Università degli Studi di Milano Bicocca

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Informatica

Concept Invention from text with distributed NLP models, ontology concepts and hyperbolic geometry

Relatore: Prof. Matteo Palmonari

Co-relatore: Dott. Federico Bianchi

Tesi di Laurea Magistrale di:

Manuel Vimercati

Matricola 793553

Anno Accademico 2018-2019

Contents

Chapter 1

Introduction

Humans represent the world with concepts, concepts are fundamental abstractions used by humans to organise their knowledge. Children learn words that refer to general concepts while interacting with real-world objects that are instances of these concepts. For example, the word *'dog'* is bounded with the perceptions (visual, emotional, etc.) caused by lived experiences with dogs, the iterative grounding process allows the child to make a generalisation and create the concept of *Dog* [54].

In Computer Science, in particular in Artificial Intelligence, the ways to represent knowledge (*Knowledge Representation*) is a fundamental research field. A Representation *"is most fundamentally a surrogate that is used to enable an entity to determine consequences by thinking rather than acting, that is, by reasoning about the world rather than taking action on it"* [25]. A representation is, therefore, a resource which partially embodies the whole semantics of the real represented object [25].

Just like for humans words are surrogates for mental concepts [18], in KR *word representations* are surrogates for words. These representations can either be logic structures that connect words or vectors which express relationships between words in term of geometric distances.

Concept representations define the semantics of the *concepts* using different approaches: through Ontological structure such as axioms (explicit relations between types, e.g., defining a concept hierarchy), or based on real-world usage relations between *concepts* (for example *Footballer* and *Stadium* are related *concepts*).

Distributed Models of Language generated from large text corpora have now become the pillars of a multitude of downstream Natural Language Processing (NLP) applications [61], as well as of several theoretical studies in computational linguistics. They can be based on Distributional Semantics theory which states that word's meaning is given by word's context, for example, words relative to cities will have very similar context and the meaning of these words will be similar. In these models, word meanings are encoded into vectors of fixed dimensionality also known as word representations or embeddings. A ***representation space (or word embedding)***, is a vector space that encodes the representation of a word meaning as a vector in a multidimensional space. Figure 1.1 shows and example of distributional word space in which word are coloured by their class.

Several models have been proposed to generate word embeddings, some of these models associating one representation per word, e.g., Word2vec [51] and GLOVE [59], others associating more representations to each word, where each representation expresses the meaning of that word in a particular context, e.g. ELMo [61] and BERT [26]. In these models, word meanings are vectors in a vector representation space. A vector space built from a text corpus covers a very large number of words and, in models like ELMo, word meanings, and accounts for lexical knowledge, i.e., knowledge about meaning and usage of words. This lexical knowledge can be used when working with concepts because the meaning of *words which denote concept* is represented. The meaning of *words which denote nouns of concept's instances* is also represented and can be used to generate the representations of concepts. For example, the representations of the word *'city'* and the representations of city's nouns (*'Madrid', 'London', 'Rome'*) are certainly related to the representation of the concept of *City* and can be used to generate it.

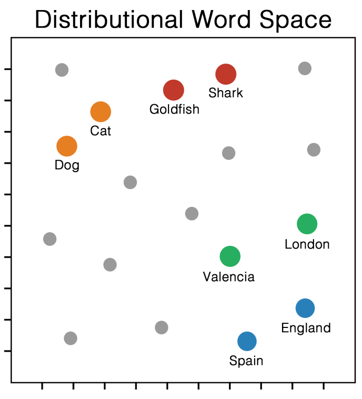

Figure 1.1: Example of distributional word space

Ontologies have been proposed as knowledge representation abstractions designed to represent *main concepts* of the entities in a given domain, *relations* occurring among these entities and *axioms* which specifying the meaning of these concepts and relations in some logical language. In this thesis term Type will be used to refer to ontology classes, also referred to as concepts, e.g., the Type *Person* is used to describe each entity which represents a person. In practice, the term will be used as equivalent to ontological class or concept. Ontologies provide the schema of Knowledge Graphs (KG), another abstraction to represent knowledge at large scale that is nowadays very popular also in the industry. A KG consists of a set of *Types* (semantic classes which summarise real-world objects), *Entities* (which are used to describe single real-world objects) and *Relations* (which connects Entities and Types each other with semantic relations). Ontology types are associated with entities in the KG and organised in a *sub-type graph* by a set of *sub-type axioms*. Despite logic provide the foundation for the semantics of KGs and ontologies, vector-based representations of KGs and ontologies have been recently proposed as a complementary model to associate semantics with KGs (entities and facts) and ontologies (types and relations). These representations are in fact deemed useful for several tasks like KG completion, relation extraction, entity classification and entity resolution [83]. Ontologies and KGs account for structured knowledge, which is a result of abstraction on a given domain. As a consequence, the number of entity types covered in KGs and their ontologies is usually much smaller than the number of words in a language. As a matter of fact, KGs and ontologies cover most relevant entities and are known to be incomplete.

The objective of this work is to define a model for Concept Invention using distributed representations, machine learning, ontological resources, textual resources and Concept Learning techniques. To make this, during the Thesis will be defined a model to map a word space, which contains representations for a large number of words and their meanings (in the order of 200k), to more spaces representing concepts, which come in a number that is usually much smaller than the number of word meanings (DBpedia has 760 Types used to represent concepts). The intuition behind this work is that we can project the vast lexical knowledge encoded in word spaces, into the ontology space, in such a way that we can estimate the representations of those concepts that are not explicitly represented in the concept space.

From another point of view, the task addressed in this thesis is how to estimate relevant and newer concept representations in the concept space, which are not given in the input ontology. To explain the main idea consider the following example: we have concept *Mammal* and *Fish* we do not have concept *Bird*; we want to find a representation in the concept's space that can be considered a good representation for the concept of *Bird*, for instance if it was projected in the ontology tree embedding, the found point is able to express the relations with the concept *Animal* (*Bird* will be a son of *Animal*) and with *Mammal* and *Fish* (*Bird* will be a sibling of them).

The addressed task can be considered a first step towards a variety of more practical applications. For example, given a word meaning, that can be imputed to a word in a sentence, the most similar concept in the ontology can be estimated as well as its distance from other concepts. For instance, the word '*Jaguar*' in the sentence *"Friday I bought a Jaguar"* will be mapped next to the concept of *Automobile*, on the contrary, the same word in the sentence *"The jaguar is a wild cat species"* will be mapped next to the concept of *Animal.*

The model presented in this thesis considers one word meaning space built using ELMO, a state-of-the-art word embedding model that is pre-trained and can generate a vector for each occurrence of each word in the corpus upon which the fine-tuning is made. ELMo is chosen because word embeddings' models like Word2Vec [51] present some problems with polysemous words: Word2Vec generates an embedding for each unique word in the corpus, so polysemous words' representations are contaminated by their meanings, e.g, the vector of *'Jaguar'* will be similar to the vector of *'Maserati'* and the vector of *'Lion'*, but because these two words express concepts which instances have different context, the vector of *'Jaguar'* cannot express all of the meanings and will present a bias based on the corpus occurrences (if in most of the corpus the word *'Jaguar'* is used for the *Animal*, the vector will be more similar to animals' vectors). ELMo generates a vector of each occurrence, so the problem showed above expires.

Instead, as ontology concept space, two models are used to generate the concept embeddings that covers two complementary aspects of concept meaning. One model is a **Hierarchical Concept Representation Vector Space** (Figure 1.2) which represent the *hierarchical relations* between concepts. This representation space is built applying a technique of graph embedding: starting from the DBpedia Ontology Tree[1] which organises all DBpedia's Types in a Tree, the HyperE [66] approach is applied on that tree to obtain an embedding in a Poincarè Disk Model. Hyperbolic Geometric Spaces are used because in these spaces the distance scale in an is naturally suitable to embed trees.

The other model is a **Distributional Concept Representation Vector Space** [6] (Figure 1.3) which represents *real-world usage relations* between concepts. This model is built applying a *Named Entity Linking* tool on the DBpedia abstract corpus and, after the replacement of all entities with

[1]http://mappings.dbpedia.org/server/ontology/classes/

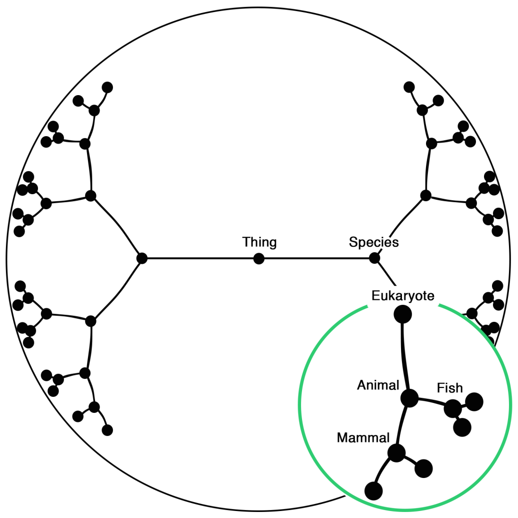

Figure 1.2: Example of concept's tree embedding in Poincaré Disk

their own *minimum Type* in DBpedia and the elimination of all other words, applying a Word2Vec model on the obtained corpora to learn a distributional representation for each found *concept*.

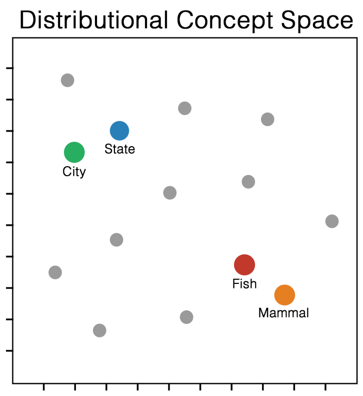

Figure 1.3: Example of Distributional Concept Space

The proposed model consists in a multi-task learning deep neural network that learns how to map an input word meaning into each of the two ontology concept spaces, i.e., to find the conceptual vectors that are estimated to represent the meaning of input word into the concept spaces (one vector for each space). During the learning some tuples $< Word, Concept >$ are used, these tuples are composed by a word which is an entity's name or a class' name paired with the concept belonged by the meaning of the word. Examples of tuple are: $<$ '*London*', *City* $>$, $<$ '*hamster*', *Animal* $>$ etc. Tuples are made like that because a concept is related both with the meaning of a word which is used to describe it and with the meanings of words used for its instances. Using tuples the model learns how to find a concept representation in each concept space which is closer to the real concept representation in that space. To generalise the projection process, the model has to find patterns in the words representations whose meaning belongs to the same *concept*, to another point of view, every point in the *Word representation space* can be mapped somewhere in the *Concept representation spaces.*

The presented work can thus also be considered a **first step towards neural concept invention**. In particular, the model will be able to invent concepts that do not exist in the ontology, based on the available lexical knowledge, where the invention consists in the identification of a vector representing a concept with a given label (the input word).

Clearly, for a consistent invention, many mappings of words representation about the same concept are analysed and a centroid which better represents the majority of prediction will be imputed as the representation of the invented concept.

To evaluate the quality of the invented representations several approaches are listed and discussed. Also, an experimental phase is described and results are reported. As a result, is achieved that the seen concepts gave a bias to the unseen ones based on the aggregate word similarity between the concept's words in the word space.

This thesis is organised as follows: in chapter 2 are introduced preliminary definitions that are used in this work; in chapter 3 the concept invention field is introduced and discussed; in chapter 4 the proposed model is described and discussed; in chapter 5 the implementation of the model is showed; in chapter 6 experiments, research question and results are reported; in chapter 7 conclusion are reported and future works are showed and discussed.

In the Appendix is reported a paper submitted to the conference AIIA 2019[2] based on this thesis work.

[2]https://aiia2019.mat.unical.it/home

Chapter 2

Preliminary Definitions

Before discussing the literature about the main problem addressed into this thesis, i.e., concept invention, we introduce a set of preliminary definitions that are used in this work. These definitions concern *neural networks and multi-task learning*, *distributional semantics in Linguistics*, *ontologies and knowledge graphs*, and *distributed knowledge representation models.*

2.1 Neural Networks and Multi-task Learning

Neural networks are machine learning models based on the Perceptron model [63]. A perceptron is a model inspired by biological neurons, it has n input and n input weights, m output and m output weights and a *bias value.* A perceptron is generally used to approximate a function $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ which takes as argument n data feature and returns a vector. A neural network is a composition of perceptrons (called *unit*) which are organised in layers. Each layer can be composed of a different number of units. A neural network can be used to perform classification or regression task. In the former each class is represented with a dimension and the output vector is used to assign a class; in the latter, the output has exactly the dimension of the solution of the regression problem.

To train a neural network generally a *loss function* is defined to describe the error on the proposed solution, the neural network is trained to minimise this value. The method to learn from a loss function value is the *backpropagation* [65]. The loss function can be seen as a function of weights, the *backpropagation* calculates the gradient of this function and update the values of the weights to minimise the function. This process is repeated for each input example, a common technique to speed up the computation and perform more general *backpropagation* is to apply the *backpropagation* as the mean of gradients computed on a batch of examples with size r. This prevents that outlier example will affect the generalisation process.

The machine learning models assume that the inputs which belong to a certain class will have a characteristic distribution and a machine learning model will works if is able to generalise this distribution starting from the training examples, in such a way that new inputs which show the generalised distribution will be assigned to the correct class.

A very important problem that has to be faced during a neural network training (and even using other machine learning models) is the *overfitting* [38]. This is a phenomenon which causes bad performances of the model even if, during the training, performances seems to be good. The overfitting appears when the model *memorises* the training distribution instead of *generalising*, so is not able to recognise any new input which shows a similar distribution.

To overcome the overfitting problem many techniques are proposed. An architectural solution is the usage of *Dropout layers* [75]: a dropout layer turns off a random fraction of units and so force the network to learn a generalisation because different combinations of units have to be used to correctly assign each example. Another method to face the overfitting is to modify the loss function with a regularisation function. A regularisation function is a function which applies heuristics to avoid overfitting, for example, an assumption is that if the weights have a high variance, the model will probably overfitting, so a regularisation function can be a function which penalises the loss if the variance is too high.

Networks can be used to face more tasks at the same time, this approach is called Multi-Task Learning. A Multi-task learning neural model is designed as the merge of single models (one for each task). There are two kind of architecture (Figures 2.1 and 2.2): Hard-parameter sharing and Soft-parameter sharing. The former architecture uses at first shared layers and only the last is oriented to the single task. The latter connects separated architectures in such a way that layers can share back-propagation.

MTL presents many advantages: the most obvious is that a single model is used instead of n, moreover, a *shared backpropagation* lets a task to learn from others; lastly each task works as regularisation function with respect to the others, this is because learn an intermediate representation which helps to generalise many tasks avoid the overfitting on one of these tasks [64]

Another architecture that is widely used in neural networks is called **Recurrent Neural Network (RNN)**[11]. A RNN takes in input a bunch of examples in which the order is a fundamental feature (e.g. words in a sentence, time-series data). In a RNN the result of the computation on a

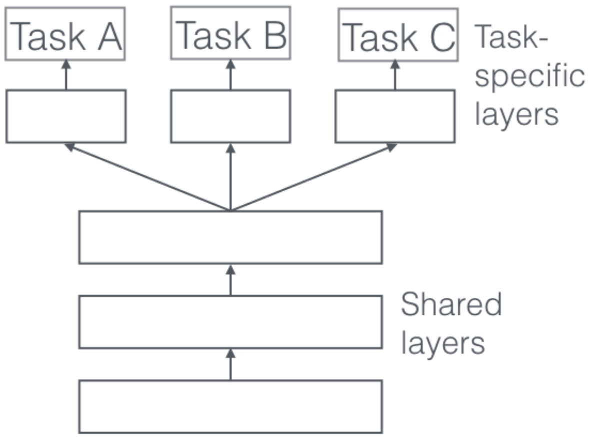

Figure 2.1: Example of hard-parameter sharing architecture for MTL

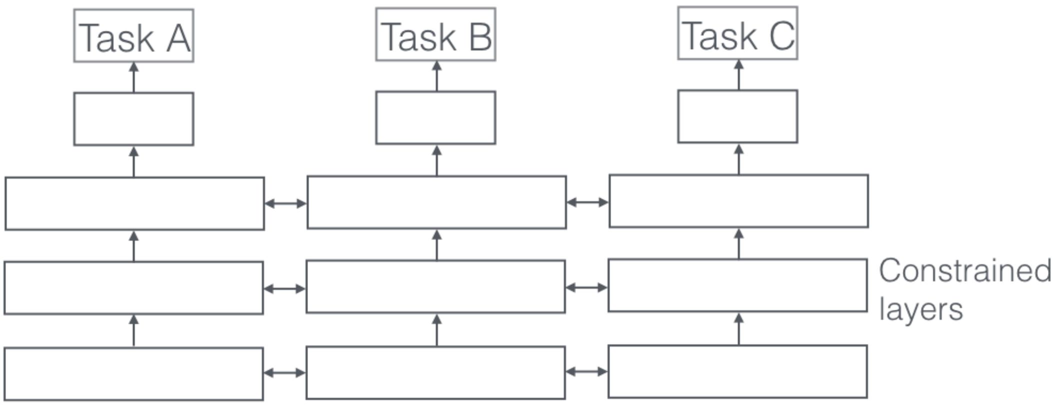

Figure 2.2: Example of soft-parameter sharing architecture for MTL

previous example is used with the following input example to obtain a new result that will be used with the next example (and so on). A RNN can have one single output at the end of the bunch or can produce n outputs, where each output is bounded with an example input.

