Technical Report

UCAM-CL-TR-967
ISSN 1476-2986

Number 967

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Gaussian Pixie Autoencoder: Introducing Functional Distributional Semantics to continuous latent spaces

Primož Fabiani

January 2022

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

© 2022 Primož Fabiani

This technical report is based on a dissertation submitted June 2021 by the author for the degree of Master of Philosophy (Advanced Computer Science) to the University of Cambridge, Hughes Hall.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

Functional Distributional Semantics (FDS) is a recent lexical semantics framework that represents word meaning as a function from the latent space of entities to a probability for each word. This thesis examines previous FDS models, highlighting the advantages and drawbacks. A new Gaussian Pixie Autoencoder model is proposed to introduce FDS to continuous latent modelling. The proposed model improves on the predecessors in terms of simplicity and efficiency setting a new baseline for continuous FDS models. The thesis shows dropout is necessary for context learning with this model type. Evaluated on contextual similarity the proposed model outperforms the discrete autoencoder and BERT baseline on one task with satisfactory performance on the other.

Acknowledgements

I would like to thank my mentor Dr. Guy Edward Toh Emerson for help, support and plenty of good advice in preparing this thesis. I would also like to thank my friends and family for the help in making this happen.

Contents

1 Introduction **9**
- 1.1 Outline . . . 9
  - 1.1.1 A note on notation . . . 9

2 Background **10**
- 2.1 Goals of lexical semantics . . . 10
- 2.2 Evaluating the performance of lexical models . . . 10
- 2.3 Challenges of lexical semantics . . . 10
- 2.4 Distributional semantics and vector space models . . . 11

3 Functional Distributional Semantics **13**
- 3.1 Model structure and the real world . . . 14
- 3.2 Separating features of referents from the word meaning . . . 15
- 3.3 Pixies and the semantic space . . . 15
- 3.4 Semantic functions and lexical meaning . . . 16
- 3.5 Probabilistic graphical model formulation . . . 16
- 3.6 Logical inference with the probabilistic graphical model . . . 17
- 3.7 Amortised variational inference and the Pixie Autoencoder . . . 19

4 Moving Functional Distributional Semantics to a continuous latent space **24**
- 4.1 Overview . . . 24
- 4.2 Continuous Pixie Autoencoder with a Gaussian latent space . . . 25
- 4.3 World model . . . 25
  - 4.3.1 Parameterising the precision matrix . . . 26
  - 4.3.2 Parameterizing the mean vectors in the world model . . . 28
- 4.4 Inference network . . . 29
- 4.5 Lexical model . . . 30
  - 4.5.1 Deeper semantic functions . . . 31
- 4.6 Computing the loss function . . . 32
- 4.7 Calculating the KL divergence between distributions . . . 32
- 4.8 Computing the Reconstruction loss . . . 35
- 4.9 Final loss function formulation . . . 36

5 Implementation **37**
- 5.1 Overview and architecture . . . 37
- 5.2 Merging the objective functions . . . 37
- 5.3 Sampling . . . 38
- 5.4 Dropout . . . 38
- 5.5 Regularization . . . 39
- 5.6 GPU support . . . 39

6 Evaluation **40**
6.1 Model training 40
6.2 GS2011 41
6.2.1 Results and comparison 42
6.3 RELPRON 42
6.3.1 Results and comparison 43
6.4 Discussion 43

7 Summary and Conclusions **44**

A Extended results **45**

List of Figures

3.1 Probabilistic graphical model with a pixie-graph world model (top) connected to the lexical model (middle) for all predicates in vocabulary V. Predicate (bottom) chosen based on truth values magnitude. Adapted from: Emerson (2020) 17
3.2 The visualization of the adapted Cardinality RBM in the world model with fully connected bidirectional weights between pixies with a semantic relation. 18
3.3 The pixie encoder network architecture convolving the observed predicates in the DMRS graph and outputting a probability distribution over features for each pixie inferred from the entire situation. Adapted from: Emerson (2020) 21

4.1 An example precision matrix on the right and its probabilistic graphical model equivalent representation. Zero values in the matrix imply conditional independence. 26

List of Tables

6.1 Hyper-parameters used for evaluated training runs 41
6.2 Comparison of Spearman's rank correlation performance on GS2011 dataset (Previous work adapted from (Emerson, 2020)) 42
6.3 Comparison of mean average precision (MAP) performance on RELPRON data (Previous work adapted from (Emerson, 2020)) 43

A.1 Performance of individual runs on the RELPRON evaluation task 45
A.2 Performance of individual runs on the GS2011 evaluation task 45

Chapter 1

Introduction

1.1 Outline

The task of representing word meaning both in its static dictionary form as well as in context is a fundamental problem of lexical semantics and natural language processing (NLP) in general. Furthermore, high-quality lexical representations that capture human intuitions on word interrelation are crucial to countless downstream applications and both NLP and linguistics research. Functional Distributional Semantics (FDS) introduced by Emerson and Copestake (2016) and Emerson (2018) is a novel probabilistic lexical semantics framework that separately models latent representations of entities being described and words as functional mappings from those latent entity representations to word predicates. This thesis aims to expand the work done on Functional Distributional Semantics by introducing a new continuous latent representation machine learning model for inferring and representing word meaning. This thesis first focuses on explaining the previous computational models based on Functional Distributional Semantics and exploring their strengths and weaknesses. Next, a new continuous model is proposed to simplify and improve the previous formulations. The new model utilizes a Gaussian latent space with a separate world model used to simultaneously learn the prior distribution over the latent space and train an encoder network to perform amortised variational inference of contextual latent entity representations. In addition, a lexical model is jointly trained to perform the functional mapping from latent representations to the probability of word membership for all vocabulary words. Finally, the model is evaluated to demonstrate the effectiveness of the proposed method so that the model can serve as a starting point for further integration of FDS with modern downstream applications.

1.1.1 A note on notation

Due to sometimes dealing with higher dimensional tensors and the general desire to keep notation coherent, this thesis adopts the notation conventions used in Emerson (2018). This includes index notation for tensors which means all tensors are written with subscripts that range over dimensions of the vector space. For example, v_i could represent a vector and M_{ij} a matrix. Thus, the order of the tensor always matches the number of subscripts. The Einstein summation convention is also adopted so that indexes repeated in an expression are summed over. For example $v_i u_i$ represents a vector dot product while $M_{ij} L_{jk}$ represents a matrix multiplication. Indices required to distinguish variables from each other which do not correspond to dimensions are going to be shown by superscripts.

Chapter 2

Background

To motivate the work done in this thesis and show the advantages of introducing Functional Distributional Semantics to continuous latent space, this chapter presents an overview of the goals of lexical semantics and briefly explain key approaches, their benefits and ways FDS addresses their issues.

2.1 Goals of lexical semantics

Computational lexical semantics aspires to infer and represent word meaning. This encompasses the task of representing both the standing meaning of words as given by dictionary definitions as well as the contextual meaning, a word might express in a specific sentence or text. Many different approaches have emerged to tackle the issue such as dictionary and thesaurus methods, but the focus has largely shifted to distributional methods. The foundation of distributional semantics is the distributional hypothesis formulated by Harris (1954), which postulates that words that occur in the same contexts tend to have similar meanings. This observation is especially appealing since it presents the opportunity to use readily available bodies of text to extract the context in which words appear that can then be used with machine learning methods. This opened up the field of **word representation learning** which is the core task presented in this thesis and has, for a long time, been one of the core NLP problems.

2.2 Evaluating the performance of lexical models

The evaluation of the performance of word representation learning computational models is difficult as, for the most part, models train on an unsupervised objective such as the reconstruction of contexts in which the word appears while being evaluated in agreement with human annotators and benchmark sets on the quality of the learned lexical representations. This often takes the form of lexical similarity as measured by the distance between learned representations or word sense disambiguation. The idea of word meaning can be formulated in many ways, each with its own set of assumptions and expectations.

2.3 Challenges of lexical semantics

The first such issue is the underlying **vagueness** of word meaning. A content word usually refers to either an entity, event or concept and those have a lot of overlap and ambiguity.

For example, two words might be synonyms in which case the set of things they refer to should strongly overlap or in the case of hyponyms and hypernyms stand in a subset relation. Due to this, a well-motivated approach should allow for the representation of word meaning to express that relation in some way, which is not always easy to balance with other practical considerations when choosing the representation.

Another important challenge is **homonymy** which is the condition of a single word having several unrelated senses that just happen to coincide lexically. A classic example is the word *school* which refers both to an educational institution and a group of fish where the senses are semantically disjoint.

Finally, there is polysemy which is the case where a word has several related but distinct meanings. Consider for example the use of the word newspaper in the following sentences.

- "She put the newspaper on the table."
- "The local newspaper was sued for libel."

The word newspaper in the first sentence refers to a physical object while in the second refers to an institution. The senses are clearly semantically connected while still being distinct in how a person imagines them and their properties.

2.4 Distributional semantics and vector space models

The most popular approach to creating lexical representations from distributional data is using vector space models. The key idea is representing the word meaning by a vector of features. A key vector space approach in early word meaning representation by Lund and Burgess (1996) was for features to be based on context word counts simply describing the ratio of words likely to appear around the target word. The crucial variable in this approach is how the neighbourhood or context is defined based on direct word distance or other syntactic properties. A major drawback of such frequency-based vector space approaches is the unavoidable high dimensionality of the representations bound to vocabulary size. The vector space approach has further been very successfully expanded to **vector embedding models** which are now a standard in NLP research. The embedding approach utilizes machine learning techniques such as the word2vec model by Mikolov et al. (2013a) that introduced the Skip-gram and CBOW architectures and GloVe models by Pennington et al. (2014), to learn lower-dimensional real-valued vector word representation from simpler distributional information such as count-based vector representations which can be obtained directly from text corpora. Word embeddings are very effective in storing valuable semantic information, both for direct analysis on tasks such as word similarity and analogy (Mikolov et al., 2013b), as well as downstream tasks of all kinds including machine translation (González-Rubio et al., 2013), sentence classification (Kim, 2014), question answering (Bordes et al., 2014) and numerous others. Despite all the undeniable successes of the approach, there are issues with how word meaning is represented in this stream of approaches. The main issue, as pointed out by Camacho-Collados and Pilehvar (2018) in their survey of vector space models, is the **meaning conflation deficiency** - the problem of constraining all word meanings to a single word representation and thus creating the inability to discriminate among them. This seriously limits the granularity of meaning present in most text by, for example, reducing the ambiguous meaning of the word *nail* as both a body part and a fastener to a single representation. This implicit limitation of word representations to being monosemous thus forces downstream applications to multiplex the meaning back to a multi-modal distribution over possible

meanings, which likely increases the needed downstream model complexity unnecessarily, if fine-grained understanding of word meaning in different contexts is required.

To address the challenge posed by the meaning conflation deficiency, the **word sense representation** approaches were introduced. This approach suggests that the word meaning representations should be constructed in a context-dependent way so that every use of a word could be assigned a different representation. This approach was deeply explored by Erk (2010) who suggested that word meaning would better be modelled as a graded degree of sense membership which would allow for multiple senses to apply to a single occurrence. To accomplish this, a large number of unsupervised and knowledge-based approaches were developed. These mainly focused on using external word sense inventories such as WordNet (Miller, 1995; Miller et al., 2008). They also often focused on first disambiguating of a corpus into senses before applying word sense learning methods on the data as done by Iacobacci et al. (2015) and Mancini et al. (2017). For a more thorough overview of these methods see (Camacho-Collados and Pilehvar, 2018; Navigli and Martelli, 2019).

