs are not equipped with this sort of distilled but salient information. On the other hand, common sense knowledge represents exactly the type of knowledge crucially used by humans for efficient heuristic reasoning, and it could be adopted by artificial systems aiming at providing forms of plausible automatic reasoning. In next Section I show a concrete application requiring this sort of knowledge, extracted by the TTCS system.

4.2 A CASE STUDY IN CONCEPTUAL REASONING AND CATEGORIZATION

The obtained conceptual space resource produced by the TTCS has been additionally evaluated in a practical case study involving a basic conceptual categorization task. As we already know, the system employed to perform the categorization task is DUAL-PECCS (Sec-

tion1.3): given a simple common-sense linguistic description, the target concept had to be identified. In this evaluation the output provided by TTCS is compared by means of two different executions: in the former case DUAL-PECCS used manually annotated conceptual spaces, whilst in the latter one it was fed with the conceptual spaces extracted by the TTCS system. A set of 60 common-sense textual descriptions has been given in input to the categorization system in both conditions (with manual or automatically obtained conceptual space); these stimuli have been built by a multidisciplinary team composed of neuro-psychologists, linguists and philosophers in the frame of a project aimed at investigating the brain activation of visual areas in tasks of lexical processing.

The whole categorization pipeline works as follows: the input to the system is a simple sentence, like "The feline with mane and big jaws", and the correct output is a given category evoked by the description (e.g., the category *lion* in this case). Correctly identified categories represent a gold standard which has been individuated based on the results of an experiment involving human subjects (Lieto et al., 2015); both outputs have thus been compared to human answers.

RESULTS The obtained results are reported in Table 8, that provides a comparison between the accuracy of the categorization system adopting the automatically obtained conceptual representations, and the accuracy of the same system endowed with manually annotated representations.

Although the obtained accuracy is lower than that obtained when using knowledge annotated by hand, the performance of the system using data extracted by the TTCS is still acceptable, especially if we consider the increase in the coverage. In fact, the extracted knowledge bases includes around 500 conceptual representations, while only 300 conceptual representations were present in the manually annotated knowledge base. Also, we are now able to extract salient information to fill conceptual spaces representations with virtually no do-

Table 8: Results on a conceptual categorization task, where the output of the system fed with information extracted by the TTCS system is compared against the results obtained by the categorization system fed with information annotated by hand.

CSs annotation	number of input descriptions	correct categorization
manual	60	52 (87.7%)
TTCS	60	41 (68.3%)

main restriction, thus attaining a much broader resource representing common-sense knowledge in terms of Lexicalized Conceptual Spaces (i.e., conceptual spaces whose representations are fully endowed with BabelSynsetIds).

5

CONCLUSIONS

In this thesis I have introduced the TTCS, a system that –given a textual input– returns the corresponding common-sense conceptual representation encoded in terms of Lexicalized Conceptual Spaces. This representation is obtained through a novel method leveraging different linguistic resources such as BabelNet, NASARI and ConceptNet. The results obtained in a qualitative experimentation, involving the assessment of human users, are encouraging for what concerns the acceptability of the extracted representations and their usefulness in the context of a wider cognitively-inspired categorization system, as well.

Please note that even if in our application the TTCS system was designed to produce a vectorial representation suitable for a conceptual space, it is important to point out that the results obtained during the semantic extraction phase could be easly re-adapted to fill any kind of common-sense representation. The true value of the semantic extraction is in fact the capability to enrich a subset of ConceptNet by assigning BabelSynsetIds to its nodes.

The future work consists in introducing a series of improvements to the current version of the TTCS system:

- ConceptNet can be connected with other resources more encyclopedic-like that can provide further values for the quality dimensions.
- The procedure to obtain the translation maps and the dictionaries should be fully automated, in order to obtain a richer set of possible values for the quality dimensions and to make the overall procedure fully domain independent. One possible solution is to exploit WordNet supersenses as the values for the family

quality dimension; this approach would benefit from the simplified form of word-sense disambiguation currently performed by the TTCS (please refer to Section 3.2.2).