2.2 Distributional Semantics in Linguistics

Distributional Semantics theory [37] states that the meaning of each word is represented by its context, another definition is: '*The meaning of a word is represented by the way to use it in phrases*'.

The **Distributional Hypothesis (DH)** [44] is fundamental for Distributional Semantic models and is exposed as: *'The degree of semantic similarity* *between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear'*. So the assumption behind any model of distributional semantics is that the notion of semantic similarity between *meanings* can be defined in terms of linguistic distributions.

Distributional Semantics can be applied over ***Corpora*** using ***Statistics***: *Corpora* because the study of the meaning of each word can be based on the word's context in the input corpora, so the distributions (and the meanings) will be strongly connected with the corpora which have to reflect the real usage of words in a language. *Statistics* are also a fundamental component to work with DH because allows the researchers to define functions, metrics and spaces through which a distributional model can be built.

There are two versions of the Distributional Hypothesis: the *'weak'* DH and the *'strong'* DH; these two versions differ for how the *contextual representation* of words is used during the *analysis of meaning.*

The ***'weak' Distributional Hypothesis*** explores the word meaning with distributional analysis. Approaches which are based on *'weak' DH* use distributions to identify semantic properties of words and meanings that are shared and can explain the distributions themselves. These approaches treat meanings like *latent variables* which are responsible for the linguistic distributions and try to detect the meaning of distribution by inspecting a high number of other distributions. Therefore the *'weak' Distributional Hypothesis* is naturally equipped to work with various kind of linguistic research because the final product of this approach can be seen as linguistic features.

The ***'strong' Distributional Hypothesis*** uses the distributional behaviour of words to explain the semantic content at the cognitive level of language. It aims to explain the real meaning of words based only on distributional representation. This way of use distributional representation is good to explain certain properties i.e. classification or part-of property, but lacks on prototypical properties [2] (intrinsic characteristics that are not often in the same context with the main concept). An example of a property that is maintained is in [36] which shows how distributional features can be used to determine the valence (positive or negative) of a predicate.

In general, word meaning is, in part, determined by its combinatorial behaviour in linguistic contexts; peoples who advocate against DH arguing about the difficulty for distributional semantics to represent fundamental properties of natural words like *grounding*, *compositionality* and *inference.*

These properties are hard to represent in a distributional way because these approaches don't use an explicit explanation of meaning; for example the *grounding knowledge* is inspired by human behaviour, a human knows the real meaning of a word that describes an entity (i.e. an animal) if he can use in a sentence, but even if he can recognise the entity in the real world. A distributional representation cannot give this *'cognitive'* information because of its statistical nature. An example of Hypothesis than can represent grounding is the *ECH (Embodied Cognition Hypothesis*) which says that symbol (word, concept) is known only if it is grounded in a sensory-motor experience [70]. In the literature there are many works on *compositionality*, which can be used to generate the vectors of *'word phrases'* (such as *'New York Times'*) using the word vectors of the single words. Simpler approaches [52] find a mathematical operation (such as plus or dot product) between the involved word representations to obtain a good representation of the compound word, the problem of these approaches is that the word order is not accounted (e.g. the vector of *'A dog bites a man'* is the same as *'A man bites a dog.'*. *Inference* can be addressed with hybrid models who combines logic axioms and uses vector as variables/terms [33, 3].

2.3 Ontologies and Knowledge Graphs

An ***Ontology*** *(or Knowledge Graph, KG)* is a logical-axiomatic structure which aims to represent a *domain knowledge.* Formally can be seen as a tuple $< C, E, \subseteq, \in >$, where:

- C is a set of concepts (often also referred to as classes), e.g., *Pianist*;
- E is a set of entities that represent domain objects, e.g., *Elton_John*;
- $\subseteq$ is a *subclassOf* relation that holds between pairs of concepts in C; we assume that the relation is transitive, reflexive and anti-symmetric and defines a partial order over the set of concepts, e.g., $Pianist \subseteq MusicalArtist$;
- $\in$ is the *instance-of* relation that holds between an instance and a concept, e.g., *Elton_John* $\in$ *Pianist*

In Computer Science an Ontology can be represented using ***Entities, Relations and Types***: an *Entity* represents a real world object like *Barack Obama*, *The Tower Bridge*, *Tupac* and so on; a *Type* captures a conceptualisation like *Person, Animal, Planet etc*; a *Relation* expresses the semantic relations between Entities, for example *Friend-of, Birthplace, is-a* and other like this. An example of Ontology is DBpedia[1] which is the Ontological version of Wikipedia. There are KG that does not contain *Entities* like

[1]https://wiki.dbpedia.org/

Taxonomies; those are similar to Hierarchical Graphs because are generally structured as a Tree or POSET.

From the advent of semantic web [69] a large number of open Knowledge Graph were defined, e.g DBpedia [1], Wikidata [80], Yago [76]. Large knowledge graphs require to cover entities of different concepts and their ontologies are often incomplete because of the complex abstraction processes required to model ontologies, where a subset of most important concepts are usually selected, and the intrinsic dynamic nature of these processes. For example, the DBpedia ontology includes concepts such as *'Guitarist'* and *'Singer'* but does not include concepts such as *'Pianist'* and *'DJ'*, despite entities that are specifically pianist and DJs are described therein.

A fundamental characteristics of KG is that is a formal structure; this is a good property of a representing model, but a formal structure is hard to manipulate. Hence ***KG Embeddings*** have been proposed, because distributed representations are versatile and help in tasks like: KG completion[12][84], relation extraction[85][62], entity classification[57][58] and entity resolution[35].

RDFs is a framework which is used to model ontological knowledge through triples. A triple is composed by subject-property-object. RDFs defines basic classes and properties such as:

- `rdf:Resource` Every object described through RDFs is a Resource;
- `rdfs:Class` which is used to define semantical classes;
- `rdf:Property` which is the classe used to describe properties, which are special resources used to connect the resources in the KG by specifying semantical relations between them;
- `rdf:Type` which is the property used to specify the type (semantical class) of a resource, connects entities to types;
- `rdfs:SubClassOf` property used to create a hierarchical order between classes;
- `rdf:Literal` class used to describe literals (basic data like string, numbers, date, etc.).

Inside triples, subjects can be entities or types, objects can be entities, types or literals.

An example of KG is DBpedia which is an open KG based on the information contained in Wikipedia. DBpedia uses RDF to define the ontology and describe the entities and can be queried by using SPARQL[2]. DBpedia is composed of more than six million entities, organised using 760 types and described through more than nine million triples.

2.4 Distributed Knowledge Representation Models

Distributed knowledge representation models are models of KR which represent knowledge through vectors in a vector space model. Since '90, these representations are largely used to work with language.

Working on *natural language* is a very hard challenge because any language has a structure that can be various and filled with exceptions. Another big problem is that to formally represent a language it is necessary a way to represent the dictionary.

The simplest possible distributed model that can be built applying DS is to see each word as a feature and represent meaning by counting the co-occurrence of that word, generating ***sparse vectors*** that counts the presence/absence or the frequency of a words in a context; a sentence could be represented with the frequency of single words [47].

This representation is based on *One-Hot Encoding* and has three main problems: represent every single word is not efficient and useful because the memory needed is too large, and sparse vectors are a further waste of memory.

Based on the *Distributional Hypothesis* the formal meaning of a word could be represented by counting and recording the occurrence of other words in a *context window*. But even in this model, the vectors are sparse.

To avoid the sparsity problems ***dense vectors*** can be used. Dense vectors are obtainable by apply matrix factorisation methods or dimensionality reduction approaches. Those approaches are able to produce ***dense word representation***.

Those approaches can be based on a matrix (frequently called 'Frequency Matrix') which is sparse and for each word has a row which records the frequency (absolute or relative) of other words in its context. Starting from the 'Frequency Matrix' matrix factorisation methods (like SVD or PCA) can be applied.

To obtain dense words representations also a deep neural network can be used. [19] shows how Neural networks can produce ***embeddings*** that are able to improve performances in many NLP tasks. In 2013 Word2Vec

[2]https://dbpedia.org/sparql

[50] has revolutionised distributed model for word representation based on Distributional Semantic. Mikolov et al. defined two models: CBOW and Skip-Gram which learns words representations based on words co-occurrence. CBOW training function is optimised to return a word based on the context; Skip-Gram is trained to make the inverse projection: the training function is optimised to return the context based on the input word. The representation of each word can be generated after the training by picking the weights of the hidden layer in the models. These models create vectors that, in some cases, are clustered by their meaning. For example, the greater part of *Nations* representations is very similar. The similarity is calculated with ***cosine similarity*** that measures the cosine of the angle between vector.

Moreover [Mikolov et al.] underlines emergent and unexpected semantic properties of vectors: those models can perform some analogies via vector's sum. Another property of those embedding is that a word that reflects a compound meaning (like the Volga is a Russian river) can be derived by making a sum between the atomic vectors (e.g., *'Volga'* + *'river'*).

After Word2Vec many neural-based models have been proposed, here will be reported GLOVE [59], ELMo [61] and BERT [26].

GLOVE is a hybrid method because uses a neural network to optimise a function which is based on a 'frequency matrix'. This function is optimised by adjusting the weights in such a way that the scalar product between the weights of a term A and a term B is related with the frequency of B in the context of A and vice versa.

ELMo is a deep learning model which creates a language model during the training on a prediction task. ELMo uses a Bidirectional Language Model (BiLM) to predict a word based on its left context (forward model) or its right context (backward model). A BiLM is composed of two RNN, one which uses the left context and one which uses the right context. In ELMo the RNNs are trained to predict the next (or previous) word based on the left (or the right) context. After the training, ELMo produces so two representations for each word. The BiLM concatenates these representations creating a representation which is based on all words in the context.

The ELMo Architecture has three layers: after the training, the layer 0 encodes the words based on their characters (because ELMo is a Char-based Language Model), the layers 1 and 2 encode the contextual representation of the words.

Once the BiLM is trained, ELMo's representations are obtained through a fine-tuning on the target task which lets ELMo learn how to combine the representations which come from the forward and backward models to maximise the performances in the fine-tuning task.

BERT is based on an ELMo-like architecture which introduces the random masking of words. That approach increases the generalisation capacity of the model and removes the possibility of bias.

Graph Embedding is a task that aims to embody the information contained in a graph in a vector space, preserving the main graph characteristic and preserve node's and edge's properties.

Represent a graph in a vector space is a very challenging task because a graph holds various kind of information, another problem is that graphs can be very huge so the approach has to be scalable; lastly, the embedding dimension represents a trade-off between representation power and complexity and generally depend on the purpose of embedding.

Network embedding algorithms are typically unsupervised algorithms and they can be classified into four groups: matrix factorisation, random walks, construction approaches and deep learning approaches.

Matrix Factorisation methods are generally based on the *Adjacency Matrix* whose properties address the mathematical method; for example if the matrix is positive semidefinite *eigenvalue decomposition* is a common choice. The major problem of the mathematical model is that they are not capable of learning an arbitrary function, e.g., to explain network connectivity.

Random Walk based approaches uses a set of *random walks* to establish paths in which **nodes' order** is the main information. Those methods generate embedding by maximising the sum of the log probability to encounter the nodes in that order (based on embeddings).

A random walk example is *DeepWalk* [60] which uses random walks on the graph to generate the embedding and dot product to reconstruct the graph. Another example is *node2vec* [36] which provide a trade-off between breadth-first and depth-first.

Random walk methods naturally preserve local information about nodes and the properties to be preserved can be controlled by hyper-parameters like walk length, the probability of breadth-first, the probability of depth-first and so on.

The **Construction Approaches** built the embedding node by node, the heart of these methods is the way which is used to select the right position for each node.

Deep Learning based methods are able to capture many properties of graphs, like non-linear structure, node's proximity, node's similarity etc. By including various measures in loss function those approaches can create embeddings that preserve various properties and combinations of them.

Deep learning base methods generally use autoencoders to generate the embedding; they can also use a convolutional layer to capture spatial information, examples are: *Structural Deep Network Embedding* [82] which uses the adjacency matrix and optimise proximity between connected nodes and between nodes that partial share neighbourhood; *Graph Convolutional Network* [41] which through convolutional layer aggregates nodes and iteratively uses the current and the previous aggregation to generate embeddings.

These Autoencoder models are very good methods, but like all other approaches needs a very high number of dimensions to generate good embeddings. This is not a limit of models, but is a limit of the geometry beyond them, because Euclidean Geometry needs many dimensions to express Graph Properties with geometric measures.

Recently, for these motivations, approaches that are based on Hyperbolic Geometry are used.

Non-Euclidean Geometriesare geometric models that born when the Euclid's *parallel postulate* is rejected. This axiom says that: *In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point.*

If the *axiom* is replaced, depending on the replacement, two Geometric models are created: *Elliptical* and *Hyperbolic.* In the *Elliptical Geometry* there are no parallel lines. In the *Hyperbolic Geometry* there are always at least two parallel lines given a line and a point not on it.

There are four models commonly used for hyperbolic geometry: the *Klein model*, the *Poincaré disk model*, the *Poincaré half-plane model* and the Lorentz or *hyperboloid model.* These models define a hyperbolic space which satisfies the axioms of hyperbolic geometry. The Poincaré disk model is a model of 2-dimensional hyperbolic geometry in which the points of the geometry are inside the unit disk. It can be generalised to the *Poincaré Ball Model* which is the same model but n-dimensional. In this model the distance (Equation 2.1) scales in an exponential way the more the points are closer to the limit of the model (r, generally $r = 1$). This scaling is given by the denominator in the function.

$$d(x,y) = arccosh\left(1 + \frac{2*r^2*\|x-y\|^2}{(r^2-\|x\|^2)*(r^2-\|y\|^2)}\right) \tag{2.1}$$

Hyperbolic Embedding uses hyperbolic space to create hierarchical embeddings. A hyperbolic space has a natural hierarchical structure and, intuitively, can be thought of as a continuous version of trees. This property becomes evident in the distance equation of the Poincaré ball model, the distance changes smoothly concerning the norms of the involved points. Points that are close to the origin of the disc are relatively close to all other points in the ball, while points that are close to the boundary are relatively far apart. This local property of the distance is key for learning distributed representations of hierarchical structures and corresponds to the behaviour of shortest-paths in trees. Moreover, Hyperbolic space is naturally equipped to represent the power-law structure in networks, because the notion of hyperbolic distance grow exponentially, while Euclidean distance grows quadratically.

For this reason, many hyperbolic approaches were developed, in 2017 Nickel & Kiela [56] shown that a graph embedding in a hyperbolic space reaches very good performances w.r.t. other techniques in *Reconstruction* and *Link Prediction* task. Based on that work, many researchers have started to experiment with hyperbolic spaces: [48] shows that Hyperbolic Spaces can be used to learn embeddings that can represent different networks properties i.e. similarity based on node's attributes; [77] defines *Hyperbolic Disks Embedding* and use it to embed DAG; [32] defines *hyperbolic entailment cones* and use those objects to create a tree embedding; *hyperbolic entailment cones* are defined on a large class of Riemannian manifolds and induce a partial ordering relation in the embedding space; [43] combines Lorentz embedding, Hearst Pattern and Hyperbolic Entailment Cone to generate a space which is able to infer hierarchical properties of words starting from an NLP corpus; [78] redefines the GLOVE algorithm to learn word embedding in hyperbolic space.

[66] presents *HyperE*, a different approach which generalise [68]. *HyperE* is an approach to embed with very high confidence graph and trees in a Poincarè Disk Model. First, if the graph is not a tree, a *support tree* is built, the support tree is a tree built on the starting graph which maximises the information preserved. Once the tree is obtained, the root is positioned at the origin and its neighbourhood is disposed at a distance $\approx \tau$ (hyperparameter) from the origin and equally distanced each other; one node in the neighbourhood is selected and its neighbourhood is placed using the latter as the root, via circle inversion. The major property of hyperbolic representation is that they can preserve input structure using fewer dimensions w.r.t. classic embeddings, moreover the hierarchical nature of hyperbolic models is a fundamental property and permits to create a hierarchical representation of information.

Distributional Concept Representationis the task which focus is to find an embedding that preserves semantic properties (such as Relatedness) of Concept. It differs from KG Type Embedding mentioned before because the latter aims to represent hierarchical Types' properties.

[6] proposes an approach that is based on a corpus composed by DBpedia's abstract to learn DBpedia Type's representation. Entities in that corpus are annotated with DBpedia's entities and replaced with the *minimum Type*, lastly, all the words that were not annotated are removed. The *minimum Type* is the deeper Type in the DBpedia Ontology Tree.

A *Skip-Gram* model is then applied and the embedding (Type2Vec, T2V) can represent semantic properties between Types. These vectors can have many applications: [6] concatenates Entities vectors with their Types and this representation is able to resolve analogies based task better w.r.t a simple word representation and even to entity embedding.

As said before, the main property of this representation is that it captures a similarity that is similar to the concept of Relatedness and is a way to replicate human-like type-associations. As shown in [6] T2V similarity does not correlate with topological, or path-based, measures because this model expresses a different similarity, but [Bianchi et al.] defines a measure that is able to capture the depth of a node: starting from the assumption that two siblings with depth n are more similar w.r.t. two siblings with depth m where $m < n$ (specialised types share more information that general types), the *Children Information Distribution* is calculated for each node as the *mean similarity between a node and its children.* Results show that *CID* increasing with the grow of depth.