The vector embedding approach to word meaning representation often means that the sense embeddings have to be produced by two-stage methods, such as first performing word sense disambiguation and then the word meaning representation learning or applying the learning first and then correcting via clustering, which can make determining which representation correlates to an unseen use in a downstream approach challenging.

Another major development in the sphere comes from **contextualized embeddings** produced by language models such as BERT by Devlin et al. (2019) and ElMo by Peters et al. (2018). The contextualized embedding methods take an unsupervised approach to learning sense embeddings and avoid the finite sense directory problem by inferring a custom word sense representation for each word used in context and does not restrict itself to a finite set of word sense representations. The contextualized embedding approach has had considerable success in recent years and has become very popular in the field. It has also been shown by Chang and Chen (2019) that contextualized embeddings can be mapped back to word sense collections, demonstrating the information needed for word sense disambiguation is present in the embeddings. The approach is, however, still based on a vector semantic space where each word use gets a custom embedding that represents both the word's meaning and the properties of what it is referring to. The Functional Distributional Semantics approach developed by Emerson and Copestake (2016) and Emerson (2018) offers an alternative formulation of word meaning that does not make this assumption.

Chapter 3

Functional Distributional Semantics

Functional Distributional Semantics is a computational semantics framework introduced by Emerson and Copestake (2016) and Emerson (2018). It introduces a way to merge the distributional semantics' bottom-up ability to learn the semantic structure of the language directly from distributional information present in the text with the model-theoretic semantics' top-down rigour in defining the notion of truth and compatibility with first-order logic and λ calculus. This chapter presents the Functional Distributional Semantics Framework and gives linguistic motivations for many of its properties. Next, the computational models based on FDS are presented. The goal is to show both the key advances introduced by the previous models as well as explain their drawbacks and motivate the model proposed by this thesis.

Model theory postulates that linguistic expressions acquire meaning by being interpreted in the model structure. A model is structured as a set of **individuals** also called **entities**, such as objects and people. The meanings of content words describing such entities are called **predicates**, mapping entities to truth values. The truth value thus describes whether the entity belongs to the set defined by the predicate. This means the predicate can also be viewed the other way around as a set of entities or, in the case of n-place predicates, n-tuples of entities belonging to a set of the predicate called the **extension**. In more practical terms, this means that a model is formed of a set of variables representing entities that predicates can take as arguments in determining their truth values and thus membership. This is enough to form logical propositions composed of predicates and entities. The framework also incorporated the Neo-Davidsonian idea that action and event predicates themselves should be assigned entities in the model structure since they themselves have properties. A very convincing example is given by Davidson (1966) where the sentence *"John did it slowly, deliberately, in the bathroom, at midnight."* refers to John buttering a piece of toast. Here the 'it' in the sentence refers to the act of 'buttering' which has properties describing how, where, etc. that belong to the action and not just the actor, and thus require predicates to define its membership. The framework incorporates this by assigning an entity variable to events and actions that can then be given verbal predicates to define their corresponding predicate. Furthermore, the participants of the event then have to be connected to the event via other predicates no longer provided by the event itself. This provides the opportunity for generalization where the model utilizes the idea of semantic roles as predicates. Semantic roles are predicates denoted here by ARG1 and ARG2 for a two participant event example that link the participant entity variables x and y to the event variable z. For example, "John butters toast" can be decomposed as *John(x), butters(z), toast(y), ARG1(z,x),*

ARG2(z,y). Such use has the advantage of semantic roles (ARG1, ARG2) only defining the relation between entities with no reference to the predicates describing their content *(toast(), butters())* and can thus be reused for all examples with the same semantic structure of semantic relations. This approach also benefits from the generalizability to other parts of speech as adjectives, adverbs and propositions can similarly be assimilated into the model structure via the introduction of new entity variables and semantic roles. In order to computationally represent this structure the DMRS graph structure presented by Copestake (2009); Copestake et al. (2016) is used throughout this thesis.

3.1 Model structure and the real world

With the linguistic background of the framework defined it is prudent to ask how the representation formed by the model structure relates to and represents the world. The first important thing presented in the Functional Distributional Semantics framework is the idea that a specific model does not aim to represent the entire world any more than any specific image invoked in the mind does. The model, constructed from a statement, only pertains to the elements of the underlying situation being described in the text. Thus, the model describes a situation that can be viewed as a possible state of a part of the world. This allows for models to be constructed based on situations ranging from the very simple, such as the statement "John butters his toast", to arbitrarily large and detailed descriptions without changing the rules by which they are constructed. It also allows for hypothetical and imaginary situations to be modelled, since the framework represents the relations and structure of the entities rather than the factual content of the situation. To judge the plausibility of the described situation, two additional levels are necessary.

We want the model to learn something about the **plausible relations entities have** with each other, based on what model structures are likely, given examples of human-created text. This is largely what semantic models have done historically, as described in the section on vector models. This is achieved within the Functional Distributional Semantics framework via the world model sub-component that learns to represent the probability of different entities forming different situations. The second aspect of determining the truth of propositions formulated as a model is connecting the word meaning to the real world it aims to describe. This process is called **grounding** and forms a fundamental challenge to NLP. This was pointed out in the linguistic context by (Harnad, 1990) who showed that if all word meanings are constructed only from information obtained from the text, then all definitions are circular and form a closed ungrounded system. This problem goes back to the very foundations of analytic philosophy and was of issue to (Frege, 1948) and the problems of connecting logical statements to the real world. The problem is often illustrated by the famous Chinese room experiment by (Searle, 1984) where a semantic model can learn a detailed model of a language without gaining any real understanding of its meaning in relation to the real world. The FDS framework does not attempt to achieve grounding in this sense but nonetheless offers a compelling formulation that allows for the word meaning and world knowledge to remain separate but linked and thus provides a bridge between formal logic and world knowledge, leaving the problem of true grounding for future work.

3.2 Separating features of referents from the word meaning

Functional Distributional Semantics takes the approach of model theory in distinguishing **concepts** representing word meaning from **referents** which are the entities in the concept's extension. Thus a situation as described above consists of referents in some relation that can be described by concepts. This leads to the intuition that word meaning as concepts should describe the relation between referents and their belonging to concepts. To achieve this purely with vector-based embeddings would generally require one of two different approaches. Either the concept embeddings representing general word meaning would have to be embedded in the same space as entities (individuals) which would mean additional distance-based or clustering methods are needed to bind entities to corresponding concepts. This approach assumes the properties represented in features of entities and those in the representation of concepts are of the same type which makes modelling multi-modal distributions of meaning such as in the case of polysemy difficult. The alternative approach is to have concepts and entities embedded in different vector spaces but this likewise introduces overhead, since an external mapping between the two spaces has to be learned.

The separation of word meaning from entity properties also opens the question of *context dependence*. When considering the properties of referents represented by entities, the meaning changes quite drastically depending on the context in which they appear. A good example given by Emerson (2018) is the case of a *small elephant* where the referent is not a small animal whereas a *large ant* is. This shows the need for representations of the referents to be context-dependent. The usefulness of this approach is also evident in the success of context-dependent embedding models described above. This, however, leaves the issue of words and corresponding concepts having an existence independent of any particular use which presents a need for a consistent and generalizable way of representing them. The Functional Distributional Semantics framework makes this distinction explicit by adopting the idea of **occasion meaning** to represent the former and **standing meaning** to describe the latter, following Recanati (2012).

3.3 Pixies and the semantic space

Given the challenges outlined above, the FDS framework presented by Emerson (2018), Emerson and Copestake (2016) proposes to model word meaning in two parts. Entities representing features of the referents are placed in a discrete semantic space where each point in the space represents a set of features that could belong to an individual. Depending on the dimensionality of the space, the feature representations can be more or less fine-grained and thus each point might represent more than one actual referent. Since the space in the original formulation is discrete, each such point of the space was called a *pixie*, following the intuition of being a pixel of the discrete space. Pixies, as members of the semantic space, thus model referents as entities by representing their features as opposed to representing generalized word meaning.

However, not all combinations of features are created equal as not everything that could exist does. The framework thus imagines the semantic space to have a natural probability distribution over likelihoods of entities (individuals, events ...) represented by pixies actually existing. This leaves us with a probabilistic semantic space that allows for probability information about the world to be learned.

$$\text{picture} \xleftarrow{\text{ARG1}} \text{tell} \xrightarrow{\text{ARG2}} \text{story}$$

3.4 Semantic functions and lexical meaning

The generalized word meaning envisioned under model theory as a predicate must then define a set of those entities (as represented by pixies in the semantic space) that belongs to the set of its extensions. However, since there is significant ambiguity in which extensions belong to each set and overlap in predicate boundaries, FDS follows Erk (2010) in modelling belonging to a predicate in a graded way. Emerson (2018) explores a range of options but settles on a probabilistic model representing the degree of belonging to a predicate as a probability. Thus, the standing meaning of a word is a **semantic function** that maps pixie elements of the semantic space to the probability of predicate membership. Each semantic function can thus be viewed in two ways: a binary classifier and a conditional probability of word predicate being true conditioned on the pixie entity representation. The semantic function can then be represented as:

$$t(x) = \mathbb{P}(T = \top \mid X = x) \tag{3.1}$$

This formulation is very elegant since it allows for probabilistic composition of knowledge of the a priori probability of the pixie existence with the conditional probability of the pixie belonging to a word predicate. The combined model of semantic functions for all predicates is called the **lexical model**. This formulation of word standing meaning as functions also introduces a natural way to express the mapping in terms of regions of the semantic space where a particular semantic function has a high probability.

3.5 Probabilistic graphical model formulation

We can now focus on the probabilistic graphical model. The first thing to discuss is how context-dependence can be modelled in the framework. The probabilistic graphical model forms a graph where each node represents a pixie, here imagined as a random variable, while the edges indicate probabilistic dependence between pixies. The edges between pixies are undirected as they represent two-way dependence. The graph of pixie nodes and their edges composes the world model. The goal of the world model is to learn the relations between pixies by learning which correlations of features form probable situations based on examples observed in a body of text parsed into semantic DMRS graphs. The edges and dependence relations are undirected since both random variables in a semantic relation codetermine each others' meaning. The second part of the model is the lexical model comprised of the set of semantic functions for all vocabulary words that represent the unidirectional conditional probability of each predicate given a node (pixie) in the world model.

Since each semantic function only shares a directed dependency with one of the pixies, the probability of each predicate is independent of other pixies in the latent situation represented by the graph. Conversely, the values of nodes in the latent situation are codependent on each other but not on the truth values produced by the semantic functions. This is motivated by the observation that the existence of objects and events with specific features is independent of how you describe them linguistically, yet depend on the context of surrounding entities.