- The notion of relevance employed during the extraction step should be refined. The goal here is the proper managing of text chuncks obtained from ConceptNet: every chunck of text is searched in the various NASARI synsets without performing any kind of normalization or combined search of the words contained in the chunck.

- An alternative method should be devised to accomplish the concept identification of extracted terms. This improvement would bring a significant expansion of extracted terms, with related benefits for the obtained bag-of-concepts.

The final aim is to build a wider, more general, common-sense knowledge resource in term of Lexicalized Conceptual Space. Such knowledge resource would be complementary to, and easily integrable onto existing encyclopedic knowledge resources such as BabelNet, since the interface between the lexical and conceptual level would be grounded on the BabelNet synsets.

In addition, such a resource would benefit from the geometrical features proper to conceptual spaces representations, that are especially helpful in applications that mix different types of reasoning strategies for tasks such as conceptual categorization, question answering, *etc.*. The obtained resource will also allow to extend the present evaluation towards a larger coverage and more quantitative scenario, which will furnish further insights for iteratively refining the TTCS system.

Part III

APPENDIX

A

QUALITY DIMENSIONS DATA

For each quality dimension of the employed conceptual space it is required to provide a set of collateral informations needed during the semantic matching phase i.e. for their automatic population.

In particular:

- Every metric dimension requires a translation map (Listing 17).
- Every dimension populated by a ConceptNet driven process requires a set of driving ConceptNet edges (Listing 18).
- Every dimension populated by a dictionary driven process requires a dictionary (Listing 19).

```
# Translation map for concept 00020726 (color)
# Source: manually populated

# File is tab separated and only the first two columns will be parsed,
    the others can be used for comments.

# Bsi     Translation      Comments

00081052  [100, -0, -0]    white
00010816  [0, 0, 0]        black
00006313  [77, -0, -0]     silver, light gray
00041510  [53, -0, -0]     gray, dark gray
00066631  [53, 80, 67]     red
00053508  [25, 48, 38]     maroon, dark red
00081866  [97, -21, 94]    yellow
00028521  [51, -12, 56]    olive, dark yellow
00653349  [87, -86, 83]    lime
00041648  [46, -51, 49]    green, dark lime green
00024680  [91, -48, -14]   aqua, cyan
00011518  [48, -28, -8]    teal, dark cyan
00011477  [32, 79, -107]   blue, dark blue
00025242  [12, 47, -64]    navy
00052683  [60, 98, -60]    fuchsia, magenta
00065306  [29, 58, -36]    purple, dark magenta
```

Listing 17: Example of translation map (associated to the color quality dimension).

```
# Driving edges for concept 00075653 (symbol)

# File is tab separated and only the first column will be parsed (one tab
     is needed!), the others can be used for comments.

# EdgeType  Comments

SymbolOf    Something that by association or convention represents
    something else
```

Listing 18: Example of set of driving ConceptNet edges (associated to the symbol quality dimension).

```
# Dictionary for concept 00032896 (family)
# Source: manually populated

# File is tab separated and only the first column will be parsed (one tab
     is needed!), the others can be used for comments.

# Bsi     Comments

00003620  amphibian
00005282  arachnid
00024150  crustacean
00046870  insect
00053079  mammal
00055576  mollusk
00062612  fish
00067220  reptile
00010605  bird
00081635  worm
00012462  bovine
00014997  camelidae
00015259  canidae
00040856  caprinae
00017394  cervidae
00011456  cetacean
00031307  equine
00033982  feline
00053563  marsupial
00033151  mustelidae
00060080  pachyderm
00062520  pinniped
00064379  primate
00010639  raptor
00040819  rodent
00068536  ruminant
00033468  suidae
00033559  ursidae

00064208  present
00064208  transport
```

```
00006129  furniture
00012059  book
00046965  musical_instrument
16711476  architectural_element

00036686  fruit
00071215  store
00013722  building
00024507  currency

00077585  tool
```

Listing 19: Example of dictionary (associated to the family quality dimension).