To motivate the usage of a multi-task learning network it was experimented to verify the correlations between different similarity measures (Table 2.4). Each similarity measure expresses a different semantics relation: T2V and Cosine express the distributional semantics respectively between types and between words which refer to types. The Hyperbolic and the Graph measures express the hierarchical semantics between types in a hierarchical tree. The sematch distance is a hybrid measure which expresses the graph distance weighted by the information contents of represented types.

Cosine	1				
Hyperbolic	- 0.6888	1			
Graph	- 0.5479	0.7317	1		
T2V	0.2355	- 0.2177	- 0.2126	1	
Sematch	0.5352	- 0.7116	- 0.8079	0.2982	1
	Cosine	Hyperbolic	Graph	T2V	Sematch

Table 2.1: Correlation between different similarity measures

As can be seen by Table 2.4 the measures which express the same semantic relation are highly correlated. The very low correlation between Hyperbolic and T2V is a motivation to use both of them. This analysis motivates the fact that these measures express a different semantics and can be used jointly to enhance the performance of representation tasks.

Chapter 3

Concept Invention

Concept Invention is positioned in the Computational Creativity research field [20, 86], which aim is to analyse, formalise and implement models which can reach a creative behaviour. Creative behaviour is a notion which comes from cognitive psychology, a definition can be: *"Creativity can be defined as the ability to generate novel, and valuable, ideas. Valuable, here, has many meanings: interesting, useful, beautiful, simple, richly complex, and so on. Ideas cover many meanings too: not only ideas as such (concepts, theories, interpretations, stories), but also artefacts such as graphic images, sculptures, houses, and jet engines)"* [10]. The computational creativity field is based on Boden's work [9] which categorises different kinds of creativity. Boden distinguishes combinatorial re-use of existing ideas, exploratory search for new ideas, and transformational creativity where the search space is also subject to change. Based on this first categorisation Concept invention can be divided into these tasks [87]:

- *Concept Extraction* is the task of extracting and transforming a conceptual representation from an existing but different representation of the same idea; this research field is related to the information extraction methods, for example, Hearts patterns [39] which are able to extract Hyponym/Hypernym relations between words from text corpora;

- *Concept Induction* is based on *instances of concepts* from which a concept or concepts are learned; this can be supervised (Concept Learning) or unsupervised (Concept Discovery).

- *Concept Recycling* is the creative re-use of existing concepts; like the study of semantic shift in temporal word embedding [27] or the Concept Blending field [30, 28];

- *Concept Space Exploration* takes as input a search space of possible new concept and locates interesting concepts in it.

In computational creativity are proposed models with different focuses on creativity: cognitive models which aim to formalise and even implement creative processes [86, 28] and models which use logic or other resources to obtain new knowledge starting from a source of knowledge (e.g KG [45]).

3.1 Concept Extraction

Concept Extraction approaches are related to information extraction techniques [67] and aim to extract knowledge from resources (generally corpora of texts). In these resources, knowledge can be about Entities present in texts and/or Relations between Entities (*Relation Extraction*). This information is recorded through the usage of labels.

The basic models are **rule-based methods** which define rules (lexical patterns) from which knowledge can be extracted and search these patterns into text. Rules can be defined manually or derived by a set of examples, an example of a manually defined rule can be: *"if there is a word preceded by Dr. and the word starts with a capital letter, this word identify an Entity of Person"*. The research on rule-based approach aims to find a small set of rule which can maximise the quantity of information extracted. The application of this kind of rule produces results which generally have high precision but low recall, these performances are motivated by the nature of these rules: finding a small set of rules which is quite general to cover a large set of patterns is really difficult. So, in general, rules are very precise to recognise a pattern but due to this nature, a set of rules can recognise a small set of patterns. Automatic rules can be generated from labelled input text to expand the coverage of the approach, but these will be an unsupervised method, so the precision will fall.

Statistical methods decompose a text into parts and model distributions to label each part jointly or independently. Each word can be seen as a token and can have feature (e.g. grammar features), this information can be used to assign a label (or a probability to belong to a semantical class) to the word similarly to classification tasks. Information can be extracted even with Markov models.

Relation Extraction approaches can be feature-based (like the Entity extraction) or kernel-based which convert text into structures and analyse them to extract relation between entities. An example of relation to extracting can be the marriage that holds between two persons or the relation between a state and his capital, etc.

A common rule-based approach is given by Hearst patterns [39], which is a set of pattern that is able to extract relations from words which appear in a certain pattern into sentences. An example is given by the sentence: *"cat and other animals"* in which a *subclass relation* can be recognised between the word *'cat'* and the word *'animals'*. In concept, extraction approaches the words involved in Hearst pattern can be used as a label to extracted concepts; synonyms can be used to group different words as the same concept. Hearst pattern are commonly used to extract *is-a* relations, other patterns can be used to find other kind of relations, for example, [4] uses pattern to extract *part-of* relations. **Pattern methods** can be focused on **pattern generalisation** that is the searching of general patterns which are able to raise up the recall maintaining good performances on precision like [53] which defines *star patterns* based on wildcards. There are also **Iterative Methods** which uses pattern as query and use new knowledge (matched patterns) iteratively to perform new queries [42]. After applying pattern extraction, distributional representation of words can be used to train, and after predict, if an *is-a* relation holds between two words basing on their representations. Another way to extract hierarchies from distributional data is given by the Distributional Inclusion Hypothesis (DIH) [34]. DIH is one of the fundamental rules in building non-symmetric measures. It assumes that the context of hypernym gathers all contexts of hyponyms.

3.2 Concept Induction

Concept induction methods learn concepts representations using the representations of their respective entities. This field can be divided into **Concept Learning**, which is the supervised version, and *Concept Discovery* which is the unsupervised one. An example of Concept Learning is the task of automatic taxonomy construction from NLP resources like the approaches in [81] which use NLP techniques to learn a generalised version of concepts and creates a structure which organises them in hierarchic order. In Concept Discovery concepts are obtained by applying unsupervised techniques like clustering or LDA [8] on distributed representations of entities.

3.3 Concept Recycling

Concept recycling methods create new concepts by the total or partial re-use of existing concepts. This research field can be partitioned into **Concept Mutation** and **Concept Combination (or Blending)**. The former studies how modify a representation to obtain different concepts, two common mutational operations are *generalisation* and *specification*. Other studies in concept mutation are for example the studies on temporal word embedding: [27] proposes a neural model to obtain comparable representation spaces about different domain corpora, the difference can be temporal or generic, for example, the same approach can be applied to study language difference between different newspaper. A temporal approach highlights semantic shifts of word's meaning through time, for example, the meaning of the word *'apple'* is changed in last thirty years, similar phenomena have happened with other words related to the technology like *'Amazon'* and *'Windows'*.

3.4 Concept Blending

Concept Blending is the study of the creation of a concept by the fusion of different concepts (like *'lightsaber'* from light and sword). Concept Blending is the most in-depth area of study in computational creativity, this is probably because the conceptual blending is a very interesting domain in cognitive researches.

Starting from the work of Fauconnier and Turner [30] which exposes a logical framework to formalise and analyse conceptual blendings many studies are proposed: [15] extend the blending process from *standalone concepts* (like *'lightsaber', 'yacht'...*) to analogy concept blending, the main example is the following: the sentence *"This surgeon is a butcher"* describes a concept which is the blend between the concepts of *'Surgeon'* and *'Butcher'*. The main contribution of this paper is the observation that this kind of blending is *context-dependent*, in fact the *'Surgeon-Butcher'* can have many meanings: a surgeon which makes operations with a high quantity of blood, a surgeon which makes many amputations or a surgeon which makes a very high number of operations. This kind of blending is to be treated with different approaches. Logical structures are usually used to manage concept blending. These structures are used by Fauconnier & Turner and most of the derived work will use logical structure. For example, [5] proposes two logical models for blending. The works of Eppe, Confalonieri et al. [29, 24, 23, 21, 22, 28] propose computational framework which can be implemented using Answer Set Programming and Description logic and are the most recent works on concept blending with logic structures.

Other representation models can be used instead of logic structures to work with concept blending: [17] propose to use vector space models and design concept blending as the generation of the vector which represents the blended concept. An example of this kind of blending is given by the task to retrieve a city name using words which are related to that city, for example, the word *'Madrid'* retrieved based on the representations of *'Spain'* and *'Capital'*. Also, KG can be used to generate new concepts: [45] proposes a pipeline to create new storylines or story characters by a blending operated on KG which describes the story, the blending is driven by directives so the obtained representation can be seen as a blending from the directives and the KG.

3.5 Concept Invention for Ontology Completion

The Ontology Completion task can be faced with two strategies: by finding new triples or by extending the set of concepts and defining new *subclassOf* relations. The former can use KG embeddings which encode entities as vectors and properties as spatial transformations to predict new triples [13, 55], recent methods combines *rule templates* with DR created using gaussian distributions [14]; the latter derive new concepts and rules starting from the Ontology itself. These approaches can be designed with *Inductive Logic Programming* [16] or even with *Statistical Relational Learning* [79, 74]

Ontology completion can be seen as an automatic taxonomy construction problem, text can be used as a source of information for this building task. Most approaches use only nouns as the bricks for ontology building and disregard any ontological relations between other word classes [7]. Hierarchical clustering can be used to find hierarchy between words, clustering is applied on distributed models and the model used to generate them is crucially to obtain good results.

Taxonomy induction gathers tasks which aim to create taxonomy from is-a relations, this field is divided in: **Incremental Learning** which gather approaches that are based on a KG used as seed [72, 71]; **Clustering** problem where similar terms clustered together may share the same hypernym [49, 73]; **Taxonomy Cleansing** which removes wrong is-a relations or add transitivity to improve the quality of an ontology [46].

Recent works are: [43] which uses text corpora to infer hierarchical concepts, basing on an hyperbolic embedding and a revisited version of Hearst patterns; [88] which generates a hierarchical clustering of terms with each node consisting of several terms and finds concepts that should stay in the same cluster using embedding similarity.

3.6 Concept Invention in Distributed Models

The model presented in this thesis is a first work towards concept invention in distributed models. This work is in Concept Invention field because the presented Invention process has two output: a Concept representation in a Distributional Space and a Concept representation in a Hierarchical Space. The presented process generates representations which are in populated semantic spaces and that can be compared with already represented concepts. This process is a concept invention process and not a taxonomy learning process even if a hierarchical concept space is used because in this model a generated concept has two representations with two different semantics, and to explain the semantics of the generated concept both are to use, because both are valid at the same manner.

Chapter 4

Concept Invention in Distributed Models

In this thesis, we define the task that is the main subject of this work: *concept invention* with distributed models. Initially, I will illustrate the approach proposed to concept invention, and the evaluation strategy applied to evaluate such approach.

The approach for Concept Invention with Distributed Models discussed in this thesis has two main components:

- **Learning to Map Distributed Word Representations to Distributed Concept Representations:** this component takes a distributed word model as input and learns to return one vector for each distributed concept space as output; after this mapping is learnt, when an input word is given, e.g., a name of an entity that appears in the distributed word space, the mapping function returns vectors in the concept spaces, e.g., the vectors that are estimated to represent the entity type in the distributed concept spaces. **Example:** we learn the mapping from pairs $<$ *'London'*, *City* $>$, $<$ *'Rome'*, *City* $>$, where the first word is either a word used to refer to the *City* concept, or a word used as name for a city.

- Concept Imputation: this component uses the learnt mapping function to find a suitable representation of a new concept using names of entities that belong to the new concept; in particular, given a set of words that are names of instances of a concept that does not belong to a given ontology, this component creates one vector in each distributed concept space that provide the representation for the new concept. **Example:** let us assume that the concept *Country* is not represented in the ontology, but we have a set of names of instances of the concept;

given words such as *'Germany'*, *'Italy'*, the component generates a vector that represents the concept *Country* in a hyperbolic concept space and in a distributional concept space.

In other terms, the proposed approach is able to project word representations into concept representation spaces, so can generate a concept starting from a word's meaning (embodied by the word representation).

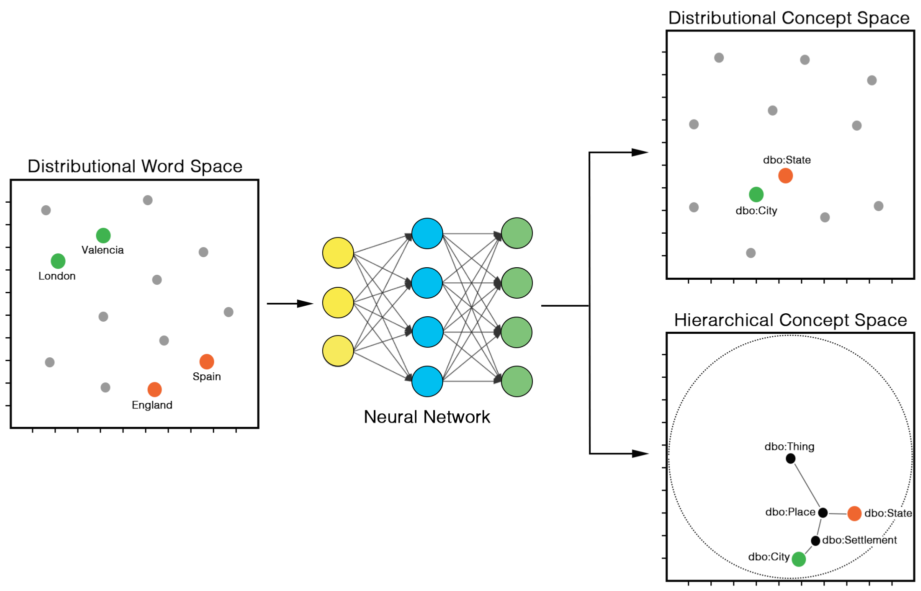

Figure 4.1: Multi-Task learning for concept invention schema

4.1 Map the Distributed Word Space to Distributed Concept Spaces

We define concept invention as a mapping function $\Phi : \mathbb{R}^n \to \mathbb{R}^m \times ... \times \mathbb{R}^q$ from word space to concept spaces. This function can be approximated with any machine learning or statistical model. In this thesis, neural models will be presented and discussed.

To make an example we refer to an example setup composed of:

- One word representation space built starting from a corpus and applying the DS.
- Two concept representation spaces, one that accounts the hierarchical relations between concepts (like taxonomy) and another which embodies the distributional relations between concepts.

- A Multi-Task Learning model to approximate Φ.

The effective creation of the distributed models and the neural network setup will be illustrated in the Implementation chapter

4.1.1 Building the Distributed Word and Concept Models

Starting from a set of concepts C, two subsets $C_T \subset C$ and $C_I \subset C$ are created with $C_T \cap C_I = \emptyset$. C_T represents the set of training concepts and will be used to learn Φ, C_I is the set of invention concepts and their representation and will be created using Φ. Each conceptual space will be created basing on C_T, in our example, we created a hierarchical representation space H_{C_T} and a distributional representation space D_{C_T}.

A corpus T is used to create a distributional space V_W of words W. A schema to which illustrate this procedure is showed in Figure 4.2.

4.1.2 Building the Training Data

Performing concept invention starting from words can be seen as a *Concept Induction* task. To learn a mapping function which can project from words to their respective concepts a dataset composed of examples of alignments $< word, concept >$ is needed.

This kind of dataset is *model and vector independent*, because is an alignment between *words* and *concepts* (e.g., $< London, City >$) and can be used on different models. Since, in our example, we using DMs we need to retrieve vectors to compose the training example tuples. So, starting from previous couples and using the aforementioned vector representation spaces the triples $< w_{word}, d_{concept}, h_{concept} >$ will be created.

Therefore, the dataset is built in the following way:

1. For each $c \in C_T$ retrieve from a KG all the entities ε for which the relation $\varepsilon \in c$ is defined in the KG, this step creates the E_c set.

2. For each $\varepsilon_c \in E_c$ find the word $w_{\varepsilon_c} \in W$ which represents ε_c (for instance, the word *'kangaroo'* for the DBpedia entity `dbr:kangaroo`), and create the couple $< w_{\varepsilon_c}, c >$; if the representation of an ε_c cannot be found in W, discharge that ε_c.