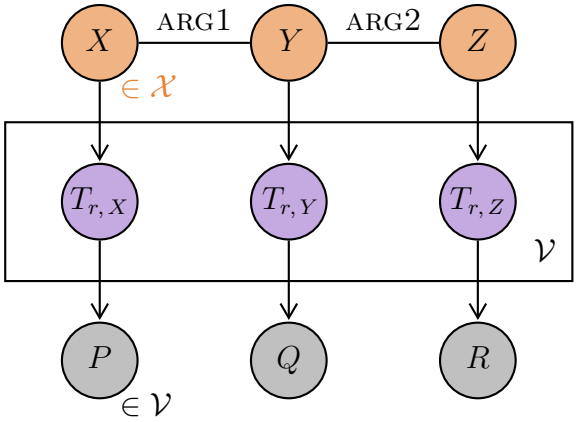

Figure 3.1: Probabilistic graphical model with a pixie-graph world model (top) connected to the lexical model (middle) for all predicates in vocabulary V. Predicate (bottom) chosen based on truth values magnitude. Adapted from: Emerson (2020)

3.6 Logical inference with the probabilistic graphical model

The original probabilistic graphical model formulation by Emerson (2018) would have to infer contextual pixie graphs for each latent situation described by the DMRS graph and then use the inferred pixies to evaluate the predicate probabilities given the semantic functions in the lexical model.

This approach to constructing a probabilistic graphical model architecture dynamic, based on the semantic relations present in the utterance describing the latent situation, allows for the framework to accommodate arbitrary semantic parse graph shapes within a framework such as DMRS. This allows for a great deal of flexibility in what the models following the framework can implement.

The world model portion of the probabilistic graphical model represents a probability distribution over pixie values. To do this, it must define a parametrization of such a distribution. Since the model mixes directed and undirected edges in the model, no existing models were found in the literature that can neatly represent the distribution. Further, since a discrete space of Boolean vectors was chosen to represent the pixie values and the features of pixies were assumed to be sparse, the chosen approach was based on using a **Cardinality Restricted Boltzmann machine** presented by Swersky et al. (2012). Unlike with the standard restricted Boltzmann machine, this formulation inverts the structure so that the weighted connections are not between visible and hidden units. Instead, a fully connected layer is placed between all pairs of pixies connected by a semantic role. The Cardinality RBM idea also introduces an additional constraint not present in the standard version that restricts the total number of active units at any time.

The RBM world model is, however, an energy-based model and does not parametrize a probability distribution in a way that can be analytically normalized. Instead, given a random variable describing the latent situation S the probability is proportional to the negative energy of the model. To get a proper probability distribution, a normalization constant Z also had to be introduced, since without it the energy is subject to infinite growth as the probabilities do not sum up to 1. The energy of the model depends on the product of all pixie value pairs connected by a link with their corresponding weights and a bias term for each pixie feature. Those properties can be expressed together to get the

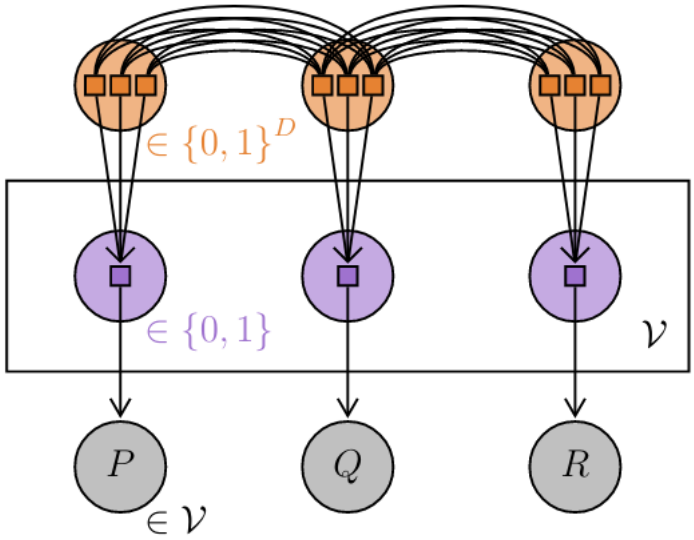

Figure 3.2: The visualization of the adapted Cardinality RBM in the world model with fully connected bidirectional weights between pixies with a semantic relation.

formulation of the world model probability distribution.

$$-E(s) = \sum_{x \to y \text{ in } s} w_{ij}^{(l)} x_i y_j - \sum_{x \text{ in } s} b_i x_i \tag{3.2}$$

$$Z = \sum_{s'} \exp\left(-E\left(s'\right)\right) \tag{3.3}$$

$$\mathbb{P}(S = s) = \frac{1}{Z} \exp(-E(s)) \tag{3.4}$$

In this formulation, each semantic role is assigned a weight matrix W of non-negative real values. The first term of (3.4) sums over the effects of all semantic relations in the graph, while the second term gives the likelihood of features being active irrespective of the graph topology.

The lexical model is tasked with the probabilistic inference of the degree of belonging between contextually inferred pixies, representing the entity and the word predicate being applicable. The FDS approaches this problem by imagining the word meaning to be a function, called a **semantic function**, that maps from the semantic space of pixies to the probability of belonging to a given predicate. Each semantic function is represented as a single-layer feed-forward neural network. The network takes a vector of parameters v that represents the strength of association with each dimension of the semantic space. The function outputs a value between 0 and 1 representing the probability of belonging. The probabilities outputted by the semantic function are not global probabilities across all possible predicates but rather represent the degree of the model's certainty that the predicate applies. To get the probability relative to all other predicates the value has to get normalized over all predicates in the vocabulary via Z. This is formulated by Emerson (2018) as shown in the following equations (3.5), (3.6),(3.7) where F score represents the degree of association and t again represents the semantic function for predicate r. In the

original formulation, the semantic function also included a bias term a governing the a priori probability of the predicate, which is dropped in subsequent models.

$$-F(x,r) = v_i^{(r)} x_i - a^{(r)} \tag{3.5}$$

$$t^{(r)}(x) = \frac{1}{1+\exp(F(x,r))} \tag{3.6}$$

$$\mathbb{P}\left(T^{(r,X)} = \top \mid X = x\right) = t^{(r)}(x) \tag{3.7}$$

To avoid the intractable normalization over all predicates to get the random variable Z for computing of relative probability of predicates, the model introduces a **mean field approximation** approach to compute the expected value for a semantic function output that avoids having Z being a random variable by summing over the frequency of predicates $f^{(r)}$. This slightly improves things but still requires predicates of all vocabulary items to be considered.

$$\mathbb{P}\left(R^{(X)} = r \mid X = x\right) = \frac{1}{Z(x)} f^{(r)} t^{(r)}(x) \tag{3.8}$$

$$Z(x) = \sum_{r'} f^{(r')} t^{(r')}(x) \tag{3.9}$$

For the probabilistic graphical model to perform inference of pixie values and learn the parameters of semantic functions, a learning schema had to be introduced. This was first achieved through the use of **Markov Chain Monte Carlo** methods, however, these require a significant amount of sampling to converge so **variational inference** methods were introduced to speed up the process. The key idea of variational inference is to alleviate the difficulty of sampling from a hard distribution P by introducing a simpler distribution Q then optimizing its parameters to match the original distribution closely before sampling from the simpler distribution. The match between distributions is measured using their Kullback-Leibler (KL) divergence from the simpler one Q to the original P since the relation is not symmetric. The model performed reasonably well when evaluated but also introduced a few challenges. The most important issue is the need to perform Bayesian inference over the latent space which is intractable to calculate exactly and requires quite a lot of approximation for the model to be feasibly performant. Furthermore, even the approximations require each graph to be inferred separately which can be very computationally expensive.

3.7 Amortised variational inference and the Pixie Autoencoder

To address the issues of the original model and improve performance Emerson (2020) developed the **Pixie Autoencoder** model that performs amortised variational inference on the DMRS graphs. The idea of amortised variational inference is to approximate the variational inference that would otherwise have to be done independently for each graph by an autoencoder neural network that learns a set of parameters to infer the latent pixie graph from a DMRS graph input. Thus the approach reduces the computation needed. To be able to work on graphs as input, the autoencoder performs graph convolutions to

learn information about the entire situation and condition all entities (pixies) present on each other. At the same time, the model should ideally still produce a latent distribution that coheres to the probabilistic graphical model.

The Pixie Autoencoder model consists of three sub-models. First, there is the world model component that represents the probabilities of different latent situations as presented in the original model. The second is the lexical model containing the semantic functions that describe the conditional probability of word predicates given a pixie latent entity representation. The final and new component is a pixie inference encoder network that learns to infer the latent pixie graph distribution given an input DMRS graph.

The world model is implemented as described above, following the Cardinality RBM architecture. The probability of an inferred latent situation is then proportional to the negative energy of the world model which in this formulation omits biases as can be seen in (3.10), (3.11).

$$\mathbb{P}(s) \propto \exp(-E(s)) \tag{3.10}$$

$$\mathbb{P}(s) \propto \exp\left(\sum_{x \to y \text{ in } s} w_{ij}^{(l)} x_i y_j\right) \tag{3.11}$$

The lexical model and semantic functions also undergo a few modifications, such as no longer containing a bias term, but generally follow the same equations.

$$t^{(r)}(x) = \sigma\left(v_i^{(r)} x_i\right) \tag{3.12}$$

Following the idea of *variational inference*, the model aims to introduce a simpler distribution Q to sample from that is tethered to the more difficult one P. The difficult distribution, in this case, is the distribution over latent situations represented by Cardinality RBM linked pixie graphs. In order to be able to use a simpler distribution Q we have to learn its parameters $\mathbf{q}$ such that it closely approximates P. The closeness of the distributions is again only relevant in one direction since we want the distribution Q to be fully aligned with the more complex P while not necessarily matching all the complexity of P. This can again be expressed as the KL divergence between the distributions. Normalizing the distribution over the latent situations is, however, still intractable and requires summing over all of them. The full probabilistic graphical model can finally be viewed to optimize the log probability of the observed graph g under distribution P as expressed by (3.13).

$$\begin{aligned} \frac{\partial}{\partial \theta} \log \mathbb{P}(g) = \left(\mathbb{E}_{s|g} - \mathbb{E}_s\right) \left[\frac{\partial}{\partial \theta}(-E(s))\right] \\ + \mathbb{E}_{s|g}\left[\frac{\partial}{\partial \theta} \log \mathbb{P}(g \mid s)\right] \end{aligned} \tag{3.13}$$

While the first graphical model combining the world and lexical models already did this kind of variational approximation the Pixie Autoencoder model goes a step further and introduces **amortised variational inference** to not only approximate parameters of Q

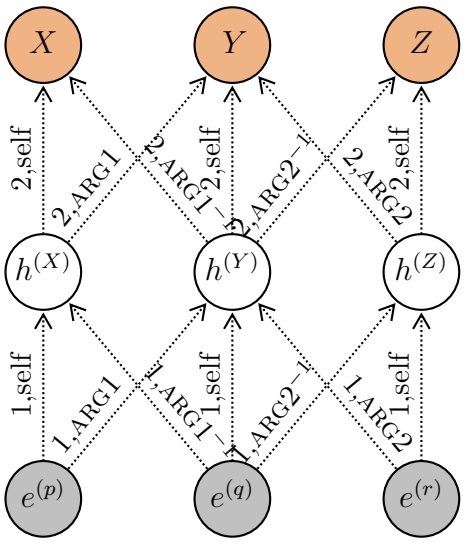

Figure 3.3: The pixie encoder network architecture convolving the observed predicates in the DMRS graph and outputting a probability distribution over features for each pixie inferred from the entire situation. Adapted from: Emerson (2020)

for each input graph but introduce an additional inference network with its own parameters ϕ that learns to infer the parameters q of distribution Q given an input DMRS graph. Since the pixie inference network is learning its parameters across a range of input graphs, this is another layer of approximation and thus the results might not be optimal. The introduction of this approach, however, offers a large improvement in efficiency, since the inference network once trained allows new examples of input graphs to be inferred in a single forward pass of the network as opposed to being individually optimized.