B

LOGS

Listing 20 shows the report file of the execution described in Section 3.4 and Chapter 4.

```
_____________________________#### TTCS System ####_____________________________


Host: 'ground.di.unito.it'
Start time: 10/03/2016 (12:13)


## Setup:

Interesting relationships:                  IsA, PartOf, MemberOf, HasA, UsedFor,
                                            CapableOf, AtLocation, HasProperty,
                                            InstanceOf, MadeOf, SymbolOf,
                                            Attribute
Inverse assertions are considered:          false

Alpha:                                      100
Beta:                                       3


## Execution time details:

NASARI loading:                             00 hours, 00 minutes, 59.66 seconds
ConceptNet loading:                         00 hours, 05 minutes, 22.55 seconds
Concept identification step:                00 hours, 07 minutes, 21.31 seconds
Extraction only (1):                        00 hours, 00 minutes, 03.16 seconds
Concept identification of extracted terms:  00 hours, 00 minutes, 00.08 seconds
-------------------------------------------------------------------------------
Total elapsed time:                         00 hours, 13 minutes, 48.15 seconds


## Average time of operations over one concept:

Identification of base concept:             00 hours, 00 minutes, 00.75 seconds
Extraction only (1):                        00 hours, 00 minutes, 00.00 seconds
Concept identification of extracted term:   00 hours, 00 minutes, 00.00 seconds



## Data details:

Loaded NASARI vectors:                      2,867,355 (length cut: 100 elements)
Loaded NASARI synsets:                      2,868,135 (of which 780 integrated)
Loaded ConceptNet assertions:               9,309,259
Loaded ConceptNet concepts:                 2,954,401
Loaded input terms:                         593


## Concept identification step statistics:

Processed input terms:                      593
Obtained LConcepts:                         592
Concept identification success rate:        99.83%


## Extraction step statistics:

Overall extracted concepts (2):             28,193 (obtained from 592 seed LConcepts)
Overall extraction fails (2)(3):            76
Average per-LConcept extraction rate (2):   47.62
Overall relevant extracted concepts:        4,868 of which 2,343 were successfully
    identified
```

```
## Semantic matching phase statistics:

Overall processed inputs (LConcepts) (4):       516
Obtained exemplars:                             486
Overall populated fields:                       2,388 (of which 384 by dictionary and 2,004
      by ConceptNet edges)
Overall populated metric fields:                41
Average per-Exemplar populated fields:          04.91



---------- Legend ----------
(1) This value doesn't include concept identification time, but does count the time used to
     check the relevance of a concept.
(2) Both relevant and not relevant extracted concepts were counted. Note that every LConcept
     launches one extraction step that can produce several extracted concepts.
(3) The extraction step for a concept C fails if 0 concepts can be extracted starting by C.
(4) The semantic matching phase starts only if the processed LConcept has at least one
     ExtractedLConcept in his correlated bag-of-concepts (i.e., wasn't an extraction fail).
```

Listing 20: Report file of the execution described in Section 3.4 and Chapter 4.

BIBLIOGRAPHY

Algom, Daniel (1992). *Psychophysical approaches to cognition*. Vol. 92. Elsevier (cit. on p. 8).

Ashby, F Gregory (2014). *Multidimensional models of perception and cognition*. Psychology Press (cit. on p. 8).

Baker, Collin F, Charles J Fillmore, and John B Lowe (1998). "The Berkeley Framenet Project." In: *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pp. 86–90 (cit. on p. 16).

Bejan, Adrian and James H Marden (2006). "Constructing Animal Locomotion from New Thermadynamics Theory: Although running, flying and swimming appear to be distinctly different types of movement, they may have underlying physics in common." In: *American Scientist* 94.4, pp. 342–349 (cit. on p. 33).