3. For each couple $< w_{\varepsilon}, c >$, retrieve all vectors $v_w \in V_W$, for each v_w create a tuple $< v_w, d_c, h_c >$ with $d_c \in D_{C_T}$ and $h_c \in H_{C_T}$

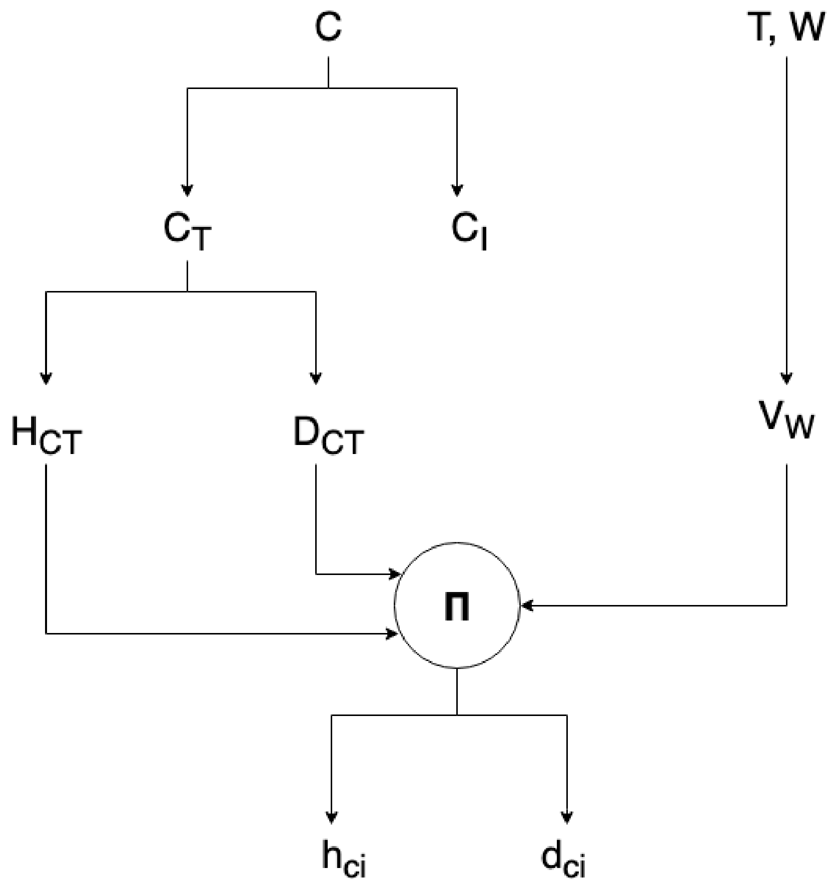

Figure 4.2: Schema of resource usage

The training dataset can now be used to approximate Φ. Once the function Φ is learnt, the model is able to generate new concept representations. Starting from C_I and applying (1) and (2) of the previous procedure words $w_{c_i} \in W$ which belong to concepts in C_I can be retrieved. Φ can be used to predict $h_{w_{c_i}} \in H_{C_T}$ and $d_{w_{c_i}} \in D_{C_T}$ and performing a concept invention process. The quality evaluation of these vectors will be discussed in the next sections.

4.2 Concept Imputation

A concept invention process which starts from words has to be suitable to face one problem: different words can belong to the same concept. The number of words which belong to a concept becomes higher with the grow of abstraction of concept. For example, in the dataset there are a lot of words which belong to the concept *'Fish'* and there is also an incredibly higher number of words for the concept of *'Animal'*, which is on a higher level of abstraction.

In this thesis to create a single representation of an invented concept starting from multiple representations the centroid is used. A centroid of a set of vectors is a vector which minimises the distances with all input vectors.

4.2.1 Centroid in Euclidean Space

Compute the centroid in Euclidean space (like the distributional spaces) is simple:

- Starting from a set of vectors V the centroid $c(V)$ can be obtained with the formula:

$$c(V) = \frac{\sum_{v \in V} v}{\|V\|} \tag{4.1}$$

This centroid will minimise the Euclidean distance and the cosine similarity with the input vectors.

4.2.2 Centroid in Poincaré Disk Space

Once the predictions on the test set are obtained, **for each concept**, the hyperbolic centroid is calculated through a procedure which is based on the Hyperboloid model.

A point $(x_0, ..., x_n) \in \mathbb{H}^n$ in the Poincarè Disk Model can be mapped to a point $(t, y_0, ..., y_n) \in \mathbb{S}^+$ in the Hyperboloid Model through this formula:

$$(t, y_i) = \frac{(1 + \sum_{i=0}^{n} x_i^2, 2x_i)}{1 - \sum_{i=0}^{n} x_i^2} \tag{4.2}$$

To map from $\mathbb{S}^+$ to $\mathbb{H}$ this formula can be used:

$$x_i = \frac{y_i}{1 + t} \tag{4.3}$$

To obtain the centroid this procedure can be applied:

- For each point $h_i \in \mathbb{H}$, calculate the Hyperboloid Projection $P(h_i) \in \mathbb{S}^+$
- The centroid $P(h_c) \in \mathbb{S}^+$ is obtained by:

$$P(h_c) = \frac{\sum P(h_i)}{abs(< \sum P(h_i), \sum P(h_i) >)} \tag{4.4}$$

where the $< u, v >$ in the Equation 4.4 is the Lorentzian Inner Product.

- The centroid $h_m \in \mathbb{H}$ is obtained using the Equation 4.3.

Equation 2.1 is used to calculate the distance between the imputed concept representation (the centroid) and the real concept representation.

4.3 Evaluating the Concept Invention Model

The evaluation of the created concept is a quality evaluation and depends on the semantics embodied by the distributed model of concepts.

Following the aforementioned methodology, a concept space with only C_T is created. The correctness of each invented concept representation $h_{c_i} \in H_{C_T}$ or $d_{c_t} \in D_{C_T}$ cannot be directly compared with true representations of concepts $c_i \in C_i$ because to have a concept space without bias the space cannot contains these representations.

To evaluate the quality of the invented representation the correlation with respect to semantic measures can be computed or alignment with respect to other spaces can be performed.

4.3.1 Correlation with other measures

Once is built, a space embodies a semantics, so each vector expresses a semantic relation concerning the other. For example, distributional models express the relatedness between represented objects, if the representation of *'Footballer'* is created with the model, to evaluate its quality this vector can be compared with the vectors of *'Stadium'*, *'SoccerClub'* and so on. If the DM embodies hierarchical relations the invented representation of *'Footballer'* can be compared with the representations of *'Athlete'* or *'Person'*.

The comparison will generate a similarity measure, to make a *quantitative* evaluation a gold standard can be used.

Given a gold standard composed by triples $< object_1, object_2, measure >$, the presented process can retrieve or create each *object*'s representation and compare the computed similarity and the gold standard similarity.

Ontologies can be used as the gold standard for hierarchical measures, instead, for distributional measures can be used word's similarity dataset based on wordnet like SimLex-999 [40] or WordSim-353 [31].

4.3.2 Alignment with other spaces

Another evaluation technique can be: create a space with all concepts C and making alignment between the training space. Once the process generates the representations for the new concepts, those representations can be projected in the space with all concepts through the alignment and, in this space, a direct comparison can be performed.

This approach may seem good, but alignment between spaces cannot be never perfect, so the alignment will introduce errors in the evaluation. This crucial aspect has to be considered before picking a decision.

Good setup to reduce the alignment error is to use the most of concept for the training and only fewer for the invention. In this way, the alignment would have better quality and so the alignment error will be minimised.

4.4 Biased Representation Spaces of Concepts

The resource partition showed at the beginning of the chapter will produce unbiased concept representations. The usage of unbiased representation is motivated by the fact that if a training concept representation is obtained using concepts that are to invent, the invention process will be eased. To clarify the concept of *unbiased representation*, the presence of bias in representations which embodies different semantics is discussed:

Bias in hierarchic representation space of concepts: if the created space is H_C, even if only representations of $c \in C_T$ are used in the training phase, the training can be biased by some $c \in C_I$. This is because in the input tree the depth of the training concept can be influenced by the others. For example, if the concept of *Fish* is in the training concept and *Animal* has to be invented, in a biased hierarchical space the representation of *Fish* will be deeper than the representation of *Animal* and so, if the creation of *Animal* starting from animal instances is based on the similarity between animals' representations and fish' representations, the correctness of the imputed vector for *Animal* will be influenced by the real position of *Fish* and *Animal* in the biased space.

Bias in distributional representation space of Concepts: in a distributional DM each labelled representation is conditioned by almost all object in the source corpus. This is because when Distributional Semantics is applied in the training of a predictive neural model, the optimised task is to obtain the representation of words in the context of the input concepts. This process is based on all contexts of all concepts because concepts with similar context will have a similar intermediate representation to obtain a similar context during the learning phase. So if a training concept and a concept to invent have a similar context, in a DM obtained using both this concept the two contexts will interact each other and so the obtained representations will be vitiated by this implicit interaction.

Chapter 5

Implementation

In this chapter will be shown an implementation of the model presented in chapter 4. First will be presented the resources and how to build distributed models of these resources after the machine learning model will be presented and the design will be discussed.

5.1 Creation of Distributed Models

This is a first work towards concept invention in distributed models. The focus of this Thesis is to design the task and to provide initial results. So as conceptual spaces will be used biased spaces, generated with a large conceptual knowledge.

5.1.1 Distributional word space

Corpora: To obtain distributed models which embodies distributional relations between words a corpus is needed. A corpus contains the co-occurrence information about words and so is a fundamental resource whenever you want to apply Distributional Semantics over words. The corpora used is the DBpedia Abstract Corpus[1] relative to the 2016 dump of DBpedia. Due to the complexity of the model, only the first 80 thousand rows are used to obtain a distributional word space.

Distributed Representation generation: The distributed model which embodies distributional relation about words is obtained using ELMo [61]. ELMo is a deep learning model which creates a language model during the training on a prediction task. ELMo uses a Bidirectional Language Model

[1]http://downloads.dbpedia.org/2016-10/core-i18n/en/long_abstracts_en.tql.bz2

(BiLM) to predict a word based on its left context (forward model) and on its right context (backward model). Once the BiLM is trained, ELMo's representations are obtained through a concatenation of forward and backward. The ELMo's BiLM has three layers, the *layer 0* contains the word representation, the *layers 1* and *2* contains the contextual representation of the word in its context. A fine-tuning on the target task which lets ELMo learn how to combine the three layers (each layer has a weight) to maximise the performances in the fine-tuning task (which can be another prediction task or another NLP tasks). After the fine-tuning, each layer contains the representations multiplied for the layer weight.

The *pre-training* of the used ELMo model is performed on the *Billion Word Corpus*[2]. The model is then *fine-tuned* to obtain a proper representation of DBpedia Abstract Corpus' language model, and reflect the distributional word's meaning in this corpus. In the obtained model each word occurrence representation is based on his sentence context in the *DBpedia Abstract Corpus* and on all contexts of all occurrences of that word in the *Billion Word Corpus*.

ELMo model produces three vectors for each occurrence of each word in the fine-tuning corpus, in the implementation the *layer 1* is used because the vectors in it embody the contextual representation of the word but even the *layer 2* can be used. Due to the ELMo's hyperparameters, the dimension of a contextual word representation is 1024.

5.1.2 Distributed Concept Spaces

Distributional Concept Space:
Corpora: The distributional concept space is obtained using DBpedia's Types as concepts and are used to process a corpus: starting from the *DBpedia Abstract Corpus* an *entity linking algorithm* is used to obtain the entities contained in the Corpus, unlinked words are eliminated, subsequently each entity is replaced with its *minimal Type*, which is the deeper Type in the *DBpedia Ontology Tree*[3].
Distributed representation generation: The distributed model is obtained by applying Type2vec [6], a Word2Vec [51] model which is trained on the CBOW architecture on the task to predict a word (in this case a type) based on its context. After the training, the representation is contained in the hidden layer. Due to implementation restrictions (the accuracy of the entity linker, the absence of all types in the corpora) the obtained space con-

[2]https://opensource.google.com/projects/lm-benchmark
[3]http://mappings.dbpedia.org/server/ontology/classes/

tains the representations of only 403 DBpedia types where the total number is 760. The dimension of the representation is 100 due to Type2Vec's hyper-parameter.

Hierarchical Concept Space
Input graph: To obtain a hierarchical representation a hierarchical input is required. In this work the *DBpedia Ontology Tree* is used. This structure organises the DBpedia Types in a hierarchic order: the most abstract type represents the concept of *'Thing'*, this node has a very high number of sons. The deeper node in this tree has a depth equal to 7 and so is very full of sub-trees.
Distributed representation generation: The distributed representation of concept which embodies the hierarchical semantics is created with HyperE [66], an algorithm that embeds a graph in a Poincaré Disk Model. The obtained embedding maintains almost exactly the node distances in the graph and has a *low distortion*, meaning that the encoded information is only the ones who were in the input graph. The hierarchical distributed model H_C is obtained by applying the HyperE approach to the DBpedia Ontology Tree. The vectors are 2-dimensional and they embody the hierarchical structure of the Ontology tree. Each labelled vector $h_c \in H_C$ (h_{Thing} excluded) has $Norm(h_t) \ll 1$, this is due to the HyperE approach and is a characteristic which has to be treated in the learning phase because these vectors are very closer to the limit of the Poincaré Disk.

5.2 Multi-Task Learning for Concept Invention

The invention problem can be approached as a Multi-Task Learning (MTL) problem, this choice is due mainly for two reasons: (i) distributional and hierarchical semantics are very different. A combined loss which receives the backpropagation from both spaces can use the information from one space to improve the performance on the other space; (ii) as shown in many cases, in MTL each task works as a regularisation function w.r.t. other tasks.

Figure 5.1 shows the architecture of the network. The loss function optimized by the MTL network is $L = \lambda L_D + (1 - \lambda)L_H$ which is a convex combination of two loss functions which express the distance in the respective spaces:

- L_D is the mean cosine similarity (Equation 5.1) between the predictions and the real vectors for each example e in the training set TS.

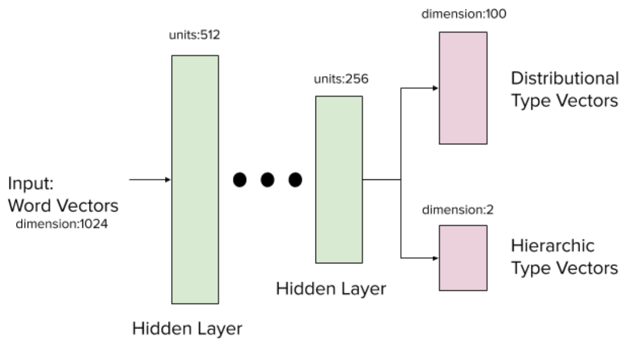

Figure 5.1: Architecture of the multi-task feedforward network

Minimising this generates *distributionally similar vectors* to the real ones because of share a very similar angle w.r.t. the origin.

$$L_D = \frac{\sum_{e \in TS} cossim(pred, true)}{|TS|} \tag{5.1}$$

- L_H (Equation 5.2) is the mean hyperbolic distance between the predictions and the true vectors for each example e in the training set TS with two penalisation terms. By minimise this value, the predictions are forced to be closer to the real vectors, the concept of closeness is based on the Poincaré Disk Model topology.

$$\begin{aligned} L_H &= \frac{\sum_{e \in TS} d_H(pred, true)}{|TS|} \\ d_H(p,t) &= \phi(p) arccosh\left(1 + 2\frac{\|p-t\|^2}{max(K, 1-\|p\|^2)(1-\|t\|^2)}\right) \\ \phi(p) &= \begin{cases} 1 & \text{if } R < \|v\| < 1 \\ P & \text{otherwise} \end{cases} \end{aligned} \tag{5.2}$$

In Equation (5.2) K is a penalisation parameter forcing the predicted vectors to be inside the disk of radius 1, while P is a penalisation parameter that amplifies the distance if the predicted vector is inside the radius R disk ($R < 1$). The former penalisation is imposed to force the predicted vectors to be placed inside the Poincaré disk model domain (inside the unit disk); conversely, the latter penalisation forces the vectors to be near from the edge of the disk, which is the region where the predictions are placed in our dataset.

Chapter 6

Experiments

This chapter will show all the experiments conducted to see if in a facilitating environment (because of the bias) good results can be reached. We will first describe our research questions, then we will explain the experimental setting and finally we discuss the results answering the defined research questions.

6.1 Motivations

In the following section, we are going to answer some research questions that underline the approach we have defined and implemented. We organise the research questions in three different categories and at the end of the experimental section, we will provide a discussion over the answers to these questions.

Distributed Space Analysis

- **Q1:** Do words that belong to the **same** concept share geometrical patterns in distributional representation?
- **Q2:** Do words that belong to **similar** (close in the hierarchy) concepts have similar distributional representation?

Concept Invention Task

- **Q3:** Can distributed models of words be aligned with distributed models of concepts by a mapping function? (e.g., given a region in the word space, can we be able to find a representation for the concept belonged by the word representations which populate that region?)
- **Q4:** Can new concept representations be generated from word representations with the mapping function of Q3?

Machine Learning Specific

- **Q5:** Can different semantics enhance a machine learning task focused on concept representation generation?
- **Q6:** How a space based on hyperbolic geometry interacts with a regression task and classical deep learning approaches? (which are grounded in Euclidean geometry)

6.2 Data set

To create a dataset a set of training classes has to be selected to represent concepts. To make an experiment without introducing complexity the selected classes try to minimise the number of common entity's name (this to avoid the usage of polysemous words), but a complete non-polysemous dataset is really difficult to build.

The selected concepts are the DBpedia types which identify these concepts and are shown in Table 6.1. These concepts will be used to learn how to map the word space to the concept spaces. In the following sub-sections, there will be an analysis of the distributed models with regards to these classes. To select these classes also the distributional concept space is considered because the experimental set uses a space which does not contain all DBpedia Types but only the 53% (403 on 760).

Starting from the classes, tuples are to create. To create tuples the DBpedia endpoint[1] is queried with SPARQL to retrieve the URIs of the entities which belong to these Types. Entities are ordered by the usage of rank based on *PageRank* computed on Wikipedia. This technique is adopted because the cardinality of the entities for each class is very high and so it has been decided to use the 'most popular' entities, under the assumption that if an entity is very popular it will appear a lot in the corpus, so its representations will be truthful. The first 10000 entities in the rank are taken, in Table 6.1 are showed the cardinalities for each population of each class.