The amortised variational inference network presented by Emerson (2020) is structured as a variational autoencoder that takes in DMRS graphs and outputs a probability distribution over features of each pixie. Since pixies are here represented by binary vectors, the output is formed as a probability of activation for each binary feature in the pixie. Since the encoder works on DMRS graphs, the encoder network needs a way to propagate the information along the graph structure. This is important since the autoencoder aims to create contextualized representations of pixies conditioned on the entire situation. For example, the pixie representing the features of the entity inferred from the word *cut* should be different if inferred from the sentence *The gardener cut the grass*, where it would be similar to one inferred from *mow* as opposed to one inferred from the sentence *the child cut the cake* where it may be more adjacent to the meaning of *slice*. To achieve this, the encoder network uses a graph convolutional approach inspired by Kearnes et al. (2016) and Gilmer et al. (2017). As shown in figure (3.3), the model generally has three layers since the graphs trained on are mostly triplets. The network defines a weight matrix w for each semantic relation represented as an edge in the graph for each direction since there is no reason that semantic relations are symmetric in meaning as, for example, the relation verb $\rightarrow$ subject and subject $\rightarrow$ object. A set of weights is also introduced for self-mapping from layer to layer, transmitting the information about the input predicate in constructing the pixie representation. The values of the hidden unit h in the network can therefore be calculated as follows for layer k and node X.

$$
\begin{aligned}
h_i^{(k,X)} = f(&w_{ij}^{(k,\text{ self})} h_j^{(k-1,X)} \\
&+ \sum_{Y \leftarrow X} w_{ij}^{(k,l)} h_j^{(k-1,Y)} \\
&+ \sum_{Y \rightarrow X} w_{ij}^{(k,l^{-1})} h_j^{(k-1,Y)})
\end{aligned}
\tag{3.14}
$$

The Pixie Autoencoder architecture is similar to the architecture of standard variational autoencoders presented by Kingma and Welling (2014) and Doersch (2021), the objective function of the autoencoder is comprised of two parts the - **reconstruction error** which describes how closely the autoencoder replicates input values on the output and the **divergence error** usually measured by the KL divergence between the distributions that describes to what extent the learned latent distribution is coherent with the target latent distribution.

The divergence error is based on the divergence of the inferred latent to the target latent which is usually a distribution with known properties in traditional variational autoencoders. The Pixie Autoencoder attempts to approximate the latent space of the world model in the probabilistic graphical model. To achieve this, the encoder network is tethered to the probabilistic graphical model by measuring the KL divergence between the latent spaces and optimizing for minimal divergence. Since the desired latent probability distribution over situations is not only hard to sample from but also unknown, the probabilistic graphical model and the pixie inference encoder networks learn their parameters together while both are forced to minimize the KL divergence from the distribution Q given by the latent of the encoder and P given by the probabilistic graphical model. This gives the best of both worlds as the distribution Q is easy to sample while being locally consistent with P. When all pixies in the latent pixie graph are inferred, the lexical model can be applied to produce a conditional probability for all predicates in the vocabulary by applying semantic functions. The total model is thus a variational encoder that aims to match the distribution of the world model and the lexical model as a decoder with all components training simultaneously.

The reconstruction error is the probability of the input predicate at the output for each pixie. Put abstractly the reconstruction error can be incorporated into the KL divergence if the lexical model is assumed to be a part of the probabilistic graphical model and not a separate component. This formulation is used by Emerson (2020) that defines the total loss of the Pixie Autoencoder model as seen in (3.15) which describes the KL divergence between the distributions.

$$
D(\mathbb{Q} \| \mathbb{P}) = -\mathbb{E}_{\mathbb{Q}(s)} \left[\log \left(\frac{\mathbb{P}(s \mid g)}{\mathbb{Q}(s)} \right) \right] \tag{3.15}
$$

This can further be expanded into a three-part formulation more clearly outlining the components as shown in (3.16).

$$\begin{aligned}\frac{\partial}{\partial\phi}D(\mathbb{Q}\|\mathbb{P}) = &-\frac{\partial}{\partial\phi}\mathbb{E}_{\mathbb{Q}(s)}[\log\mathbb{P}(s)]\\ &-\frac{\partial}{\partial\phi}\mathbb{E}_{\mathbb{Q}(s)}[\log\mathbb{P}(g\mid s)]\\ &-\frac{\partial}{\partial\phi}H(\mathbb{Q})\end{aligned} \tag{3.16}$$

The first term is the negative energy of the inferred pixies and can be calculated directly from them. The third term represents the entropy H which can be computed easily. Together the first and third term correspond to the divergence error component of the ELBO. The second component represents the log probability of generating the correct predicate and corresponds to the reconstruction error in the ELBO. Computing the exact probability of the true predicate being chosen requires summing over the conditional probabilities of all the predicates. The cost of doing this for the whole vocabulary can, however, be alleviated by using a subset of negative samples as an approximation. The second issue is the incorporation of variance into the probability of the predicate. Emerson (2020) resolves the issue by using an approximation based on the work of Murphy (2012). The approximation assumes a Gaussian input for the expected value while deriving the variance $Var[x]$ directly from the pixie feature probability vector q as $q(q-1)$.

$$\mathbb{E}[\sigma(x)] \approx \sigma\left(\frac{\mathbb{E}[x]}{\sqrt{1+\frac{\pi}{8}\operatorname{Var}[x]}}\right) \tag{3.17}$$

The application of amortised variational inference speeds up the learning time and makes the inference of new example graphs, once the training is completed, exceptionally efficient compared to the previous approach. The Pixie Autoencoder approach matched or even surpassed the performance of some of the most successful contemporary models when examined on the tasks of contextualized lexical similarity and outperformed the previous results of directly applying the probabilistic graphical model.

The framework, however, also offered many avenues for further development and experimentation. One of the major drawbacks of the model was the unwieldy nature of the Restricted Boltzmann Machine based world model. The RBM by itself does not form a valid probability distribution out of the box and has to get normalized. Since the equation (3.4) only guarantees the probability is proportional to the model negative energy the calculation of the exact probability requires summing over all possible latent situations i.e. the entire latent space which is completely intractable even for a discrete model. Thus normalizing the RBM prior probability in the discrete Pixie Autoencoder model is intractable and requires approximation. The method proposed in Emerson (2018) and Emerson (2020) for performing the approximations relies on belief propagation methods (Yedidia et al. (2003)) that increase computational complexity and introduce issues with GPU support. The move to a continuous latent space model proposed in this is thus motivated by the ability to choose a probability distribution with known integral over the latent space so that the normalization of probabilities can be done much more efficiently. In addition, most modern lexical semantics models as presented in the background use continuous space models for good reasons and introducing a model that introduces Functional Distributional Semantics to the space of continuous models offers a range of new possibilities for integration with other models and might thus be of great value for further work in the field.

Chapter 4

Moving Functional Distributional Semantics to a continuous latent space

4.1 Overview

Given the promise of the Functional Distributional Semantics and the Pixie Autoencoder architecture, the inherent drawbacks of the RBM-based world model might be avoided by structuring the latent space according to a different probability distribution. This could mean transitioning to a different binary probability distribution or attempting to test whether the methodology can be translated to a continuous space formulation that avoids the pitfalls of the old model. Most modern lexical semantics models work with continuous latent spaces since that allows for degrees of intensity in each feature to be represented within a dimension and can reduce the requirements on the dimensionality of the latent space. This gives a good indication that implementing a continuous latent space model based on the FDS might be a very interesting next step to attempt. Therefore the decision was made to focus on finding a continuous probability distribution that can be used instead of the Cardinality RBM.

The desired properties of the new distribution are based on two factors. The distribution should have general desirable properties such as being simple to work with which here means having well-known properties. Furthermore, the distribution should be easy to normalize to avoid previous issues. The second relevant set of desired properties relate to the structure of the model. Since the probabilistic graphical model defines semantic relations as probabilistic links between latent variables, the shape of the graph is not known in advance, which has to be accommodated by the distribution. The bidirectional nature of the semantic relation probabilistic dependencies also means that conditional probabilities must be defined in a bidirectional way.

The multivariate Gaussian is one of the most widely used probability distributions in machine learning and is very well understood. It allows for arbitrary dimensionality and can be resized with relative ease. It also has a historical affinity with variational autoencoders where a zero mean variant is regularly used to represent the latent space. The multivariate Gaussian has also been experimented with before as a replacement of vector embeddings in favour of directly representing word meaning by it as proposed by

Vilnis and McCallum (2015). In this case, the approach was, however, quite different since the goal was to directly represent word meaning by a Gaussian distribution and then perform similarity checks by observing the relations between the distributions of different words. In contrast, the goal of Functional Distributional Semantics is to use the distribution to represent the shared latent space of all entities in the latent situation and their interaction while the lexical model defines word meaning and similarity. Despite the major differences in approach, this work offered insights that were relevant in constructing the new model. Due to these properties and the advantages they offer, the project adapted the multivariate Gaussian for the world model latent space.

4.2 Continuous Pixie Autoencoder with a Gaussian latent space

To adapt the model to continuous space, all components require some modification. First of all, the word *pixie* was originally introduced to indicate the pixel-like structure of the latent representation in the discrete space which does not make much sense in a continuous Gaussian formulation. The term is nonetheless still used throughout this thesis to refer to the latent entity representation. The next few sections outline the theoretical formulation of the proposed model component by component and explain the advantages and drawbacks of the approach.

4.3 World model

The world model aims to represent the a priori probability of different configurations of pixies defining the latent situation. Learning the world model probability distribution thus amounts to learning the likelihood of different situations with no direct reference to a particular observation. This was previously accomplished by the RBM energy-based model that would learn the strength of association of binary features between connected pixies. To examine how this can be replaced by a Multivariate normal distribution, it is useful to first examine its properties.

The general form of the multivariate Gaussian (multivariate normal) $\mathcal{N}$ for a p dimensional space is as follows. The random variable x is here a p dimensional real vector.

$$\mathcal{N}(x_i \mid \mu_i, \Sigma_{ij}) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x_i - \mu_i)\Sigma_{ij}^{-1}(x_j - \mu_j)\right] \tag{4.1}$$

Here, μ is a p dimensional mean vector for the distribution and Σ is a $p \times p$ symmetric positive definite (pd) matrix. By definition, the expectation of X is given by the mean vector, $E[X] = \mu$ and Σ represents the covariance matrix $Cov[X] = \Sigma$ (Murphy, 2007). The multivariate Gaussian distribution also has the desirable property of remaining Gaussian under both scaling $wy \sim \mathcal{N}(w\mu, w^2\sigma^2)$ and addition $\sum_{i=1}^{n} y_i \sim \mathcal{N}(\sum_{i=1}^{n} \mu_i, \sum_{i=1}^{n} \sigma_i^2)$.