Berlin, Brent and Paul Kay (1969). *Basic Color Terms: their Universality and Evolution*. Berkeley and Los Angeles: University of California Press (cit. on p. 6).

Buitelaar, Paul, Philipp Cimiano, Peter Haase, and Michael Sintek (2009). "Towards linguistically grounded ontologies." In: *The semantic web: research and applications*. Springer, pp. 111–125 (cit. on p. 17).

Camacho-Collados, José, Mohammad Taher Pilehvar, and Roberto Navigli (2015a). "A unified multilingual semantic representation of concepts." In: *Proceedings of ACL, Beijing, China* (cit. on p. 3).

Camacho-Collados, José, Mohammad Taher Pilehvar, and Roberto Navigli (2015b). "NASARI: a Novel Approach to a Semantically-Aware Representation of Items." In: URL: http://www.aclweb.org/anthology/N15-1059 (cit. on p. 19).

Derrac, Joaquín and Steven Schockaert (2015). "Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning." In: *Artificial Intelligence* 228, pp. 66–94 (cit. on p. 15).

Dessalles, Jean-Louis (2015). "Applications of Conceptual Spaces: The Case for Geometric Knowledge Representation." In: ed. by Frank Zenker and Peter Gärdenfors. Cham: Springer International Publishing. Chap. From Conceptual Spaces to Predicates, pp. 17–31. ISBN: 978-3-319-15021-5. DOI: 10.1007/978-3-319-15021-5_2. URL: http://dx.doi.org/10.1007/978-3-319-15021-5_2 (cit. on p. 11).

Eckle-Kohler, Judith, Iryna Gurevych, Silvana Hartmann, Michael Matuschek, and Christian M Meyer (2012). "UBY-LMF-A Uniform Model for Standardizing Heterogeneous Lexical-Semantic Resources in ISO-LMF." In: *LREC*, pp. 275–282 (cit. on p. 17).

Ferrández, Oscar, Michael Ellsworth, Rafael Munoz, and Collin F Baker (2010). "Aligning FrameNet and WordNet based on Semantic Neighborhoods." In: *LREC*. Vol. 10, pp. 310–314 (cit. on p. 17).

Frixione, Marcello and Antonio Lieto (2012). "Representing concepts in formal ontologies: Compositionality vs. typicality effects." In: *Logic and Logical Philosophy* 21.4, pp. 391–414 (cit. on p. 5).

Gärdenfors, Peter (2004a). "Conceptual Spaces as a Framework for Knowledge Representations." In: *Mind and Matter* 2.2, pp. 9–27. URL: http://www.mindmatter.de/mmpdf/gaerdenfors.pdf (cit. on p. 6).

— (2004b). "How to make the semantic web more semantic." In: *Formal Ontology in Information Systems* (cit. on p. 6).

Gärdenfors, Peter (2014). *The Geometry of Meaning: Semantics Based on Conceptual Spaces*. The MIT Press (cit. on p. 36).

Gärdenfors, Peter (2014). *The geometry of meaning: Semantics based on conceptual spaces*. MIT Press (cit. on p. 3).

Garner, W. R. (1974). *The Processing of Information and Structure*. Maryland: Erlbaum (cit. on p. 8).

Getz, Wayne M (2011). "Biomass transformation webs provide a unified approach to consumer-resource modelling." In: *Ecology Letters* 14.2, pp. 113–24. ISSN: 1461-0248. DOI: 10.1111/j.1461-0248.2010.01566.x. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3032891\&tool=pmcentrez\&rendertype=abstract (cit. on p. 32).

Gurevych, Iryna, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M Meyer, and Christian Wirth (2012). "Uby: A large-scale unified lexical-semantic resource based on LMF." In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 580–590 (cit. on p. 17).

Gärdenfors, Peter (2001). "Reasoning about Categories in Conceptual Spaces." In: *In Proceedings of the Fourteenth International Joint Conference of Artificial Intelligence*. Morgan, pp. 385–392 (cit. on p. 11).