If the words contained in the URI for a class is not retrieved, this word is added to the set (for instance the word 'Fish' for the type `dbo:Fish`)

6.2.1 Data cleaning: URI's Processing

Not all URIs will be used in the experiments, this is due to a limit of this model. Since the source space is a word space, each entity has to be represented with a word in the corpus and this is not always possible.

[1]https://dbpedia.org/sparql

Class	DBpedia Type	# Entities
Sport	`dbo:Sport`	1819
Fictional Character	`dbo:FictionalCharacter`	10000
Company	`dbo:Company`	10000
Sports Team	`dbo:SportsTeam`	10000
Award	`dbo:Award`	10000
Colour	`dbo:Colour`	862
Ethnic Group	`dbo:EthnicGroup`	9452
Language	`dbo:Language`	10000
Plant	`dbo:Plant`	10000
Weapon	`dbo:Weapon`	10000
Architectural Structure	`dbo:ArchitecturalStructure`	10000
Planet	`dbo:Planet`	4785
Bird	`dbo:Bird`	10000
Insect	`dbo:Insect`	10000
Mammal	`dbo:Mammal`	10000

Table 6.1: Number of considered Entities for each semantic class

URI normalization: Starting from the retrieved URI, only the prefix is deleted and a lowercase is applied (for example, from `dbr:Goldfish` to *'goldfish'*).

Parenthesis elimination: If a URI contains word between parenthesis, this part is eliminated. (for example, from *'jaguar (band)'* to *'jaguar'*)

Word Phrases: A URI can be composed of multiple words. Since ELMo is used, word phrases are not represented, so, to represent word phrases, only the word phrases with two words are maintained and into ELMo the mean vector between word vectors will be computed.

6.2.2 Creating Tuples and Train, Validation a Test sets

After the processing of URIs, tuples can be created. A tuple is a couple $< processedURI, Type >$. Some `processedURI` appears more than one, this is because the word is polysemous and each meaning is associated with one Type (e.g the word *'Jaguar'* is an animal but also a car brand). To avoid the multilabel problem, for each polysemous word the deeper Type is chosen, if many Types have the same depth, a random Type is chosen. Polysemous

Class	# words	# Compound Words
Sport	116	383
FictionalCharacter	972	1146
Company	1032	2287
Sports Team	23	506
Award	23	438
Colour	41	85
Ethnic Group	539	908
Language	205	1131
Plant	941	554
Weapon	139	374
ArchitecturalStructure	112	1140
Planet	58	89
Bird	176	335
Insect	247	170
Mammal	282	532

Table 6.2: Retrieved words from DBpedia for each class

word in the dataset are 82, after the processing the cardinality of each class is shown in Table 6.3.

To make a properly experiments a best practice with machine learning models is to use three sets: train, test and validation. The train set is used to train the model, the validation set is used during the training to monitor the overfitting and the test set is used to validate the generalisation performed by the model. The division is the following: Train 80% of the words, Val and Test 10% of the words each one.

6.2.3 Retrieving Word Vectors

Not all tuple will be used; in fact, only the tuple who contains words which are in the word vector model will be used. Since the distributed model used to represent words is ELMo, for each word w multiple vectors are labelled with w. This is not a problem because each tuple will be created n triples, where n is the number of occurrences of the word in the tuple. For word phrases, the corpus is searched and only the occurrence vectors in which the two words appear one next to the other are selected. Starting from these couples of vectors, the mean vector (midpoint) is calculated and used as a representation for the word phrase. After all these processes, Table 6.3 shows the cardinalities of each class.

Class	# Tuples	# Vectors
Sport	265	5865
FictionalCharacter	1552	10083
Company	2127	43446
Sports Team	228	1012
Award	151	5163
Colour	60	9207
Ethnic Group	705	8374
Language	490	25465
Plant	1236	18478
Weapon	252	3145
ArchitecturalStructure	674	7342
Planet	83	4859
Bird	282	2320
Insect	248	1151
Mammal	427	11537

Table 6.3: Number of tuples and vectors for each class

6.3 Data Analysis

To make a properly work a data analysis is needed. In this section will be analysed and discussed all three representation spaces.

6.3.1 Word Space Analysis

The word space can be analysed using the selected classes to see how data are disposed of.

Two analysis will be conducted:

- **Visual analysis:** The space can be visualised by applying the PCA.
- **Silhouette Score:** the silhouette score is a quantitative score which shows the clusterization of the dataset; a value closer to 1 indicate a very good clusterization.

Figure 6.1 shows that some classes (e.g. Company) are very well separable but others seems to have a very similar distribution. The silhouette score is computed using the *Euclidean Distance* and the *Cosine Similarity*, the values are: 0.03 with *Euclidean* and 0.11 with *Cosine* so the cluster are not well separated.

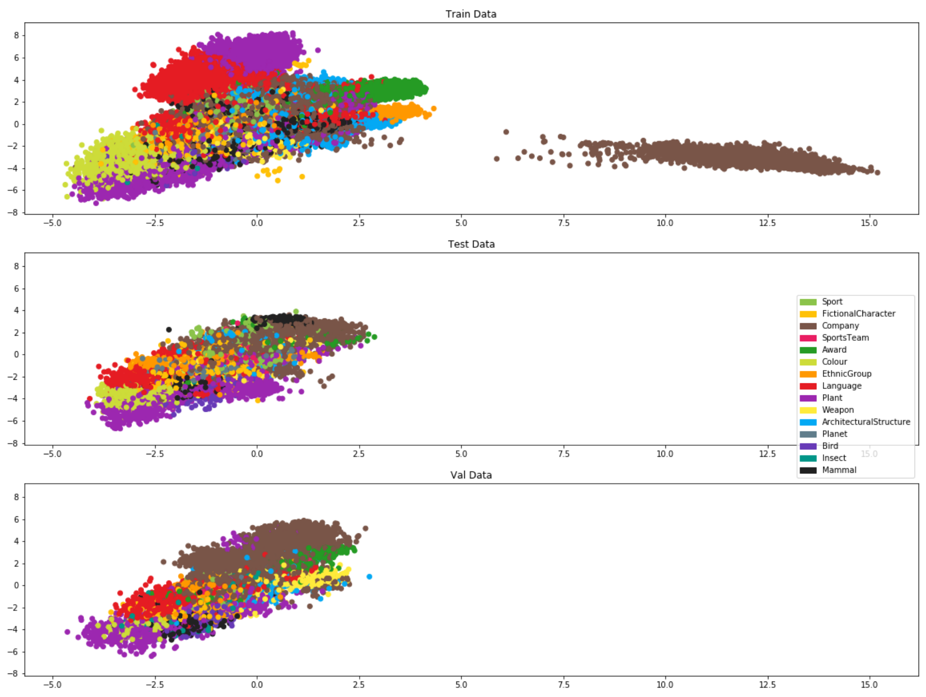

Figure 6.1: PCA of words' datasets which shows the linearly separability of the classes

6.3.2 Concept Spaces Analysis

The analysis of the **distributional** space can be summarised by the visualisation (Figure 6.2).

The analysis of the **hierarchical** space shows that this space in particular, due to the Poincaré Disk Model and the HyperE approach used to build it.

As can be seen in Figure 6.3 all class' points have a norm ≈ 1 (Table 6.4), this is due to the HyperE approach. As can be seen in the tables, some points (e.g *'Mammal'*, *'Bird'*, *'Fish'*) are very similar and this can be a problem during the training. The similar points belong to the same sub-tree and are so similar due to the HyperE approach.

6.4 Invention Dataset

To perform a concept invention process another dataset is created: following the procedure showed before a set of concept to invent is selected and word representations of words which belong to their entities are retrieved (Table 6.5). Based on these words representations the model is used to project the

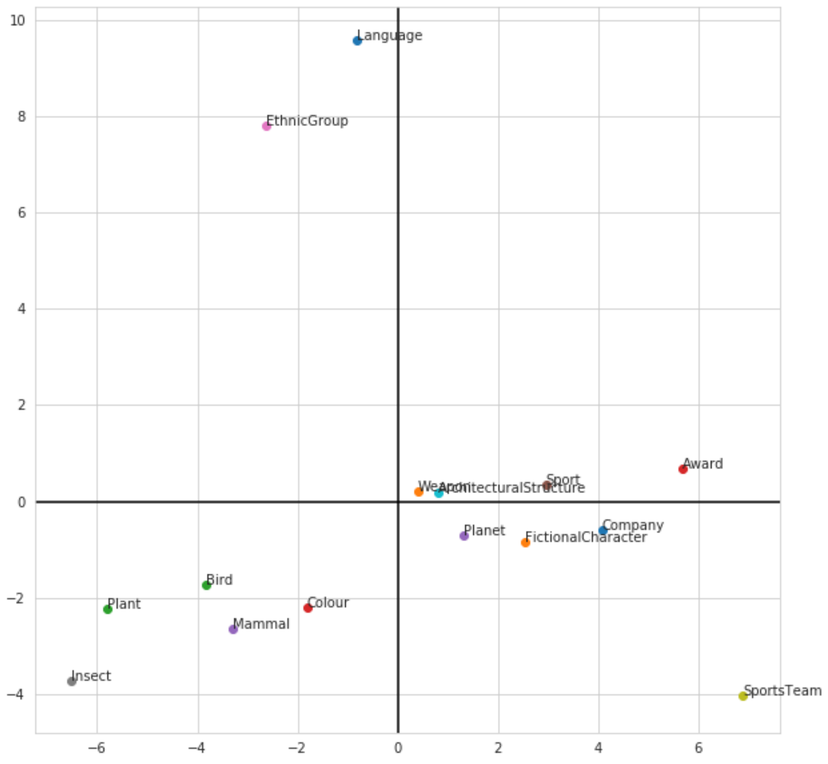

Figure 6.2: Visualisation of the distributional position of the target vectors

word in the concept spaces and the quality is evaluated with the metrics described in the next section.

6.5 Metrics

To evaluate the quality of the prediction similarity ranking between the predicted vectors and the labelled vectors are used. For a given predicted vector the neighbourhood is computed and the position of its true concept label is obtained. Relying on each rank the mean rank of the true concept label inside the ranking list (Avg) is computed and reported. Is also reported the mean position of the true concept label in the neighbourhood of a centroid for each semantic class ($Cent$). Both methods are computed over the distributional

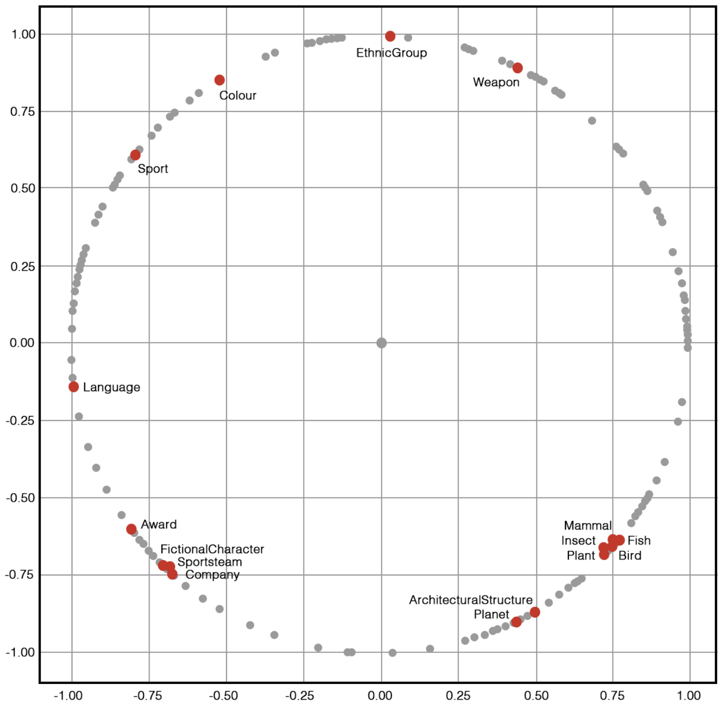

Figure 6.3: Positions of concepts vectors in hyperbolic representation space

Class	X	Y	Norm
Language	- 0.97817	- 0.12357	0.98
EthnicGroup	0.06191	0.98404	0.98
Award	-0.79765	- 0.57953	0.98
Planet	0.45851	-0.88863	0.99
ArchitecturalStructure	0.49143	- 0.86754	0.99
Bird	0.71935	- 0.69462	0.99
Insect	0.71934	-0.69464	0.99
Mammal	0.71932	- 0.69461	0.99
Plant	0.71908	- 0.69499	0.99
FictionalCharacter	- 0.74541	- 0.66653	0.99
SportsTeam	-0.73475	-0.67834	0.99
Company	-0.73426	- 0.67883	0.99
Weapon	-0.45191	0.89145	0.99
Colour	-0.52827	0.83241	0.98

Table 6.4: Target Points in hyperbolic representation space

D and hyperbolic *H* space.
Settings. Two different settings are evaluated: (i) *Cent* and *Avg* of predictions about unseen words which belong to concepts used during training and (ii) *Cent* and *Avg* of predictions about unseen words which belong to unseen concepts. The former will evaluate the invention task as standard in machine learning, so the generalisation ability on trained concepts is tested. The latter is performed to explore how similarity between word representations can be used to infer different concepts, thus being a task closer to real concept invention. One baseline model is considered and two neural networks are proposed.

6.6 Algorithms

Three models are used to perform a quantitative analysis of the invention process:

- Random Forest (**RF**), a random forest regressor model trained to predict the concept vector in the hyperbolic space or in the distributional concept space based on the word vector.
- Feed Forward Neural Network with Single Output (**SO**), a neural network which takes in input the word vector and returns a vector in one

Class	# Vectors
Fish	737
GovernmentAgency	1954
Game	991
Reptile	414
Mollusca	360
Crustacean	140
Amphibian	78
SportsLeague	41
Currency	298
EducationalInstitution	49

Table 6.5: Number of vectors for each inention class

concept space, performing a regression task.

- Multi-Task Neural Network with Double Output (**DO**), the proposed model (called also Multi-Task Neural Concept Inventor) a neural network which takes in input the word vector and returns two vectors: one in the hyperbolic concept space and one in the distributional concept space. This network uses the loss function $L = \lambda L_D + (1-\lambda) L_H$ showed above ($\lambda = 0.91$).

6.6.1 Models Setup

In this section will be described the setup of the models:

RF hyperparameters are the same for both models: `max_depth = 8`, `n_estimator=15`. The choice of these values is due to the complexity and the time required to train a random forest model.

SO hyperparameters are the same for both models: `optimizer = Adam, batch_size = 512, early_stopping on validation loss, patience = 20, epochs=200`. The architecture is the same as the MultiTask Neural Concept Inventor (Figure 5.1), but with a single output.

DO hyperparameters are the same as SO: `optimizer = Adam, batch_size = 512, early_stopping on validation loss, patience = 20, epochs=200`

6.6.2 Loss Hyperparameters

The experiments with the machine learning models have a double objective: the first objective is to learn how to predict inside the circle in the poincaré disk model, this is because outside the circle there is nothing, so a model which predict outside the circle is incorrect; the second objective is to learn how to generalise the regression task in optic to obtain good performances.

Predicting inside the circle

The usage of the poincaré disk model introduces many problems during the implementation. First of all the fact that if the prediction is outside the circle, the hyperbolic distance is not defined and so the gradient and the back-propagation will not work. The simplest loss function that can be used is the *mean of the hyperbolic distance between the prediction and the real vectors* (called *MHD*, Equation 6.1)

$$\begin{aligned} L_H &= \frac{\sum_{e \in TS} d_H(pred, true)}{|TS|} \\ d_H(p,t) &= arccosh\left(1 + 2\frac{\|p-t\|^2}{(1-\|p\|^2)(1-\|t\|^2)}\right) \end{aligned} \tag{6.1}$$

Unfortunately, the usage of this loss causes the presence of NaN (*Not a Number*) during the training. This is because the prediction can be outside and when one point is inside (the true value) and one point is outside the function is not defined. The back-propagation on a Nan generates NaN weights and so the learning fails.

To avoid NaN presence in the training phase the loss has to change. The NaN is caused by the p in the function: when the norm of p is greater than one ($\|p\| \geq 1$) the left part of the denominator becomes negative and so all the fraction becomes negative and the argument of the *arccosh* becomes lesser than 1. The *arccosh* is not defined for number smaller than 1 and generates a NaN. So a modify is required: the left part of the denominator has to be always greater than zero, also for those examples that cause a negative loss. For these examples, the loss has to have a very high value, so the back-propagation can avoid this behaviour for the next predictions.

The loss change:

$$\begin{aligned} L_H &= \frac{\sum_{e \in TS} d_H(pred, true)}{|TS|} \\ d_H(p,t) &= arccosh\left(1 + 2\frac{\|p-t\|^2}{max(K, 1-\|p\|^2)(1-\|t\|^2)}\right) \end{aligned} \tag{6.2}$$

The value of K has to be greater than zero, but lesser than each value that can be assumed by $1 - t$ (a formula which involves the true values, the target vectors) because otherwise when a prediction was about to approach the right value, the loss will penalise this prediction. This is a constraints: $0 < K < 1 - max(\|t\|^2)$. Using this kind of regularisation, the loss is forced to be inside the circle with radius $1 - K$.

The value of K will also affect the total value of the function because with a little value of K (e.g. 10^{-30}) the value will increase by approximately five times.

During the experiments, the value $1 - max(\|t\|^2)$ is 10^{-9} so the K is set to 10^{-15} to have a regularisation, avoiding NaN and avoiding too small K values.