World model structure can be imagined as a graph of pixies connected by semantic relations indicating conditional dependence. Since the distribution works over the entire situation, all pixies present in the graph have to be joined into a single representation. Furthermore, since the distribution is Gaussian, there exist only two sets of parameters to describe it, namely the mean vector μ and the covariance matrix Σ. The mean vector of each pixie can only represent a single expected value for each feature and thus cannot

represent any interactions. Therefore, the mean vectors of all pixies in the graph can simply be stacked to produce the combined mean vector of the situation. The mean vector of the situation for pixies a,b and c is thus just their concatenation.

$$\mu_S = \mu^a \oplus \mu^b \oplus \mu^c \tag{4.2}$$

Looking at it in more detail, we can see every feature represented by a dimension in each pixie might or might not have a conditional dependence on any other feature of any pixie, including other features of its own pixie. This would result in a fully connected graph of conditional dependence. This can be represented by a covariance matrix where each side is of the dimensionality of the situation mean vector thus the sum of the lengths of all present pixies. Creating such a fully connected graph would, however, ignore the information provided by the semantic relations in the DMRS graphs and be both wasteful and unmotivated. The features within a pixie should not be constrained to being conditionally dependent since we want them to represent as much information as possible and not be correlated to each other.

Similarly, we desire for features to not depend on features of pixies they do not share a semantic relation with. Setting all such parameters in the precision matrix to 0 reduces the complexity of the model and utilizes semantic relations to guide the model in learning the important information. Therefore, the desired structure only allows conditional dependence between features of pixies that share a semantic relation.

4.3.1 Parameterising the precision matrix

The covariance matrix is, however, not just a collection of values that can be used as weights arbitrarily and has a specific meaning and properties. Each value in the covariance matrix represents the absolute correlation of the corresponding feature pair. This is a bit of a problem since a coefficient of 0 in the covariance matrix indicates the features are marginally independent without observing other variables, which is a very strong claim to make. What we really want to represent is the conditional dependence of the features, given all other variables. Conveniently, this can be represented by the **precision matrix** Q which is nothing more than the inverse of the covariance matrix. The precision matrix has the nice property of having the coefficient of 0 for all pairs of features that are conditionally independent of each other, given other variables. The precision matrix also has a convenient graphical interpretation of representing the strength of dependence as the weight of the edge between variables where the coefficient of 0 can be interpreted as no edge being present between the features in the probabilistic graphical model.

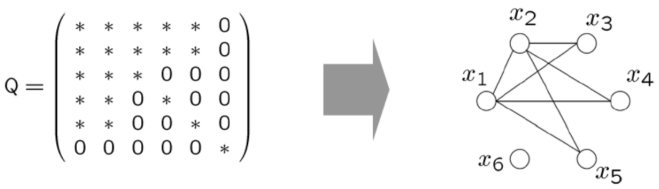

Figure 4.1: An example precision matrix on the right and its probabilistic graphical model equivalent representation. Zero values in the matrix imply conditional independence.

The multivariate normal distribution can now be rewritten using the precision matrix. Assuming mean μ and $Q = \Sigma^{-1}$ being the precision matrix the probability density function can be expended as shown in the general form in eq. (4.3):

$$\mathcal{N}(x_i \mid \mu_i, Q_{ij}) \stackrel{\text{def}}{=} \frac{1}{(2\pi)^{-p/2}|Q|^{1/2}} \exp\left[-\frac{1}{2}(x_i - \mu_i)Q_{ij}(x_j - \mu_j)\right] \tag{4.3}$$

In this case, we can conceptualize the world model as an $N \times D$ matrix of real-valued pixies where each of N columns is a pixie of D dimensions. Much like in the original model, there is a fully connected layer of weights between each pair of pixies that share a dependency in the DMRS graph. In order to parameterize this, we can use the precision matrix of size $p \times p = (N \times D) \times (N \times D)$. This can be imagined as one large precision matrix split into meaningful blocks. Since this would be a huge matrix for even very small graphs we do not actually need to represent or evaluate all the parameters for a few reasons:

1. **Unit diagonal** is assumed. The correlation of each variable to itself is defined to always be 1 and need not be computed. There was serious consideration of having diagonal elements also be learned but was abandoned for two reasons. First, the magnitude of all values in the multivariate Gaussian is only relevant relative to other values. Thus, choosing the variance of all features to be 1 is a reasonable assumption and does not hinder performance. Second, the covariance and consequently precision matrix must be positive semi-definite and having a static diagonal helps with stability. For this reason, the variance values on the diagonal are not used as learning parameters and need not be stored.
2. **Mutual internal pixie feature independence**. All precision matrix coefficients relating features within a single pixie are 0 as there are no connections between values in the same pixie. This means the blocks of coefficients around the diagonal are all set to 0 and also do not have to be stored as parameters.
3. **Zero coefficients for unrelated pixies** The precision matrix assigns no conditional dependence and thus connections between pixies that are not adjacent in the DMRS graph. This means all coefficients of such connections can be a priori set to 0 and left out of the model implementation. Thus, only blocks where a semantic relation is present get parameterised.

This leaves us with a model with the exact same amount of connections and weights as the original model that actually need to be computed.

$$\begin{array}{l} \\ \\ Pixie\ a \\ \\ Pixie\ b \\ \\ Pixie\ c \end{array}
\begin{array}{c}
\overbrace{}^{Pixie\ a} \quad \overbrace{}^{Pixie\ b} \quad \overbrace{}^{Pixie\ c} \\
\begin{bmatrix}
\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} & W_{ab} & \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \\
W_{ab}^T = W_{ba} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} & W_{bc} \\
\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} & W_{bc}^T = W_{cb} & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}
\end{bmatrix} \\
WORD\ A \xleftarrow{ARG\ 1} WORD\ B \xrightarrow{ARG\ 2} WORD\ C
\end{array} \tag{4.4}$$

Given this formulation, the precision matrix is parametrised as a symmetric block matrix formed of parameter blocks, empty blocks, and identity blocks. The identity blocks cover the diagonal of Q since all values in pixies are taken to be fully conditionally related to themselves and conditionally independent from each other, given other connected pixies. The empty blocks are zero matrices that represent the forced conditional independence between pixies with no semantic relation in the graph. A parameter block is thus defined for each semantic dependency type and inserted into the appropriate precision matrix slot that connects the corresponding pixies in the graph. Thus, the precision matrix ends up representing the exact link structure we expect from a probabilistic graphical model with an RBM but within the multivariate Gaussian distribution.

4.3.2 Parameterizing the mean vectors in the world model

The mean vectors of pixies in the world model represent a prior on the position of a pixie in the latent space. The simplest thing to do would be to set them all at 0 mean. This is, however, very restrictive and implements the assumption that all latent entities and events represented by pixies are clustered in a single spot in the latent space which gives little flexibility in where the model can learn to place them. Thus, a more interesting parametrization had to be devised. The key idea here is that the mean vector parameters are derived from the graph structure with no knowledge of the exact predicate. Thus the question is what DMRS properties of nodes are sensible to use. The obvious choice is to use the entity event distinction that roughly corresponds to the verb versus noun distinction in predicates. Thus, two basis mean vectors are introduced to the world model namely the **entity mean** μ_e that is present for nouns and **event mean** μ_x that is present for verbs. Both are stored in the world model and get trained as active parameters. When the full situation mean vector is needed, they get composed based on node types as described above. This allows for the world model to use structural information about the number and type of entities present in the situation and their semantic relations without getting any information about which predicates were used. This makes it able to learn the prior probability over the situation structure more precisely than would otherwise be possible.

$$\mu^{type} = \begin{cases} \mu^{entity} & : \ type = entity \\ \mu^{event} & : \ type = event \end{cases} \tag{4.5}$$

But we can go further by modifying the prior over the mean of each pixie in the graph with other known properties found in the semantic graph structure. Those properties are introduced by adding a **property vector** for each property to either the entity-mean or the event-mean as appropriate. There are 11 overall properties split between those applicable to nouns and those applicable to verbs with each having 2 or more possible values. Most of them are, however, not directly applicable either because they convey information about the predicate word that is not relevant to the referent or are, as in the case of *person* where all predicates in the training data are set to the third person, incompatible with the current experimental setup. Thus, in practice, only two additional property vector types were added. For entities, the **number** property was introduced so that the world model would be able to modify the mean by a different vector for singular or plural forms. For events, the **tense** property was introduced so that the model can encode an event's mean differently for past, present, and future.

$$\mu^{number} = \begin{cases} \mu^{singular} & : number = singular \\ \mu^{plural} & : number = plural \end{cases} \tag{4.6}$$

$$\mu^{tense} = \begin{cases} \mu^{past} & : tense = past \\ \mu^{present} & : tense = present \\ \mu^{future} & : tense = future \end{cases} \tag{4.7}$$

Thus, the prior mean vector for each pixie node in the world model can be seen simply as a sum of all applicable property mean vectors where nothing is added if the property is not applicable including the main entity/event distinction. This model can easily be extended in the future to arbitrary properties provided the dataset makes such an addition prudent.

$$\mu^{a} = \mu^{type} + \mu^{tense} + \mu^{number} \tag{4.8}$$

4.4 Inference network

The inference network is the component that infers the parametrised probability distribution over the pixie graph, given a DMRS graph of word predicates and their relations.

Much like in the previous version (Emerson, 2020), the inference network follows the variational autoencoder structure and utilizes graph convolutions to propagate information across the whole graph. This ensures all the probability distribution parameters represented as pixies are inferred from the whole context of the sentence. Since the goal of the inference network is to infer the parameters of a simple probability distribution Q that approximates the more complex probabilistic graphical model distribution P, there is no need to explicitly represent the interactions of the pixies since the values are inferred jointly and thus already conditioned on each other. The discrete version of the model used a single vector of binary probabilities to represent the probability distribution over all features in each pixie. In the new version, the distribution produced by the inference network must approximate a multivariate normal of the probabilistic graphical model. To this end, each pixie distribution is given as the **mean and variance vector** pair. The co-variances can be ignored since the mean values are already conditioned on each other.

$$\mathrm{L}^{a} = (\mu_i^a, \mathrm{Var}_i^a) \tag{4.9}$$

This parametrization is widely used with variational autoencoders and gives a clear and interpretable notion of a region of the semantic space with features matching the model's image of the referent. The variance gives further depth by showing the model's degree of uncertainly about the features of the referent and allows for hypernym and hyponym words describing the referent to be mapped in clear relation to each other with similar means but varying variances.