Havasi, Catherine, Robert Speer, and Jason Alonso (2007). "ConceptNet: A lexical resource for common sense knowledge." In: *Recent advances in natural language processing V: selected papers from RANLP* 309, p. 269 (cit. on p. 16).

Jones, Karen Spärck (1972). "A statistical interpretation of term specificity and its application in retrieval." In: *Journal of Documentation* 28, pp. 11–21 (cit. on p. 20).

Lenat, Douglas B, Mayank Prakash, and Mary Shepherd (1985). "CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks." In: *AI magazine* 6.4, p. 65 (cit. on p. 4).

Levin, Beth (1993). *English verb classes and alternations: A preliminary investigation*. University of Chicago press (cit. on p. 16).

Lieto, Antonio, Daniele P. Radicioni, and Valentina Rho (2015). "A Common-Sense Conceptual Categorization System Integrating Heterogeneous Proxytypes and the Dual Process of Reasoning." In: *Proceedings of the International Joint Conference on Artificial Intel-*

ligence (IJCAI). Buenos Aires: AAAI Press, pp. 875–881 (cit. on pp. 11, 12).

Lieto, Antonio, Andrea Minieri, Alberto Piana, and Daniele P. Radicioni (2015). "A knowledge-based system for prototypical reasoning." In: *Connection Science* 27.2, pp. 137–152 (cit. on pp. 11, 65).

McCrae, John, Elena Montiel-Ponsoda, and Philipp Cimiano (2012). "Integrating WordNet and Wiktionary with lemon." In: *Linked Data in Linguistics*. Springer, pp. 25–34 (cit. on p. 17).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*, pp. 3111–3119 (cit. on p. 15).

Miller, George A. (1995). "WordNet: A Lexical Database for English." In: *Commun. ACM* 38.11, pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748 (cit. on pp. 16, 19).

Navigli, Roberto and Simone Paolo Ponzetto (2010). "BabelNet: Building a very large multilingual semantic network." In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 216–225 (cit. on p. 16).

— (2012). "BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network." In: *Artificial Intelligence* 193, pp. 217–250 (cit. on pp. 3, 16, 17).

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14, pp. 1532–1543 (cit. on p. 15).

Pilehvar, Mohammad Taher, David Jurgens, and Roberto Navigli (2013). "Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity." In: *ACL (1)*, pp. 1341–1351 (cit. on p. 40).

Quine, Willard van Orman (1969). "Ontological Relativity." In: *Ontological Relativity and Other Essays*. Ed. by Willard van Orman

Quine. New York: Columbia University Press, pp. 26–68 (cit. on p. 7).

Reeve, Lawrence and Hyoil Han (2005). "Survey of semantic annotation platforms." In: *Proceedings of the 2005 ACM symposium on Applied computing*. ACM, pp. 1634–1638 (cit. on p. 15).

Rosch, Eleanor (1975). "Cognitive Representations of Semantic Categories." In: *Journal of Experimental Psychology: General* 104, pp. 192–233 (cit. on pp. 3, 6).

Smith, E. E. and D. Medin (1981). *Categories and Concepts*. Cambridge, MA.: Harvard university press (cit. on p. 10).

Speer, Robert and Catherine Havasi (2012). "Representing General Relational Knowledge in ConceptNet 5." In: *LREC*, pp. 3679–3686 (cit. on p. 3).

Tonelli, Sara and Emanuele Pianta (2009). "A novel approach to mapping framenet lexical units to Wordnet synsets." In: *Proceedings of the Eighth International Conference on Computational Semantics*. Association for Computational Linguistics, pp. 342–345 (cit. on p. 17).

Turney, Peter D, Patrick Pantel, et al. (2010). "From frequency to meaning: Vector space models of semantics." In: *Journal of artificial intelligence research* 37.1, pp. 141–188 (cit. on p. 15).

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Final version as of May 15, 2016.