Weight of compound loss

Generally, in a multi-task model, the loss is compound by different losses, in general, one loss per task. The problem is that these losses can take values with a very different range, this is a problem because if a loss value is very greater than another, the gradient will be dominated by the higher value and so the multi-task model becomes a single task because optimising the smaller loss is not convenient enough.

In general having a loss function such as $L = L_1 + L_2$ the more widespread technique is to make a convex combination between the losses through an hyperparameter $0 \leq \lambda \leq 1$. The general weighted loss has this equation: $L = \lambda L_1 + (1 - \lambda)L_2$. In these experiment the value of λ is set to 0.91 so the compound loss became $L = 0.91L_D + 0.09L_H$. This value is chosen due to the values of the two loss functions during the training. L_D is a mean cosine similarity so its value has a range between $[-1, 1]$. To determine the range of L_H a single task network is trained to make a regression from word to hierarchical concept space, during the training the minimum value of the loss is around 10 (while the theoretical limit is 0) so comparing the two optimal values (1 and 10) to have a properly interaction between gradients the weight for L_D has to be 10 times the weight of L_H, so 0.91 and 0.09.

Data-based Regularisation

The model with the loss described above has the desired behaviour, so each prediction is inside the circle. But there is another problem: the using of K induce the model to not predict points close to the limit, because the penalisation is very high even if a reduced number of prediction is outside the circle; the target points instead are very closer to the limit of the circle (as shown in Table 6.4) so another regularisation is needed to force the prediction's norm to be as higher as possible (and in any case lesser than 1).

This regularisation has to work when points have norm lesser than a certain value, this value can be seen as a circle centred in the origin inside the poincaré disk from which prediction has to be pushed out. So two hyperparameters have to be established experimentally: the radius R of the inside circle and the penalisation P. During the experiments the optimal value were: $R = 0.8$ and $P = 4$.

The final loss function with the experimental hyperparameters is:

$$\begin{aligned} L &= 0.91 L_D + 0.09 L_H \\ L_H &= \frac{\sum_{e \in TS} d_H(pred, true)}{|TS|} \\ d_H(p,t) &= \phi(p) arccosh\left(1 + 2\frac{\|p-t\|^2}{max(10^{-9}, 1 - \|p\|^2)(1 - \|t\|^2)}\right) \\ \phi(p) &= \begin{cases} 1 & \text{if } 0.8 < \|v\| < 1 \\ 4 & \text{otherwise} \end{cases} \end{aligned} \tag{6.3}$$

6.7 Multi-Task vs Single-Task

So far the usage of a Multi-task network is motivated in this thesis with theoretical assertion and state of the art analysis, but in the machine learning domain, each problem and each dataset is unique and does not exist a single right way but a multitude of best practice. So to test the effective utility of the multi-task network in this domain a first experiment is performed.

To test the performances of an architecture with respect to others the problem is simplified. The three architecture (random forest regressor, single network and multi-task network) are tested with a similar setup: the loss function will be the *'mean squared error' (mse)* so the weight problem through different loss expires and the poincaré topology is set aside.

The models are evaluated as a classic regression problem, so the *mse* between the predictions and the true points is used to evaluate the correctness of the prediction. The hyperbolic predictions are treated as normal regression prediction, so they are accepted even if they are outside the Poincaré Disk Model. As can be seen in Table 6.6 the multi-task model has a better chance to make a correct generalisation because of the interaction of the two semantics.

Model	Avg D	*Cent D*	Avg H	*Cent H*
RF	108	64	**16**	234
SO	106	30	23	6
DO	**99**	**25**	20	**2**

Table 6.6: Predictions ranks on unseen words which belong to known concepts with mse as loss function

6.8 Results

In this Section, the results of a *Concept Invention* task performed by the models are presented. Example of predictions are shown and numerical results are provided to support the evaluation of the entire process. This section aims to provide some first evidence on the capabilities of the *MTNCI* to *invent* concepts. Experiments will be generally based on the following task: generating concept representations of unseen words (words not in the training set) and to check how close they are to the vector of their real concept. The intuition can be provided via an example: the predicted vector of the word *'beaver'* will be closer to the vector of the concept *Mammal* because the representations of the words which belong to *Mammal* are similar to the representation of *'beaver'*.

For each class, $c \in C_I$ multiple vectors are predicted, so h_c and d_c have to be imputed. Generating predictions from multiple word vectors V_w of words w which belong to the concept c is a good process: a generalisation of concept *'Fish'* has a better quality when it is based on different instances of fish.

6.8.1 Qualitative Analysis

To evaluate the quality of a concept invention process some words which belong to entities which in DBpedia2016 were *instanceOf Thing* are selected. These words are chosen because MTNCI can be used to improve the quality of the representation of these entities. The first positions of the rank of the similarities between the centroid of the predicted vectors and the labelled vectors in the concept spaces are reported. Table 6.7 shows some examples of predictions.

6.8.2 Quantitative Analysis

Table 6.8 shows that the neural model is able to correctly project a centroid into the distributional concept space and, on average, the projected vectors

Word	Ranking in D_C	Ranking in H_C
Algae	Plant	Thing
	Conifer	Species
	Fern	ChemicalSubstance
	Insect	Polyhedron
Beaver	Mammal	Species
	Bird	ChemicalSubstance
	Reptile	Thing
	Fish	Polyhedron
JPEG	Company	Agent
	Bank	Award
	Automobile	Thing
	Software	PublicService

Table 6.7: Qualitative results on word of unseen concepts

into the hierarchical concept space have a correct position. The quantitative results show that the distributional projection is probably noisy, show also that the hyperbolic centroid procedure is highly affected by 'wrong' predictions. Results suggest that while the performance of the models is still far from one of a perfect predictor, the approach is promising; moreover the qualitative predictions (Table 6.7) seem to support this mapping methodology. The neural models seem to be better suited for this task then the baseline. Still DO suffers from some limitations that might be due to the optimisation process; nevertheless, its ability to learn from two different spaces and aggregate the information is important for novel tasks related to concept invention; results are heavily influenced by the hyper-parameters of the MTL model (e.g., λ).

Model	Avg D	*Cent* D	Avg H	*Cent* H
RF	108	64	**16**	234
SO	**85**	**4**	17	208
DO	115	5	18	**206**

Table 6.8: Predictions ranks on unseen words but with known concepts

Model	Avg D	*Cent D*	Avg H	*Cent H*
RF	181	107	**18**	372
SO	**175**	**82**	35	**330**
DO	183	**82**	31	333

Table 6.9: Prediction ranks with unseen words and concepts

6.9 Discussion

The results show that neural models are better than the baseline, when the correct losses are used the multi-task network results similar to the single-task, this is probably because despite the weight λ, which is chosen based on the best case of the loss, is not enough when the internal regularisation (ϕ) is active. Moreover, as discussed during section 6.6.2, a loss function can force neural models to project under different kind of constriction (in this case project inside the poincaré disk), this is a very important feature of neural models.

Other considerations that can be deduced from the results are: the predictions in the distributional concept space are noisy, this can be seen by the comparison between *Avg D* and *Cent D*. Another consideration is about the centroid in the hyperbolic space, although the process is correct, is highly influenced by the wrong prediction and so, in a hierarchical way, the centroid represents a very abstract object in the hierarchical structure. The results of RF are the same shown in Table 6.6 because RF uses the *mse* to optimise the learning.

6.10 Summary of the Research Questions

Distributed Space Analysis

- **Q1:** Do words that belong to the **same** concept share patterns in distributional representation?
- **Q2:** Do words that belong to **similar** (close in the hierarchy) concepts have similar distributional representation?

Data analysis of words representation in training dataset answer to Q1, words which meaning belongs to the same concept are clustered in the distributional word representation space. Analysis on Invention task suggest that if similar results are reached both on training concept and invention concept, probably the generalisation and the mapping learnt on the training data works in a similar way even with new input distributions of words representations which belong to new concepts, so the similarity between words which belong to similar concepts seems to be enough to perform a good projection.

Concept Invention Task

- **Q3:** Can distributed models of words be aligned with distributed model of concepts by a mapping function? (e.g., given a region in the word space, can we be able to find a representation for the concept belonged by the word representations which that populate this region?)
- **Q4:** Can new concept representations be generated from word representations with the mapping function of Q3?

Different machine learning models show that given a word-concept alignment a function which is able to map from word space to concept space can be learnt with good performances on the regression task. Results are produced using this function, it seems that more specific models (which uses hyper-parameter and loss function designed on this task) produce better results, so the concept generation problem can be enhanced by improving machine learning models quality.

Machine Learning Specific

- **Q5:** Can different semantics enhance a machine learning task focused on concept representation generation?
- **Q6:** How a space based on hyperbolic geometry interacts with a regression task and classical deep learning approaches? (which are grounded in Euclidean geometry)

The usage of combined loss function has enhanced the generalisation process, the usage of a particular geometric space was, of course, a problem to be faced, but its special topological characteristics are fundamental to allow the usage of hierarchic relations. The Euclidean version of the gradient, the one generally applied with NN, shows stability problems when has to correct predictions in Poincaré Disk Space. A properly regularisation helps a lot to hold the space constriction.

Additional studies and experiments on the interaction between different geometric models in NN surely will enhance the performances in Concept Invention on distributed models because the main problem faced in this thesis was to understand how to use (and bring) the hyperbolic space.

Chapter 7

Conclusions and Future Work

In this thesis is proposed a first approach to tackle concept invention combining information coming from distributed representations. While the results are preliminary, the experiments show that this approach is promising.

The future planned work are multiple: first the results are to validate, so other data sets are needed to see if results are similar to that presented in this thesis; second the loss' hyperparameters are to be studied, the hyperparameter optimisation in this thesis is made without theoretical model, using Bayesian optimisation or Gaussian Processes can help to enhance the performance. The architecture of the model can change and experiments in this way will be conducted.

Moreover, it will be a study on new semantic spaces for concepts, because using two spaces or use three, four or more is an implementation problem but theoretically will boost up the potential of the invention process.

Other studies can be conducted: a reverse mapping, from concept representations to word, can be performed to find the regions in which the words which belong to a concept are probably to find a position. This process can be implemented with a multi-modal network, which is a network with multiple inputs (in this problem the concept spaces).

A concept of space exploration can be performed in this way: given a concept representation v, the mapping function can be seen as a function which has to minimise the distance with v, so the problem will be faced through an exploration of the word space, using the word vectors in a Bayesian optimisation problem to find the minimisation of the function.

Also, the Concept Blending can be inspected in two ways: starting from concept spaces, techniques to combine concept representations to obtain blended concept can be studied in deep, also techniques to combine words to obtain representations which belong to the blended concept can be studied.

The multi-space setup can be also used to evaluate different representation spaces: given a trained model, new word spaces can be added to evaluate if the new information contributes to having a better invention process. For example, if the new word space is only a rotation of the former, probably the performances will not change and so the added space will results useless in this task.

All these future works will start from this thesis and from the paper reported in the Appendix, which was submitted to AIIA 2019[1] on June 2019.

[1]https://aiia2019.mat.unical.it/home

Bibliography

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, pages 722–735, 2007.

[2] Marco Baroni and Alessandro Lenci. Concepts and properties in word spaces. *Italian Journal of Linguistics*, 20(1):55–88, 2008.

[3] Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42(4):763–808, 2016.

[4] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, 1999.

[5] Tarek Richard Besold and Enric Plaza. Generalize and blend: Concept blending based on generalization, analogy, and amalgams. In *ICCC*, pages 150–157, 2015.

[6] Federico Bianchi and Matteo Palmonari. Joint learning of entity and type embeddings for analogical reasoning with entities. In *NL4AI@AI*IA*, pages 57–68, 2017.

[7] Chris Biemann. Ontology learning from text: A survey of methods. In *LDV forum*, volume 20, pages 75–93, 2005.

[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *NIPS*, pages 601–608, 2001.

[9] Margaret A Boden. *The creative mind: Myths and mechanisms.* Routledge, 2004.

[10] Margaret A Boden. Computer models of creativity. *AI Magazine*, 30(3):23–23, 2009.

[11] Mikael Boden. A guide to recurrent neural networks and backpropagation. *the Dallas project*, 2002.

[12] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[13] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2787–2795, 2013.

[14] Zied Bouraoui and Steven Schockaert. Automated rule base completion as bayesian concept induction. In *AAAI*, 2019.

[15] Line Brandt and Per Aage Brandt. Making sense of a blend: A cognitive-semiotic approach to metaphor. *Annual Review of Cognitive Linguistics*, 3(1):216–249, 2005.

[16] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. Dl-learner—a framework for inductive learning on the semantic web. *Journal of Web Semantics*, 39:15–24, 2016.

[17] Hiram Calvo, Oscar Méndez, and Marco A Moreno-Armendáriz. Integrated concept blending with vector space models. *Computer Speech & Language*, 40:79–96, 2016.

[18] John Bissell Carroll. Words, meanings and concepts (i). In *Readings in Educational Psychology*, pages 33–95. Routledge, 2012.

[19] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[20] Simon Colton and Geraint A. Wiggins. Computational creativity: The final frontier? In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, pages 21–26, 2012.

[21] Roberto Confalonieri, Tarek Besold, Mihai Codescu, and Manfred Eppe. Enabling technologies for concept invention. In *Concept Invention*, pages 189–219. Springer, 2018.

[22] Roberto Confalonieri, Manfred Eppe, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, and Enric Plaza. Upward refinement operators for conceptual blending in the description logic $e\ l^{++}$. *Ann. Math. Artif. Intell.*, 82(1-3):69–99, 2018.

[23] Roberto Confalonieri, Marco Schorlemmer, Oliver Kutz, Rafael Peñaloza, Enric Plaza, and Manfred Eppe. Conceptual blending in EL++. In *Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, April 22-25, 2016.*, 2016.

[24] Roberto Confalonieri, Marco Schorlemmer, Enric Plaza, Manfred Eppe, Oliver Kutz, and Rafael Peñaloza. Upward refinement for conceptual blending in description logic: An asp-based approach and case study in EL++. In *Proceedings of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25-27, 2015.*, 2015.

[25] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI magazine*, 14(1):17–17, 1993.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[27] Valerio Di Carlo, Federico Bianchi, and Matteo Palmonari. Training temporal word embeddings with a compass. In *AAAI (to appear)*, 2019.

[28] Manfred Eppe, Ewen Maclean, Roberto Confalonieri, Oliver Kutz, Marco Schorlemmer, Enric Plaza, and Kai-Uwe Kühnberger. A computational framework for conceptual blending. *Artif. Intell.*, 256:105–129, 2018.

[29] Manfred Eppe, Ewen Maclean, Roberto Confalonieri, Oliver Kutz, W. Marco Schorlemmer, and Enric Plaza. Asp, amalgamation, and the conceptual blending workflow. In *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, pages 309–316, 2015.

[30] Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

[31] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, 2002.

[32] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. *CoRR*, abs/1804.01882, 2018.

[33] Dan Garrette, Katrin Erk, and Raymond Mooney. A formal approach to linking logical form and vector-space lexical semantics. In *Computing meaning*, pages 27–48. Springer, 2014.

[34] Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer, editors, *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 107–114. The Association for Computer Linguistics, 2005.

[35] Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *CoRR*, abs/1301.3485, 2013.

[36] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[37] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[38] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[39] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[40] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[41] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[42] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. *Proceedings of ACL-08: HLT*, pages 1048–1056, 2008.

[43] Matt Le, Stephen Roller, Laetitia Papaxanthos, Douwe Kiela, and Maximilian Nickel. Inferring concept hierarchies from text corpora via hyperbolic embeddings. *CoRR*, abs/1902.00913, 2019.

[44] Alessandro Lenci. Distributional semantics in linguistic and cognitive research. 2009.

[45] Boyang Li, Alexander Zook, Nicholas Davis, and Mark O Riedl. Goal-driven conceptual blending: A computational approach for creativity. In *Proceedings of the 2012 International Conference on Computational Creativity, Dublin, Ireland*, pages 3–16, 2012.

[46] Jiaqing Liang, Yi Zhang, Yanghua Xiao, Haixun Wang, Wei Wang, and Pinpin Zhu. On the transitivity of hypernym-hyponym relations in data-driven lexical taxonomies. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[47] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[48] David McDonald and Shan He. Heat: Hyperbolic embedding of attributed networks. 2019.

[49] Kevin Meijer, Flavius Frasincar, and Frederik Hogenboom. A semantic approach for extracting domain taxonomies from text. *Decision Support Systems*, 62:78–93, 2014.

[50] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[51] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[52] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[53] Roberto Navigli and Paola Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30(2):151–179, 2004.

[54] Katherine Nelson. Concept, word, and sentence: interrelations in acquisition and development. *Psychological review*, 81(4):267, 1974.

[55] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *NAACL*, pages 460–466. The Association for Computational Linguistics, 2016.

[56] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *CoRR*, abs/1705.08039, 2017.

[57] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 809–816, USA, 2011. Omnipress.

[58] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 271–280, New York, NY, USA, 2012. ACM.

[59] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *EMNLP*, pages 1532–1543. ACL, 2014.

[60] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014.

[61] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *NAACL*, pages 2227–2237. ACL, 2018.

[62] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. 2013.

[63] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[64] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017.

[65] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[66] Frederic Sala, Christopher De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4457–4466. PMLR, 2018.

[67] Sunita Sarawagi et al. Information extraction. *Foundations and Trends® in Databases*, 1(3):261–377, 2008.