The encoder architecture to compute the pixie (mean, variance) tuples is for the most part identical to the one used in the discrete version and has already been discussed in chapter 3. The main difference is that the final layer is split in two where one side computes the mean and the other the variance. This means having joint layers in the encoder up to the last layer where the data is passed through 2 separate dense layers one for mean and one for variance. Since the variance is always positive $\mathrm{Var} = \sigma^2$, the variance is not

computed directly here but rather the inference network outputs the logarithm of the variance. This avoids the need to explicitly use a computation that returns a positive value here in favour of a simple dense layer. More importantly, this simplifies a later computation of the loss function. This, however, makes no conceptual difference to the interpretation of the model so can, for the purpose of understanding, mostly be ignored as the logarithm can be reversed on demand.

$$\begin{aligned} \mu^a &= W_{ij}^{\mu}\ x_j + b_i^{\mu} \\ \ln \mathrm{Var}^a &= W_{ij}^{\mathrm{Var}}\ x_j + b_i^{\mathrm{Var}} \end{aligned} \tag{4.10}$$

4.5 Lexical model

The task of the lexical model is to learn the parameters of conditional probabilities of all word predicates given any pixie of the semantic space. The key idea of Functional Distributional Semantics is to instantiate that probability as a **semantic function** of learned parameters. This means the lexical model has to learn the semantic function parameters for all vocabulary word predicates. The probability of a predicate r given a pixie x is thus proportional to the semantic function value as seen in eq: (4.11). Normalising requires summing over the vocabulary.

$$\mathbb{P}(r \mid x) \propto t^{(r)}(x) \tag{4.11}$$

The semantic function model has a vector of parameters v_i for each predicate. The semantic function can thus be seen as a feed-forward neural network. Under the original discrete model, the semantic function **truth value** $t^{(r)}$ for a predicate r is calculated by multiplying the pixie (specifying the probability of each dimension) by the weights v_i^r. To get a value between 0 and 1 representing the degree of certainty, the results are passed through a sigmoid activation function as shown in eq: (4.12).

$$t^{(r)}(x) = \sigma\left(v_i^{(r)} x_i\right) \tag{4.12}$$

Since the semantic function takes an exact pixie as a point in the semantic space, but the encoder network gives us the parameters of a probability distribution over such a pixie, there are two possible approaches to take. The standard approach with variational autoencoders is to sample a value from that distribution and use that for decoding purposes. The sampling approach is resistant to systematic bias but introduces significant variance in outputs which can be a problem for datasets where most values are going to get sampled only a few times. As most vocabulary items occur very rarely, this is relevant for this application. The second approach adopted by Emerson (2020) in the original version of the model is directly approximating the value based on the variance. The approximation used (4.13), adopted from Murphy (2012), assumes roughly Gaussian input and gives the corrected expected value based on the variance.

$$\mathbb{E}[\sigma(x)] \approx \sigma\left(\frac{\mathbb{E}[x]}{\sqrt{1 + \frac{\pi}{8}\,\mathrm{Var}[x]}}\right) \tag{4.13}$$

The approximation risks introducing slight systematic bias but provides better stability for training sparse predicates and is faster to compute. Since the approximation is well-

suited for Gaussian input, the proposed model also follows this formulation where pixie mean (expected value) and variance are directly provided by the inference network.

4.5.1 Deeper semantic functions

The presented lexical model is fully adapted to a continuous space model, but there are additional improvements that might be of value. Since the semantic functions presented above are simple single-layer feed-forward networks, an obvious suggestion would be to introduce additional hidden layers and thus make the network deeper and more powerful. This might be especially important when considering the case of polysemy and homonymy. In both cases, a word might have a multimodal probability distribution over the latent space where for example the word *bank* might have a high probability for regions in the semantic space corresponding to features of financial institutions as well as topological features around rivers. In this case, a deeper semantic function might be able to represent a more complex probability distribution that can capture such phenomena.

The first idea here was to simply add a **matrix of weights for all predicates** W^r that can be used as a hidden layer. This approach, however, quickly proved problematic. The size of vocabulary can be adjusted based on the needs but would, for a serious language model, be expected to be quite large. With the vocabulary size ranging from tens of thousands to millions of words and each of which is assigned a matrix of similar dimensionality as the latent space, the size of the model parameters grows too fast for available memory. Furthermore, due to the long tail of word use frequency, most words would have to train a deeper network with only a fraction of the dataset which can be difficult. This approach was implemented but, due to memory constraints, could not be tested at all.

To correct for the intractable memory requirements, a **shared hidden layer** approach could be used. Under this formulation, all predicates share a single matrix of weights for the hidden layer while maintaining separate final layer parameters v^r as before. This approach has the benefit of training the hidden layer at every single application of any semantic function thus removing the issue of sparsity. This, however, makes the hidden layer mapping exactly the same for all predicates which simply corresponds to a global transformation of the latent space which might still be useful for separability but is not really introducing deeper semantic functions.

To reap the benefits of both low memory requirements and unique mapping for all predicates a **merged approach** was proposed. Under this scheme, a parameter vector a^r_k of length K would be stored for all predicates in the vocabulary. In addition, a rank three tensor B_{ijk} with dimensions (K, D, D) where D is the pixie dimensionality would be kept. Then, for any predicate r, a hidden layer parameter matrix could be computed by contracting the tensor B with the vector a^r corresponding to a predicate. This would create a matrix M^r_{ij} combination of the K different D x D matrices specific to the predicate.

$$M^r_{ij} = B_{ijk}\ a^r_k \tag{4.14}$$

This approach, though not totally independent, allows for each predicate to construct its own hidden layer mapping from a shared set of weights. It also has the benefit that the hidden layer parameters get trained on every use and no longer suffer from the sparsity problem or at least suffer no more than the single-layer approach. Finally, the memory requirements are dramatically reduced, since a vector is kept for each predicate instead

of a matrix. The created matrix M_{ij}^r can then be used as before to compute the output of a hidden layer with activation function f as shown in (4.15)

$$h_i^r = f(M_{ij}^r x_j) \tag{4.15}$$

4.6 Computing the loss function

The last piece of the puzzle is how the three models can be trained together to learn how to represent word meaning. Since the architecture follows the structure of the variational autoencoder, it might be sensible to start the discussion of the loss function there. In the classic formulation of variational autoencoders given by Kingma and Welling (2014), the goal is to maximize the probability of the data given the model. This can be expressed as the combination of the KL divergence of the inferred latent distribution Q and the data distribution P, and a term called **evidence lower bound** ELBO, also called variational lower bound. The parameters of the inferred distribution Q are represented by ϕ, the parameters of the target distribution are given by θ, with x stands for input and z for latent representation.

$$\log p_\theta\left(x^{(i)}\right) = D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_{\boldsymbol{\theta}}\left(z \mid x^{(i)}\right)\right) + \mathcal{L}\left(\theta, \phi; x^{(i)}\right) \tag{4.16}$$

Since KL divergence is non-negative, the ELBO term represents the lower bound on the probability of the data.

$$\log p_\theta\left(x^{(i)}\right) \geq \mathcal{L}\left(\theta, \phi; x^{(i)}\right) = \mathbb{E}_{q_\phi(z|x)}\left[-\log q_\phi(z \mid x) + \log p_\theta(x, z)\right] \tag{4.17}$$

The term can be rewritten further to show it is composed of the KL divergence of the latent distribution to the prior and the reconstruction error.

$$\mathcal{L}\left(\theta, \phi; x^{(i)}\right) = -D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta(z)\right) + \mathbb{E}_{q_\phi\left(z|x^{(i)}\right)}\left[\log p_\theta\left(x^{(i)} \mid z\right)\right] \tag{4.18}$$

4.7 Calculating the KL divergence between distributions

Unlike most variational autoencoders, the latent distribution prior is not static under this model but rather is defined by the world model network. This means that the KL divergence has to be computed between two changing distributions. The way the three models were parametrised, however, gives us some serious advantages. First, since both the inference network latent pixie distribution and the world model distribution can be directly interpreted as multivariate Gaussians, the KL divergence can be calculated directly from the distribution parameters. This avoids a lot of the issues with the energy-based RBM world model since the probabilities can be normalized directly. The KL divergence between multivariate Gaussians can be given in general form using only the distribution parameters:

$$D_{\mathrm{KL}}[p^1 \mid\mid p^2] = \frac{1}{2}\left[\log\frac{|\Sigma^2|}{|\Sigma^1|} - n + \Sigma_{ij}^{(2)-1}\Sigma^(1)_{ji} + (\mu_i^2 - \mu_i^1)\Sigma_{ij}^{(2)-1}(\mu_j^2 - \mu_j^1)\right] \tag{4.19}$$

As shown in eq: 4.20, the KL divergence between 2 multivariate Gaussians can be calculated directly from the multivariate normal parameters μ and Σ. Here the equation is segmented into three parts to better show how the computation is done.

Part 1 is the log of the ratio of determinants between the covariance matrices of both distributions $\log \frac{|\Sigma^{(2)}|}{|\Sigma^{(1)}|}$. Here, we can simplify a few things. First, since the covariance matrix $\Sigma^{(1)}$ of the distribution Q parametrised by the encoder is a diagonal matrix composed of the variance vectors of all pixies, the determinant can be trivially calculated as the product of the diagonal elements. This, however, leads to a problem. The product of a few hundred values, such as the diagonal in question, tends to either shrink to 0 or grow to values out of range. Both scenarios lead to computational issues even if they are formally correct. This was resolved by following the example of Kingma and Welling (2014) which keeps the determinants in a logarithmic form throughout the calculation. Thus, the encoder outputs the ln σ^2 instead of the direct variance. This is very helpful since the goal is to compute the logarithm of the determinant which can be moved to the other side so that the log variance vector produced by the inference network can simply be summed to get the logarithm of the determinant.

$$\begin{aligned} ln\, det(\Sigma_1) &= ln \prod \text{diag}(\sigma^{2(1)} .. \sigma^{2(n)}) \\ &= \sum (\text{ln diag}(\sigma^{2(1)} .. \sigma^{2(n)})) \\ &= \sum (\ln(\sigma_i^2)) \end{aligned} \tag{4.20}$$

The determinant $\det(\Sigma^{(2)})$ of world model distribution P can not be computed directly by constructing the full precision matrix P due to limitations of the Dynet automatic differentiation methods. This is a purely practical issue but had to be addressed in the overall design. Thus, a new approach was devised. The full precision matrix is a square matrix composed of (N x N) square matrices of dimensions (D x D) each, for N being the number of pixies in the graph and D being the pixie dimensionality. This allows for block matrix rules to be used to perform the computation in parts and avoid the overhead of constructing the full matrix.

The goal is then to compute the determinant of a large precision matrix by working directly on blocks. This can be done by brute force but that can be quite computationally expensive. Luckily, the Leibniz formula for determinants (4.21) also applies for block matrix calculations in the 2x2 and 3x3 cases if comutativity of blocks in all operations can be assured. This is always the case under the proposed approach since the blocks of the leading diagonal are always identity matrices and in the 3x3 case the semantic graph is always a tree which guarantees an additional set of zero matrices will be present. Thus the commutativity constraint is always guaranteed to be satisfied.

$$\det(A) = \sum_{\tau \in S_n} \text{sgn}(\tau) \prod_{i=1}^{n} a_{i,\tau(i)} = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} a_{\sigma(i),i} \tag{4.21}$$

The usual downside to the formula is the need to compute all permutations of the elements and is thus asymptotically proportional to $n!$ in performance. This is, however, not an issue here since the graph is only ever going to contain a handful of block matrices and thus finding the permutations is computationally cheap. Some additional effort is needed in computing the sign of permutation for all permutations. This can be done with a

few different procedures but given the small graph sizes, might be most elegantly and efficiently done with the quadratic time algorithm counting the disordered pairs (4.22).

$$\text{sgn}(\sigma) = (-1)^{\sum_{0 \leq i < j < n} (\sigma_i > \sigma_j)} \tag{4.22}$$

Finally, instead of getting the product of values, we must here get the product of block sub-matrices. This computation is also allowed since we can compute the product of matrices A and B composed of block matrices by components as shown in (4.23) (4.24).