[68] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011.

[69] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[70] Lawrence Shapiro. *Embodied cognition*. Routledge, 2010.

[71] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. A graph-based approach for ontology population with named entities. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 345–354. ACM, 2012.

[72] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics, 2006.

[73] Yangqiu Song, Shixia Liu, Xueqing Liu, and Haixun Wang. Automatic taxonomy construction from keywords via scalable bayesian rose trees. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1861–1874, 2015.

[74] Gustav Šourek, Suresh Manandhar, Filip Železnỳ, Steven Schockaert, and Ondřej Kuželka. Learning predictive categories using lifted relational neural networks. In *International Conference on Inductive Logic Programming*, pages 108–119. Springer, 2016.

[75] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[76] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th International Conference on the World Wide Web*, pages 697–706, 2007.

[77] Ryota Suzuki, Ryusuke Takahama, and Shun Onoda. Hyperbolic disk embeddings for directed acyclic graphs, 2019.

[78] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. *CoRR*, abs/1810.06546, 2018.

[79] Johanna Völker and Mathias Niepert. Statistical schema induction. In *Extended Semantic Web Conference*, pages 124–138. Springer, 2011.

[80] Denny Vrandecic. Wikidata: a new platform for collaborative data collection. In *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 1063–1064, 2012.

[81] Chengyu Wang, Xiaofeng He, and Aoying Zhou. A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In *EMNLP*, pages 1190–1203. Association for Computational Linguistics, 2017.

[82] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234, New York, NY, USA, 2016. ACM.

[83] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743, 2017.

[84] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[85] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *EMNLP*, 2013.

[86] Geraint A Wiggins. Searching for computational creativity. *New Generation Computing*, 24(3):209–222, 2006.

[87] Ping Xiao, Hannu Toivonen, Oskar Gross, Amílcar Cardoso, João Correia, Penousal Machado, Pedro Martins, Hugo Goncalo Oliveira, Rahul Sharma, Alexandre Miguel Pinto, et al. Conceptual representations for computational concept creation. *ACM Computing Surveys (CSUR)*, 52(1):9, 2019.

[88] Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2701–2709. ACM, 2018.

Appendix A

In this appendix is attached the paper submitted on June 2019 to the conference AIIA 2019[2]

[2]https://aiia2019.mat.unical.it/home

Mapping Lexical Knowledge to Distributed Models for Ontology Concept Invention

Manuel Vimercati, Federico Bianchi, Mauricio Soto, and Matteo Palmonari

University of Milan-Bicocca, Milan, Italy
`m.vimercati15@campus.unimib.it`,
{`federico.bianchi, mauricio.sotogomez, matteo.palmonari`}`@unimib.it`

Abstract. Ontologies are largely used but, due to the complex abstraction processed required to create them, are often incomplete. Concept invention offers a valid solution to extending ontologies by creating novel and valuable concepts starting from previous knowledge. Recently, encoding knowledge using distributed vector representations has become a popular method in both NLP and Knowledge Representation. In this paper, we show how concept invention can be complemented with distributed representation models to perform ontology completion tasks starting from lexical knowledge. We propose a first approach based on a deep neural network trained over distributed representations of words and ontological concept. With this model, we devise a method to generate distributed representations for novel and unseen concepts and we introduce a methodology to evaluate these representations. Experiments show that our model despite some limitations is a promising tool for concept invention.

1 Introduction

Ontologies define the terminology adopted to represent structured knowledge in a variety of application domains. Ontology concepts specify which types of entities are described in knowledge graphs, eventually specifying their meaning using axioms from a formal language. Large knowledge graphs require to cover entities of different types and their ontologies are often incomplete because of the complex abstraction processes required to model ontologies, where a subset of most important concepts are usually selected, and the intrinsic dynamic nature of these processes. For example, the DBpedia ontology includes concepts such as *'Guitarist'* and *'Singer'* but does not include concepts such as *'Pianist'* and *'DJ'*, despite entities that are specifically pianist and DJs are described therein.

The task of adding more concepts to a given ontology can be viewed as an ontology completion problem. Ontology completion can be interpreted under different perspectives. For instance, completion can be interpreted as adding *subclass of* relations between concepts that exist. Another possible completion mechanism is concept invention, where a new concept is created in the ontology [9]. This can be achieved by blending, where new concepts are obtained using schema-level evidence, that is, by combining existing concepts from the ontology, or by induction, where new concepts deemed to be relevant, are generated using evidence about instances of ontology concepts [9].

The most common approach to specifying the meaning of ontology concepts is by means of axioms of formal languages like RDFS and OWL, which are grounded in formal logic or grounded in rule-based semantics. However, recently, there has been significant interest in *Distributed Models* (DMs) as alternative approaches to represent the semantics of knowledge graph elements, included ontology concepts. DMs have become very popular in computational linguistics, where some model implementations like Word2Vec [23] and ELMo [27] have become standard techniques in downstream NLP applications [27], and arguments for their cognitive groundedness has been maintained [22]. Those models provide lexical knowledge starting from natural language texts and are based on Distributional Semantics, whose main hypotheses are that words' meaning is based on their usage and that words that appear in similar contexts are similar. For example, the representations of *'Footballer'* and *'Stadium'* will be similar because those words are used in similar contexts. Since several approaches have proposed models for learning distributed representations of entities and properties [4], few approaches to learn distributed representations of ontology concepts have also been proposed. For example, Type2Vec proposes concept representations based on distributional semantics that encode similarity and/or relatedness between the concepts [2]. Other approaches have been proposed to encode the *subclass of* relation and concept hierarchies [25, 21, 29]. These models use Hyperbolic Geometries that are better suited to capture tree-like structures[20].

In this work, we want to show that *DMs* can provide natural and suitable frameworks to support novel ontology completion tasks that require concept invention. Distributed models are vector-based representations of entities or properties or concepts that are generated from data through machine learning approaches that optimize one or more functions. As a result of this optimization process, while the individual components of the vectors may not be interpretable, the vector space encodes different relations between the represented elements, such as similarity, subclass relations, etc.

The key intuition behind the use of distributed models for the concept creation tasks is based on the following observations: each one of the infinite set of vectors that constitute a vector space representing concepts encodes some semantics; vectors that are not associated with a known concept, may be in fact potential new concepts in the ontology; relevant new concepts can be identified by finding some *hot spots* in this space, based on some kind of evidence. For example, we expect to be able to find a suitable position for the unseen concept *'DJ'*, by using some kind of available evidence.

In this paper, we propose a first approach towards supporting concept inductions where we use lexical knowledge as a source of evidence. In particular, we learn a function that aligns distributed word representations to distributed concept representations. As a source of the alignment, we use distributed representations based on the ELMo model, where more vectors are associated with each word to reflect different meanings that one word assumes in different contexts.

As target of the alignment, we consider two different distributed concept spaces: a hyperbolic space, which represents the hierarchical relations between concepts, and a Euclidean space, which is built with Type2Vec and encodes distributional semantics. This twofold representation allows us to capture simultaneously both, the semantic similarity derived from the use of the concepts in the data and the relatedness of the concepts obtained from an ontology. Intuitively, the function learns how to map concept labels to the concept representations, and, more importantly in the training process, entity names to the representations of their respective concepts. For example, we learn to generate the representation of the concept *'Guitarist'* by mapping guitarists' names into the representation of this concept in the two concept space. As a consequence, we can then feed to the networks several names of pianists, to generate the representation of the concept *'Pianist'* in the two spaces.

The main contributions of this paper are: (i) we illustrate how models can be used to represent concepts; (ii) we provide a model which can be used to obtain data set without bias; (iii) we show how new concept representation can be generated using the semantics inside distributed models; (iv) we propose evaluation methods and discuss about problems of them; (v) we provide a first example of implementation and result of concept invention in our models.

The paper structure is the following: in Section 2 we fix terminology and notation while in Section 3 we summarize related approaches in the concept invention and ontology completion fields; in Section 4 we show how tackle concept invention with distributed models and we provide an implementation of our approach; in section 5 we show the methodology to evaluate the invention process and talk about the experimental evaluation. We end the paper in Section 6 with conclusions and future work.

2 Preliminaries

A Knowledge Graph (KG) is a structure used to describe knowledge about a particular domain. Formally a KG is as a tuple $< C, E, \subseteq, \in >$, where:

- C is a set of concepts (often also referred to as classes), e.g., *Pianist*;
- E is a set of entities that represent domain objects, e.g., *Elton_John*;
- $\subseteq$ is a *subclassOf* relation that holds between pairs of concepts in C; we assume that the relation is transitive, reflexive and anti-symmetric and defines a partial order over the set of concepts, e.g., $Pianist \subseteq MusicalArtist$;
- $\in$ is the *instance-of* relation that holds between an instance and a concept, e.g., $Elton\_John \in Pianist$

Distributed Models (DMs), frequently named Embeddings, are vector-based models which are used to represent different knowledge resources, such as word meanings and ontology concepts. A distributed model is a vector space $\mathbb{R}^n$ which uses vectors in the space to represent the objects. The semantic of the represented objects is transmitted by geometrical or mathematical relations with labelled vectors in the space. Distributed models are generated by training a model with a data set. For example, word embeddings are built using a text corpus. Vector spaces used in distribution models contain , by definition, an infinite set of vectors, even though only a few of them are explicitly associated with an element after training, usually, the vectors that correspond to some element in the training data, e.g., words in the corpus. We will refer to these vectors as labelled vectors, because they are explicitly associated with a label, e.g., the word they are associated with, and to all the other vectors as unlabelled vectors. The dimension n of the space depends on the method used to create the DR. A DR can be built to encode different kinds of semantics, in this paper we consider two main kinds of semantics as further detailed below.

– Distributional semantics (DS) [18] is a semantic principle according to which the distribution of words in a corpus, i.e., their usage, determines their meaning. Different count-based [16, 10, 13, 26] or predictive [23, 27, 11] approaches can be used to generate distributed models inspired by the DS principle. All these approaches return distributed models such that we can expect that vectors of similar words will appear close in the vector space. In fact, DS encodes relatedness among words, where the relatedness can be computed using cosine similarity or Euclidean distance. DS has been used to account for word meaning extensively and in a number of models such as Word2Vec [23], Glove [26] and BERT [11]. In particular, we will use ELMo[27] as input word space in our approach. However recent work has also proposed to use DS as a principle to provide distributed representations of entities and concepts of knowledge graph. Type2Vec [2] provides a way to embed DBpedia types in the vector space by considering type to type co-occurrence over a text corpus: the corpus is firstly annotated with entities (using entity linking techniques) and then entities are replaced with their most specific type thus generating a document containing ordered sequences of types.
– **Hierarchical Semantics (HS):** we use HS to express hierarchical relations between *concepts*. Example of structures used to show hierarchical relations are the Taxonomies. Recent methods [25, 21, 29] use Hyperbolic Geometry Models to create an high quality representation since this hyperbolic geometry is well-suited for hierarchical structures[20]. The quality is commonly measured by the evaluation of the reconstruction of the original resource starting from the DR or by link prediction task [25] which asks to predict edges which where ad hoc removed from the input tree before the embedding phase.

Slightly abusing the terminology, from now on we will refer to distributional models (or representations), to refer to distributed models (representations) inspired by DS. Based on this distinction we define here the three distributed representation spaces that we will use in our approach: a distributional representation space of words, a hyperbolic representation space of ontology concepts, and a distributional representation space of concepts.

The most common Hyperbolic models used to create graph embedding are Lorentz Model (which is the upper sheet S^+ of the Hyperboloid model) and

Poincaré Disk Model (Figure 1) which is the choice in our model. The Poincaré Disk Model correponds to a two-dimentional disk of radius r endowed with its natural hyperbolic metric(Equation 1). In this model, distance between points scales exponentially when we move from the origin. This property makes this model naturally suitable to represent tree-like structures, in which the number of elements usually grows exponential with depth. In fact, trees can be isometrically embedded using this model [17].

$$d(x,y) = arccosh\left(1 + \frac{2*r^2*\|x-y\|^2}{(r^2-\|x\|^2)*(r^2-\|y\|^2)}\right) \tag{1}$$

A Distributional representation space of words V_W (Figure 2) can be built over a tuple $< T, W >$. V_W is a distributed model which is built by applying the DS over a generic corpus T. A corpus T is a collection of sentences, a sentence is an ordered set of words $w \in W$ and word phrases (like *'New York Times'*) $wp \in W$. Depending on the applied model, a labelled vector $v_w \in V_W$ exists for each word $w \in W$ (Word2Vec [23] or GLOVE [26]) or for each occurrence of each word $w \in W$ (ELMo [27] and BERT [11]).

A Hierarchical representation space of concepts H_C can be built over a tuple $< C, \subseteq >$. H_C is a distributed model of the concepts C in the POSet structure of a KG. For example, in [29] for each $c \in C$ a labelled vector $h_c \in H_C$ is created with the Poincaré disk model.

A Distributional representation space of concepts D_C (Figure 3) can be built over a tuple $< C, \varepsilon, \in, T_C, \lambda, KG >$. D_C is a distributed model built applying the DS over a corpus T_C. T_C is obtained by replacing each entity ε in a generic corpus with $\lambda(\varepsilon)$, λ is a function which represents an *Entity Linker* which for each ε returns c if in KG $\varepsilon \in c$ is true, in the case that ε is not in the KG nothing is returned. So T_C is a collection of sentences composed by ordered $c \in C$. To obtain D_C a *DS based* method is applied over T_C.

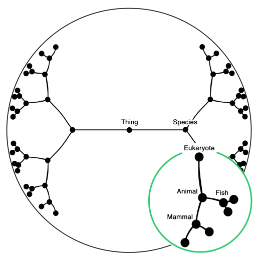

Fig. 1. Example of Tree Embedding in a Poincaré Disk Model

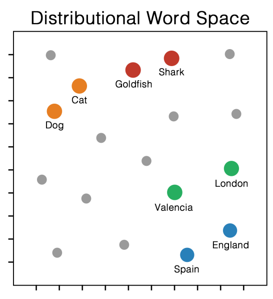

Fig. 2. Example of Distributional DR of words

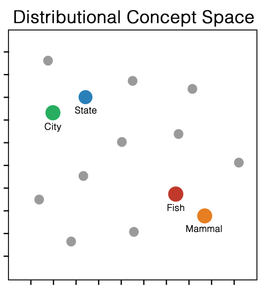

Fig. 3. Example of Distributional DR of concepts

3 Related Work

Inspiring from Computational Creativity research field [8, 33], we divided concept creation in this four tasks [34]:

- *Concept Extraction* is the task of extracting and transforming a conceptual representation from an existing but different representation of the same idea; this research field is related to the information extraction methods, for example, Hearts patterns [19] which are able to extract Hyponym/Hypernym relations between words from text corpora;
- *Concept Induction* is based on *instances of concepts* from which a concept or concepts are learned; this can be supervised (Concept Learning) or unsupervised (Concept Discovery). An example of concept learning is the task of automatic taxonomy construction from NLP resources like the approaches in [32] which use NLP techniques to learn a generalized version of concepts and creates a structure which organizes them in hierarchic order. In Concept Discovery concepts are obtained by applying unsupervised techniques like clustering or LDA [3];
- *Concept Recycling* is the creative re-use of existing concepts; like the study of semantic shift in temporal word embedding [12] or the Concept Blending field [15, 14];
- *Concept Space Exploration* takes as input a search space of possible new concept and locates interesting concepts in it.

The Ontology Completion task can be faced with two strategies: by finding new triples or by extending the set of concepts and defining new *subclassOf* relations. The former can use KG embeddings which encode entities as vectors and properties as spatial transformations to predict new triples [4, 24], recent methods combines *rule templates* with DR created using gaussian distributions [5]; the latter derive new concepts and rules starting from the Ontology itself. These approaches can be designed with *Inductive Logic Programming* [6] or even with *Statistical Relational Learning* [31, 30]

Recently [21] uses text corpora to infer hierarchical concepts. This work is based on an hyperbolic embedding and a revisited version of Hearst patterns, which are lexical patterns used to find Hyponym/Hypernym relations in a text. (for example in the sentence *'cat and other animals'* 'cat' can be recognized as an instance of 'animals').

Our work applies the Concept Induction definition to generate the representations for new concepts.

4 Ontology Concept Induction with Lexical Knowledge Mapping

We first provide an overview of the proposed approach and then we discuss in detail each of its components.

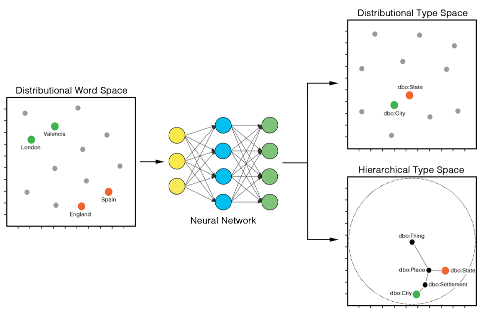

Fig. 4. Multi-Task learning for concept invention schema

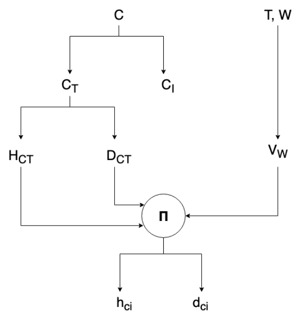

Fig. 5. Schema of resource usage

4.1 Approach Overview

We define the concept invention as a mapping function $\Pi : \mathbb{R}^n \rightarrow \mathbb{R}^m \times \mathbb{R}^q$ from word space to two concept spaces. We model Π as a Neural Network (Figure 4) which is trained in a Multi-Task learning (MTL [7]) setup, but any machine learning model can be also applied.