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1s} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{q1} & \mathbf{A}_{q2} & \cdots & \mathbf{A}_{qs} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \cdots & \mathbf{B}_{1r} \\ \mathbf{B}_{21} & \mathbf{B}_{22} & \cdots & \mathbf{B}_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{s1} & \mathbf{B}_{s2} & \cdots & \mathbf{B}_{sr} \end{bmatrix} \tag{4.23}$$

$$\mathbf{C} = \mathbf{AB} \qquad \mathbf{C}_{qr} = \mathbf{A}_{qi}\mathbf{B}_{ir} \tag{4.24}$$

Thus, by using the Leibniz formula for determinants, the block sub-matrices can first be combined without change of dimensionality with the determinant of the resulting matrix being equivalent to the total determinant of the precision matrix.

Part 2 given by the trace $\text{tr}\{\Sigma^{-1(2)}\Sigma^{(1)}\}$ does not require the matrix multiplication to be fully computed, since the covariance matrix $\Sigma^{(1)}$ is in our case given by the variance vectors produced by the encoder. This means all the off-diagonal values are 0. Conveniently, the world model precision matrix $\Sigma^{-1(2)}$ is constrained to have a constant value of 1 on the diagonal. This is helpful since the multiplication of a matrix by a diagonal matrix results in successive columns simply being multiplied by the corresponding diagonal value eq: 4.27.

$$\begin{bmatrix} a_{11} & a_{12} & .. & a_{1n} \\ a_{21} & a_{22} & .. & a_{2n} \\ .. & .. & .. & .. \\ a_{m1} & a_{m2} & .. & a_{mn} \end{bmatrix} \begin{bmatrix} k_1 & 0 & .. & 0 \\ 0 & k_2 & .. & 0 \\ .. & .. & .. & .. \\ 0 & 0 & .. & k_n \end{bmatrix} = \begin{bmatrix} k_1 a_{11} & k_2 a_{12} & .. & k_n a_{1n} \\ k_1 a_{21} & k_2 a_{22} & .. & k_n a_{2n} \\ .. & .. & .. & .. \\ k_1 a_{m1} & k_2 a_{m2} & .. & k_n a_{mn} \end{bmatrix} \tag{4.25}$$

$$i = j \rightarrow a_{ij} = 1 \tag{4.26}$$

Since we are computing a trace that only cares about the resulting diagonal, we only need to compute the item-wise multiplication of the diagonals. But as one of the diagonals is a unit vector we can omit even this and simply take the sum of the concatenation of variance vectors produced by the encoder.

$$\text{tr}\{\Sigma^{-1(2)}\Sigma^{(1)}\} = \sum(\text{tr}(\Sigma^{(1)})) \tag{4.27}$$

Part 3 given by $(\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1)$ requires the calculation to be done in block matrix form again, since the whole world model precision matrix has to be used in the computation. The implementation of this can be a bit tricky but the operation is conceptually quite simple. All operations happen on the level of pixies, where pixie means and corresponding parameter matrices get multiplied together following the rules outlined in (4.23). Multiplications including zero matrices can be ignored and diagonal unit block matrices can be inserted where needed without storing them permanently.

$$
Q = \begin{array}{c} \\ Pixie\ a\ \{ \\ Pixie\ b\ \{ \\ Pixie\ c\ \{ \end{array}
\begin{array}{c}
\overbrace{Pixie\ a} \quad \overbrace{Pixie\ b} \quad \overbrace{Pixie\ c} \\
\begin{bmatrix}
I & W_{ab} & 0 \\
W_{ab}^T = W_{ba} & I & W_{bc} \\
0 & W_{bc}^T = W_{cb} & I
\end{bmatrix}
\end{array}
$$

$$
word\ A \xleftarrow{ARG\ 1} word\ B \xrightarrow{ARG\ 2} wordC \tag{4.28}
$$

4.8 Computing the Reconstruction loss

Compared to computing the KL divergence, the reconstruction loss is quite simple. The reconstruction loss describes the difference between the input and the output of the autoencoder. The role of the decoder, in this case, is performed by the lexical model. The lexical model already contains the function for calculating the conditional probability of a predicate directly from each pixie distribution (4.13), the outputs of semantic functions for input predicates could simply be added together. This, however, ignores the condition that we want each semantic function to behave like a binary classifier and assign low scores to all the incorrect predicates. In the simple case, this can be accomplished simply by evaluating every pixie in the graph against all semantic functions in the vocabulary to obtain a vector of all truth-value scores and then calculating the probability of the input predicate by normalizing against all others.

Thus obtained probability can, however, be deceiving since we want the model to learn to place similar word predicates together and one pixie might validly correspond to many predicates. This is perhaps again best seen in the case of hypernymy where we want the hyponym and the hypernym to both give a high probability for the features of the pixie they describe. This cannot be directly addressed without introducing word similarity information into the training process and effectively cheating. However, Emerson (2020) includes both the probability and the truth value into the reconstruction loss which can be seen as an indirect way of addressing the issue. It also helps with keeping the truth value of the correct predicate high, irrespective of the ratios of probabilities. Thus, this approach was also adopted here. The total formulation can be seen in (4.29) for each pixie x with input predicate R in graph G and for the vocabulary of semantic functions V.

$$
\begin{aligned}
\mathcal{L}^{\text{reconstruction}} &= \sum_{(x,R)\in G} \frac{t^R(x)}{\sum_{r\in V} t^r(x)} + t^R(x) \\
&= \sum_{(x,R)\in G} P(R|x) + t^R(x)
\end{aligned} \tag{4.29}
$$

4.9 Final loss function formulation

Finally, we can express the total loss function of the new model in one equation (4.30). This formulation is, however, somewhat theoretical with a lot of practical machine learning concerns to discuss when it comes to implementation.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}^{\text{deviation}} + \mathcal{L}^{\text{reconstruction}} \\ \mathcal{L} &= D_{\text{KL}}[Q \mid\mid P] + \sum_{(x,R)\in G} P(R|x) + t^R(x) \end{aligned} \tag{4.30}$$

Chapter 5

Implementation

5.1 Overview and architecture

With the theoretical background and the proposed model for continuous Functional Distributional Semantics, the practical concerns of implementing a machine learning model have to be addressed. Many machine learning frameworks could be used for the implementation, each with upsides and drawbacks. The original FDS model and the discrete Pixie Autoencoder models have some characteristics that affect the modelling choices. The most important one is the dynamic nature of graph input data. Each graph can, in theory, have a range of sizes and shapes. Since the structure, of both the world model and the inference network, depends on graph topology, each graph has to pass through a custom network architecture and parameter subset. First of all, the depth of the network depends on the maximum allowed graph size, since graph convolutions need a corresponding number of layers to fully propagate the information between the most distant nodes in the input graph. Furthermore, the weights used in the world model depend on node type, for example, entity and event nodes have to load and use different parameter sets for mean vectors. The size of the precision matrix in the world model also directly depends on the number of nodes in the graph. Additionally, many architectural quirks are dynamic, such as a two-word graph that might have an ARG1 or ARG2 semantic relation which influences which weights have to be used in the encoder. Due to these concerns, previous work by Emerson (2020) utilized the Dynet machine learning framework (Neubig et al., 2017) that focuses on dynamic construction of highly dynamic computation graphs. The same approach is also used here and the code base is directly based on the previous implementation.

5.2 Merging the objective functions

Another major concern is the memory and time efficiency of the model. Since the Pixie Autoencoder network is relatively complex and uses three separate models that jointly learn from the training data, the synchronization of the learning process is a major concern. The discrete Pixie Autoencoder system utilized a split learning approach, where different parts of the model would optimize separate loss functions and even be fed modified data of the same training example, to fine-tune the learning process. Specifically, the world model, inference network, and the lexical model would be trained on separate loss functions with selective use of dropout and a range of other techniques to get the best performance from each part of the model. This presented an interesting question to

explore regarding the possibility to simplify and merge the training process so that the whole model directly backpropagates and optimizes a single loss function encompassing all parameters of the three models as is traditional for variational autoencoders. It is not intuitively obvious that this approach would be superior in terms of performance, but it seemed worthwhile exploring if a simpler model with less tinkering could also perform well while being simpler to deploy and iterate on. The simplifications in moving away from the RBM parametrization and the introduction of Gaussian latent space provided a lot of opportunities for simplifying the combined loss. Thus, the final model trained on the single loss function presented above (4.30) calculates the gradients for all parts of the model simultaneously. Nonetheless, since the parameters of different model components range quite drastically in sensitivity, the new model preserved the use of separate optimizer instances that allow for different learning rates to be used on different components and give more flexibility in hyper-parameter tuning while optimizing a joint loss function.

5.3 Sampling

The size of the vocabulary used in the experiment was limited to a filtered collection of mostly nouns and verbs following the set-up in the original model. Nonetheless, the vocabulary stands at over 80 000 words in size, each with a corresponding semantic function network. Evaluating all the truth values for all the predicates, at each data point in the training set, in the process of obtaining normalized predicate probability is computationally expensive and can be seen as wasteful as most of the evaluations serve only to compute the exact probability ratio with more predicates delivering diminishing returns in precision. Thus, this implementation follows the approach in the discrete version of the model that uses negative sampling in evaluating the probability of the input predicate. Under this scheme, only a random subset of predicates is used in approximating the probability of the correct predicate reconstruction error as shown in (4.29). This also means that only the weights of the semantic functions involved in the evaluation of the probability at each step get updated. This approximation can be motivated by the observation that using all the negative predicates would decrease the probabilities of all the predicates, regardless of their frequency in the data, thus potentially overwhelming the less frequent and rarely occurring predicates. This problem, however, also appears in the case of totally random sampling. A **frequency proportional sampling** approach is utilised to use each predicate as a negative sample in proportion to its frequency in the data, thus balancing the negative and positive updates it receives.

5.4 Dropout

Another relevant component for successful training of the model is the use of dropout. Since the inference network can directly propagate information from input predicates to the encoded pixie distribution, there is a serious risk of the learning process failing to fully utilize contextual information from the rest of the graph and thus not learning the relations between words as fully as possible. To deal with this, the discrete Pixie Autoencoder model utilized selective dropout where two versions of the graph would be inferred. In the one passed to the inference network, strong dropout would be used so that a third of the predicates would be hidden from the encoder and would thus have to be inferred purely on contextual information. At the same time, the pixie graph would be inferred on the full input graph and passed to the lexical model so that it could learn its role without interruption. The merging of the loss functions dictated that this approach

be dropped in favour of using dropout for all parts of the model. This is, however, quite dangerous as it means the errors in one part of the network can cascade to the performance of the other parts. Thus, to examine the effect of dropout on the model, a version was trained with and without dropout with the results presented below.