Starting from a set of concepts C, create $C_T \subset C$ and $C_I \subset C$, with $C_T \cap C_I = \emptyset$. C_T represents the set of training concept and will be used to learn Π, C_I is the set of invention concepts and their representation and will be created using Π.

Afterwards, a hierarchical representation space H_{C_T} and a distributional representation space D_{C_T} are created using resources which contain **only** concept $c \in C_T$. A corpus T is used to create a distributional space V_W of words W. A schema to which illustrate this procedure is showed in Figure 5.

In our work, we want to perform concept invention starting from words. This can be seen as a *Concept Induction* task. To learn a mapping function which can project from words to their respective concepts we need a data set composed of examples of $< word, concept >$ alignment.

This kind of dataset is *model and vector independent*, because is an alignment between symbols who represent *words* and *concepts* (like $< London, City >$). Since we using DMs we need to retrieve vectors to compose the training tuples. So, starting from previous couples and using the aforementioned vector representation spaces we create the triples $< w_{word}, d_{concept}, h_{concept} >$

Therefore, we build our dataset in the following way:

1. For each $c \in C_T$ retrieve from the KG all the entities ε for which the relation $\varepsilon \in c$ is defined in KG, this step creates the E_c set.
2. For each $\varepsilon_c \in E_c$ find the word $w_{\varepsilon_c} \in W$ which represents ε_c (for instance, the word *'kangaroo'* for the DBpedia entity `dbr:kangaroo`), and create the couple $< w_{\varepsilon_c}, c >$; if the representation of an ε_c cannot be found in W, discharge that ε_c.

3. For each couple $< w_\varepsilon, c >$, retrieve all vectors $v_w \in V_W$, for each v_w create a tuple $< v_w, d_c, h_c >$ with $d_c \in D_{C_T}$ and $h_c \in H_{C_T}$

Once the function Π is learned, we are able to generate new concept representations. Starting from C_I and applying (1) and (2) of the previous procedure we can retrieve words in W which belong to types in C_I.

Π can be used to predict $h_{c_i} \in H_{C_T}$ and $d_{c_i} \in D_{C_T}$ and performing a concept invention. The quality evaluation of this vectors will be discussed in section 6.

4.2 The Source: Distributional Word Space

The V_W is obtained using ELMo [27]. ELMo is a deep learning model which creates a language model during the training on a prediction task. ELMo uses a Bidirectional Language Model (BiLM) to predict a word based on its left context (forward model) or its right context (backward model). Once the BiLM is trained, ELMo's representations are obtained through a fine-tuning on the target task which lets ELMo learn how to combine the representations which come from the forward and backward models to maximize the performances in the fine-tuning task (which can be another prediction task or other NLP tasks).

The *pre-training* of our ELMo model is performed on the *Billion Word Corpus*[1]. The model is then *fine-tuned* obtain a proper representation of DBpedia Abstract Corpus[2] language model, and reflect the distributional word's meaning in this corpus. In V_W each word occurrence representation is based on his sentence context in the *DBpedia Abstract Corpus* and on all contexts of all occurrences of that word in the *Billion Word Corpus.*

ELMo model produces three vectors for each occurrence of each word in the fine-tuning corpus, we use the vectors at *layer 1* because these vectors embody the contextual representation of the word. Due to the ELMo's hyperparameters, the dimension of a contextual word representation is 1024.

4.3 The Target: Distributed Concept Spaces

Distributional Concept Space D_{C_T} is obtained using DBpedia's Types as concepts and is generated with Type2Vec [1], a Word2Vec [23] model applied over a processed corpus: starting from the *DBpedia Abstract Corpus* an *entity linking algorithm* is used to obtain the entities contained in the Corpus, unlinked words are eliminated, subsequently each entity is replaced with its *minimal Type*, which is the deeper Type in the *DBpedia Ontology Tree*[3]. The dimension of the a representation is 100 due to Type2Vec's hyperparameter.
Hierarchical Concept Space H_{C_T} is created with HyperE [29], an algorithm that embeds a graph in a Poincaré Disk Model. The obtained embedding maintains almost exactly the node distances in the graph and has a *low distortion*,

[1] https://opensource.google.com/projects/lm-benchmark
[2] http://downloads.dbpedia.org/2016-10/core-i18n/en/long_abstracts_en.tql.bz2
[3] http://mappings.dbpedia.org/server/ontology/classes/

meaning that the encoded information is only the ones who were in the input graph. H_{C_T} is obtained by applying the HyperE approach to the DBpedia Ontology Tree. The vectors are 2-dimensional and they embody the hierarchical structure of the Ontology tree. Each labelled vector $h_c \in H_C$ (h_{Thing} excluded) has $Norm(h_t) \ll 1$, this is due to the HyperE approach and is a characteristic which has to be treated in the learning phase because these vectors are very closer to the limit of the Poincaré Disk.

4.4 Multi-Task Learning for Concept Invention

We approached the invention problem as a Multi-Task Learning (MTL) problem, this choice is due mainly for two reasons: (i) D_C and H_C express very different semantics. A combined loss which receives the back-propagation from both these spaces can use the information from one space to improve the performance on the other space; (ii) as shown in many cases, in MTL each task works as a regularization function w.r.t. other tasks, this is because learn an intermediate representation which helps to generalize many tasks avoid the overfitting on one of these tasks [28]. Figure 6 shows the architecture of the network. The loss

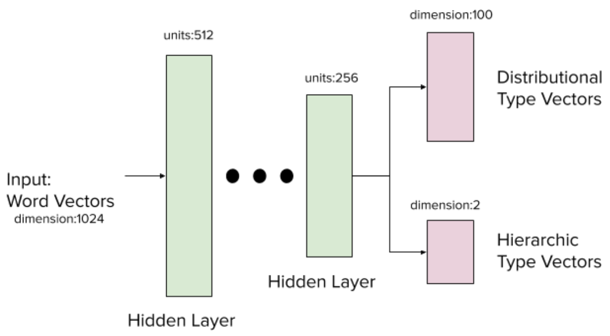

Fig. 6. Architecture of the multi-task feedforward network

function optimized by the MTL network is $L = \lambda L_D + (1 - \lambda)L_H$ which is a convex combination of two loss functions which express the distance in the respective spaces:

- L_D is the mean cosine similarity (Equation 2) between the predictions and the real vectors for each example e in the training set TS. Minimizing this generates vectors which are *distributionally similar* to the real ones because of share a very similar angle w.r.t. the origin.

$$L_D = \sum_{e \in TS} cossim(pred, true)/|TS| \tag{2}$$

– L_H (Equation 3) is the mean hyperbolic distance between the predictions and the real vectors for each example e in the training set TS with a penalization term. By minimize this value, we force the prediction to be closer to the real vectors, the concept of closeness is based on the Poincaré Disk Model topology.

$$\begin{aligned} L_H &= \sum_{e \in TS} d_H(pred, true)/|TS|, \\ d_H(p,t) &= \phi(p) arccosh \left(1 + 2 \frac{\|p-t\|^2}{max(K, 1-\|p\|^2)(1-\|t\|^2)}\right) \\ \phi(p) &= \begin{cases} 1 & \text{if } R < \|v\| < 1 \\ P & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

In Equation (3) K is a penalization parameter forcing the predicted vectors to be in the disk of radius 1, while P is a penalization parameter that amplifies the distance if the predicted vector is inside the radius R disk ($R < 1$). The former penalization is imposed to force the predicted vectors to be placed in the Poincaré disk model domain (inside the unit disk); conversely, the latter penalization forces the vectors to be near from the edge of the disk, which is the region where the predictions are placed in our dataset.

5 Evaluation: Preliminary Results and Discussion

In this Section, we present results of a *Concept Invention* task performed by the MTL network presented in previous sections. We show some example of predictions and provide some numerical results to support our points. The aim of this section is to provide some first evidences on the capabilities of our model to *invent* concepts. Our experiments will be generally based on the following task: generating concept representations of unseen words (words not in the training set) and to check how close they are to the vector of their real type. The intuition is that we hope the predicted vector of the word *beaver* to be close to the vector of the type *Mammal*.

Note that these results are not obtained from unbiased spaces: in fact, as H_{C_T} and D_{C_T} we use the entire set of concepts C, thus we have H_C and D_C. While this might impact our results, we believe that this setting can be used to provide a first feedback over the validity of our assumptions, showing that we can construct representations of unseen concept that are close to the real ones.

For each class $c \in C_I$ we have multiple predicted vectors so h_c and d_c have to be imputed. We think that generating predictions from multiple word vectors V_w of words w which belong to the concept c is a good process: a generalization of concept *'Fish'* has better quality when it is based on different instances of fish. All our data, experimental procedures and complete architectures of the neural networks are available online for replication[4].

[4] https://github.com/NooneBug/Multi_task_concept_invention

DB. Type	Uniq. Words	# vectors
Bird	203	1435
Company	1536	36647
Insect	189	953
Weapon	64	2043
Fic.Character	1127	8026
Sport	198	4459
Planet	64	2140
Colour	47	6938
Plant	901	13465
Arch.Structure	480	5077
SportsTeam	129	828
Mammal	331	9510
Language	385	20566
Award	113	2335
EthnicGroup	556	6411

Table 1. Composition of the training dataset

Word	Ranking in D_C	Ranking in H_C
Algae	Plant	Thing
	Conifer	Species
	Fern	ChemicalSubstance
	Insect	Polyhedron
Beaver	Mammal	Species
	Bird	ChemicalSubstance
	Reptile	Thing
	Fish	Polyhedron
JPEG	Company	Agent
	Bank	Award
	Automobile	Thing
	Software	PublicService

Table 2. Qualitative results on word of unseen concepts

5.1 Qualitative Analysis

To evaluate the quality of a concept invention we selected words which belong to concepts which in DBpedia2016 were *instanceOf Thing*. We choose these words because we want to show that our model can be used to improve the quality of the representation of these entities. We recorded the similarities between the centroid of the predicted vectors and the labelled vectors in the concept spaces. Table 2 shows some examples of predictions.

5.2 Quantitative Analysis

Measures. For evaluation, we use similarity ranking between the predicted vectors and the labelled vectors. For a given predicted vector we compute its neighbourhood and obtain the position of its true type label. Relying on these ranks we compute (and report) the mean rank of the true type label inside the ranking list (*Avg*). We also report the mean position of the true type label in the neighbourhood of a centroid for each semantic class (*Cent*). Both methods are computed over the distributional D and hyperbolic H space.
Settings. We evaluate consider two different settings: (i) *Cent* and *Avg* of predictions about unseen words which belong to concepts used during training and (ii) *Cent* and *Avg* of predictions about unseen words which belong to unseen concepts. The former will evaluate the invention task as standard in machine learning, we test the generalization ability on trained concepts. The latter is performed to explore how similarity between word representations can be used to infer different concepts, thus being a task closer to real concept invention. We consider one baseline model and propose two neural networks.

- Random Forest (**RF**), a random forest regressor model trained to predict the concept vector in the hyperbolic space or in the distributional type space based on the word vector.
- Feed Forward Neural Network with Single Output (**SO**), a neural network which takes in input the word vector and returns a vector in one concept space, performing a regression task.
- Multi-Task Neural Network with Double Output (**DO**), a neural network which takes in input the word vector and returns two vectors: one in the hyperbolic type space and one in the distributional type space. This network uses the loss function $L = \lambda L_D + (1 - \lambda)L_H$ showed above ($\lambda = 0.91$, experimentally computed).

5.3 Results

Table 3 shows that the neural model is able to correctly project a centroid into the distributional concept space and, on average, the projected vectors into the hierarchical concept space have a correct position. The quantitative results show that the distributional projection is probably noisy, show also that the hyperbolic centroid procedure is highly affected by 'wrong' predictions. Results suggest that while the performance of the models is still far from one of a perfect predictor, the approach is promising; moreover the qualitative predictions (Table 2) seem to support this mapping methodology. The neural models seem to be better suited for this task then the baseline. Still DO suffers from some limitations that might be due to the optimization process; nevertheless, its ability to learn from two different spaces and aggregate the information is important for novel tasks related to concept invention; results are heavily influenced by the hyper-parameters of the MTL model (e.g., λ).

Model	Avg D	*Cent* D	Avg H	*Cent* H
RF	108	64	**16**	234
SO	**85**	**4**	17	208
DO	115	5	18	**206**

Table 3. Predictions ranks on unseen words but with known concepts

Model	Avg D	*Cent* D	Avg H	*Cent* H
RF	181	107	**18**	372
SO	**175**	**82**	35	**330**
DO	183	**82**	31	333

Table 4. Prediction ranks with unseen words and concepts

6 Conclusions and Future Work

In this paper we have proposed a first approach to tackle concept invention combining information coming from distributed representations. While our results are preliminary, our experiments show that this approach is promising. We plan to extend this work in several ways: we want to edit our neural network by making it able to predict more accurate representations and we also want to explore the possibility of including different distributed representations in the training process.

References

1. Bianchi, F., Palmonari, M.: Joint learning of entity and type embeddings for analogical reasoning with entities. In: NL4AI@AI*IA. pp. 57–68 (2017)
2. Bianchi, F., Soto, M., Palmonari, M., Cutrona, V.: Type vector representations from text: An empirical analysis. In: DL4KGS@ESWC (2018)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. In: NIPS. pp. 601–608 (2001)
4. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795 (2013)
5. Bouraoui, Z., Schockaert, S.: Automated rule base completion as bayesian concept induction. In: AAAI (2019)
6. Bühmann, L., Lehmann, J., Westphal, P.: Dl-learnera framework for inductive learning on the semantic web. Journal of Web Semantics **39**, 15–24 (2016)
7. Caruana, R.: Multitask learning: A knowledge-based source of inductive bias. In: Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993. pp. 41–48. Morgan Kaufmann (1993)
8. Colton, S., Wiggins, G.A.: Computational creativity: The final frontier? In: ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012. pp. 21–26 (2012)
9. Confalonieri, R., Pease, A., Schorlemmer, M., Besold, T.R., Kutz, O., Maclean, E., Kaliakatsos-Papakostas, M.: Concept invention: Foundations, implementation, social aspects and applications. Springer (2018)
10. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. JASIS **41**(6), 391–407 (1990)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
12. Di Carlo, V., Bianchi, F., Palmonari, M.: Training temporal word embeddings with a compass. In: AAAI (to appear) (2019)
13. Dumais, S.T.: Latent semantic analysis. Annual review of information science and technology **38**(1), 188–230 (2004)
14. Eppe, M., Maclean, E., Confalonieri, R., Kutz, O., Schorlemmer, M., Plaza, E., Kühnberger, K.: A computational framework for conceptual blending. Artif. Intell. **256**, 105–129 (2018)
15. Fauconnier, G., Turner, M.: Conceptual integration networks. Cognitive Science **22**(2), 133–187 (1998)
16. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. In: Linear Algebra, pp. 134–151. Springer (1971)
17. Gromov, M.: Metric Structures for Riemannian and Non-Riemannian Spaces, vol. 152, pp. –586 (12 2006)
18. Harris, Z.S.: Distributional structure. Word **10**(2-3), 146–162 (1954)
19. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th conference on Computational linguistics-Volume 2. pp. 539–545. Association for Computational Linguistics (1992)
20. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguñá, M.: Hyperbolic geometry of complex networks. Phys. Rev. E **82** (2010)

21. Le, M., Roller, S., Papaxanthos, L., Kiela, D., Nickel, M.: Inferring concept hierarchies from text corpora via hyperbolic embeddings. arXiv preprint arXiv:1902.00913 (2019)
22. Lenci, A.: Distributional semantics in linguistic and cognitive research. Italian journal of linguistics **20**(1), 1–31 (2008)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. pp. 3111–3119 (2013)
24. Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: Stranse: a novel embedding model of entities and relationships in knowledge bases. In: NAACL. pp. 460–466. The Association for Computational Linguistics (2016)
25. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. In: NIPS. pp. 6341–6350 (2017)
26. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. pp. 1532–1543. ACL (2014)
27. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: NAACL. pp. 2227–2237. ACL (2018)
28. Ruder, S.: An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098 (2017)
29. Sala, F., Sa, C.D., Gu, A., Ré, C.: Representation tradeoffs for hyperbolic embeddings. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 4457–4466 (2018)
30. Šourek, G., Manandhar, S., Železnỳ, F., Schockaert, S., Kuželka, O.: Learning predictive categories using lifted relational neural networks. In: International Conference on Inductive Logic Programming. pp. 108–119. Springer (2016)
31. Völker, J., Niepert, M.: Statistical schema induction. In: Extended Semantic Web Conference. pp. 124–138. Springer (2011)
32. Wang, C., He, X., Zhou, A.: A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In: EMNLP. pp. 1190–1203. Association for Computational Linguistics (2017)
33. Wiggins, G.A.: Searching for computational creativity. New Generation Computing **24**(3), 209–222 (2006)
34. Xiao, P., Toivonen, H., Gross, O., Cardoso, A., Correia, J., Machado, P., Martins, P., Oliveira, H.G., Sharma, R., Pinto, A.M., et al.: Conceptual representations for computational concept creation. ACM Computing Surveys (CSUR) **52**(1), 9 (2019)