5.5 Regularization

In parameterising the multivariate Gaussian distribution via the precision matrix, another important concern emerged. The covariance matrix Σ and consequently its inverse, the precision matrix Q, has to be positive semi-definite for the probability distribution to be well-defined. This issue is bigger than originally imagined since it introduces an additional constraint on the weights that can be used. The first instinct was to find a way to explicitly enforce this constraint in training. One idea would be to test whether the precision matrix is positive semi-definite at each update step and penalise it in proportion. This approach was, however, problematic since most approaches to test for the positive semi-definite property, such as eigenvalue decomposition and Cholesky decomposition, have cubic $O(n^3)$ complexity that is simply too expensive for evaluation at every step of the learning process, especially with matrices that regularly hold over 100 000 values. A new approach by Archakov and Hansen (2020) of using a vector to parameterise the precision matrix that can then get computationally converted to a valid positive semi-definite matrix was seriously considered, despite the $O(n^3\log(n))$ complexity, as the full reconstruction might not be needed at each step and the resulting matrix would have correctness guarantees but was after some practical testing shown to also be unacceptably slow. Some additional exact methods were considered but turned out not to be compatible with the Dynet framework and had to be abandoned. However, the simplest and cheapest method of normalizing the off-diagonal parameters of the matrix to be small relative to the diagonal proved very effective. Under this scheme, the traditional L2 regularization used to prevent exploding gradients in machine learning systems was used on parameters of all three model components and was enough to prevent issues with the precision matrix with minimal additional computational overhead. While this solution does not provide a guarantee of parameters never reaching an invalid state, the practical tests demonstrated it to be more than sufficient to avoid any training issues. A more efficient way of explicitly introducing the positive semi-definite constraint on the precision matrix in this model could nonetheless be a very interesting direction to explore in future work on the model.

5.6 GPU support

The discrete Pixie Autoencoder model was not compatible with GPU support due to issues with sampling from the RBM model that were not parallelizable. It was a hope for the continuous model to overcome this issue by using a more standard Gaussian latent space. This, unfortunately, turned out to be impossible due to the limitations of the GPU support of the Dynet library (Neubig et al., 2017) when it comes to computing a matrix determinant. In addition, the tensor contraction used in the deeper semantic function formulation also lacks GPU support under the Dynet framework. These issues are, however, framework dependent and are not fundamental limitations of the proposed continuous model. Thus, for future work the GPU support for the proposed model should be available, provided a framework with GPU support for noted operations, such as Tensorflow (Abadi et al., 2015), is used.

Chapter 6

Evaluation

To evaluate the performance of the newly proposed model, a set of evaluation criteria had to be chosen. In doing so, the goal was to select an evaluation method that is directly comparable to previous work. Since the most directly related model is the discrete pixie autoencoder model by Emerson (2020) the evaluation method was modelled to be as similar as possible to facilitate comparison. The rationale for using the RELPRON and GS2011 evaluation datasets is that they focus on contextual similarity not just standing meaning similarity and as such get to measure the model's capability of using context to infer word meaning.

6.1 Model training

This project also introduced a large set of modifications to the previous discrete method far extending the core replacement of the latent space parametrization, including the shift to single objective function training, changes to the dropout approach, several versions of deeper semantic functions, as well as latent space dimensionality. This presented a challenge in exploring the exact impact each change might have on the model performance in isolation. The model is also, despite the simplifications and speed improvements, still very experimental, CPU based and relatively computationally expensive. In addition, since the goal is for the model to encode the general lexical relations of a language, the training data has to be correspondingly large to get a strong sample of text data. This model was like its predecessors trained on WikiWoods dataset by Flickinger et al. (2010) based on Wikipedia subsequently parsed into DMRS graphs. The size of the dataset resulted in long training times with a single epoch passing through the entire dataset taking over two weeks of training on a powerful CPU server. This severely limited the number of experiments that could be performed even with good planning and minimal delays. The use of deeper semantic functions also drastically extended the training time to over four weeks due to slow Dynet performance on tensor contraction operations. For this reason, the decision was made to only test the deeper semantic function versions for short runs, demonstrating the model works correctly, which were unfortunately not remotely long enough to produce full results that can be reasonably compared to previous work. Thus the decision was made to use the limited number of runs to test key features of the new latent space parametrization using the single-layer semantic function version of the lexical model.

A training run with no dropout was done first, followed by 3 runs with dropout. Sampling was generally performed with 10,000 negative samples per data point. In one run the

sample number was reduced to 6000 and in one a dropout ratio of 0.25 was used to check if a change in any of the parameters radically changes the performance. The information about the training settings of all full runs is given in table (6.1). The full list of individual evaluation results is given in the appendix.

Run	Dropout ration	Sample number
1	0	10,000
2	0.3	6,000
3	0.25	10,000
4	0.3	10,000

Table 6.1: Hyper-parameters used for evaluated training runs

When the first run, that did not make use of dropout, was evaluated on both test metrics, it became apparent that dropout was a key element in forcing the model to learn the relations between words in the graph since the RELPRON score was too low at approximately 0.06 mean average precision while the GS2011 spearman rank was negative at approximately -0.04. This made it clear that the model needs dropout to function. Thus all 3 subsequent runs were performed with dropout. For a fair comparison with the previous work, the results from the three successfully runs were averaged when presented in all subsequent results discussion.

A small scale **speed test** run on a dummy graph dataset was performed to quantify the observation that the proposed model is faster than the discrete version. Over 19 epochs the average training time over 5 runs of the discrete version was 1m 49s while the proposed model took only 1m 17s on average which is a sizable improvement even without GPU support.

6.2 GS2011

The GS2011 dataset by Grefenstette and Sadrzadeh (2011) aims to evaluate word similarity in context. The dataset is composed of subject-verb-object triples paired with an alternative verb called landmark. For each such entry, a human-annotated rating of meaning similarity between versions using each verb is recorded. Human similarity rankings are given in range 1-7 from low to high similarity. There is a total of 199 distinct contextualized verb pairs with a total of 2500 judgements from different human annotators. The goal of the evaluation is to examine to what extent model similarity judgements match human perception.

The computation of similarity is here interpreted as logical inference of conditional probability of the landmark verb given a pixie inferred in the original version of the subject-verb-object graph. The logical inference could also be performed in both directions but, since the landmark often does not form a sensible sentence with the context words the one direction inference version is used here.

Since even human annotator similarity scores have a relatively high variance there are two ways to evaluate this dataset. One approach is to directly evaluate agreement against all human annotations separately but this means the score will reflect deviation for each annotator score on the same pair. The alternative is to average over human annotator scores for each entry to get a consensus similarity score to rank against. Since both methods have been used in previous work both were also used here to facilitate comparison.

6.2.1 Results and comparison

The results of a range of models on GS2011 data is given in table (6.2.1) which is divided into four sections. The first section shows performance on related models reported in the literature. Section two describes the performance of BERT model baselines produced for comparison by Emerson (2020). Section three shows the performance of previous Functional Distributional Semantics models on the task with the performance of the new Gaussian Pixie Autoencoder model given at the end. Averaging over the three successful runs over the data a separate score of 0.134 and an averaged score of 0.163 was obtained. This shows the model similarity scores do correlate with human judgements on verbs but much less than in the discrete version.

Comparison with previous work	Separate	Averaged
Vector addition (Milajevs et al., 2014)	-	.348
Categorical, copy object (Milajevs et al., 2014)	-	.456
Categorical, regression (Polajnar et al., 2015)	.33	-
Categorical, low-rank decomposition (Fried et al., 2015)	.34	-
Tensor factorisation (Van de Cruys et al., 2013)	.37	-
Neural categorical (Hashimoto et al., 2014)	.41	.50
BERT (contextual similarity) (Emerson, 2020)	.337	.446
BERT (contextual prediction) (Emerson, 2020)	.233	.317
Semantic functions (Emerson and Copestake, 2017)	.25	-
Sem-func vector ensemble (Emerson and Copestake, 2017)	.32	-
Pixie Autoencoder (both directions) (Emerson, 2020)	.306	.374
Pixie Autoencoder (one direction) (Emerson, 2020)	.406	.504
Gaussian Pixie Autoencoder	**0.134**	**0.163**

Table 6.2: Comparison of Spearman's rank correlation performance on GS2011 dataset (Previous work adapted from (Emerson, 2020))

6.3 RELPRON

RELPRON dataset is a relative clause evaluation data set for Compositional Distributional Semantics. It contains a selection of subject and object focused relative clauses. It was created by Rimell et al. (2016) for evaluation of methods in distributional semantics. The full dataset is composed of 1,087 properties where each property contains a hypernym of the term, modified by a relative clause. Properties are paired up with 138 terms with 10 properties per term present in the dataset. The challenge posed by the dataset is to match the terms with their corresponding properties. For example, the term *cell* would have a property *room that prison have*. There are two varieties of properties in the dataset based on whether they contain a subject relative clause or an object relative clause. The difference between them is which word in the clause is replaced by the hypernym of the term the subject or the object.

The evaluation is performed by building DMRS graphs for all properties and inferring them to pixie graphs. Then the pixie of the hypernym is isolated and all target word semantic functions in the dataset are run on the pixie producing a list of truth-value probabilities. The table of truth values is then sorted by the magnitude of the truth-value so that the mean average precision (MAP) can be calculated. Mean average precision here measures the degree to which all correct properties are given high rankings in the

truth value probability table.

6.3.1 Results and comparison

Since there was little time for fine-tuning on the development set, the evaluation is performed only on the test set. The average MAP score achieved by the model over the three successful runs was 0.275, which is quite good. The most direct comparison can be made with the discrete version of the model that achieved a 0.189 MAP score so the new Gaussian formulation represents a sizable improvement.

#	Comparison with previous work	MAP
A	Vector addition (Rimell et al., 2016)	.472
	Simplified Practical Lexical Function (Rimell et al., 2016)	.497
	Vector addition (Czarnowska et al., 2019)	.475
	Dependency vector addition (Czarnowska et al., 2019)	.439
B	BERT (masked prediction) (Emerson, 2020)	.186
	BERT (contextual prediction) (Emerson, 2020)	.134
	BERT (masked prediction) & vector ensemble (Emerson, 2020)	.479
C	Semantic functions (Emerson and Copestake, 2017)	.16
	Sem-func & vector ensemble (Emerson and Copestake, 2017)	.49
	Discrete Pixie Autoencoder (Emerson, 2020)	.189
	Discrete Pixie Autoencoder & vector addition (Emerson, 2020)	.489
D	Gaussian Pixie Autoencoder	**0.275**

Table 6.3: Comparison of mean average precision (MAP) performance on RELPRON data (Previous work adapted from (Emerson, 2020))

6.4 Discussion

The results unambiguously show that dropout is essential to the training process which has only been assumed in previous work. The performance of the new model on both evaluation metrics is satisfactory and demonstrates that the proposed model successfully learns to represent latent situations and uses contextual information to condition the parameters of the inferred latent distribution. The model outperformed both the discrete model and the BERT baselines on the RELPRON evaluation while underperforming the discrete model on the GS2011 dataset. In all experiments, updates were performed after each data point as opposed to batching, the use of which might also improve performance in future work. The low performance on GS2011 appears to be partly caused by groups of unrelated predicates using the same directions in the latent space, thus a higher latent dimensionality might be a simple solution. This might also be due to semantic functions being simple linear classifiers that, on a Gaussian, can only increase in any chosen direction. This means, that for two pixies embedded on a ray from the origin the more distant one either always gets a higher or lower truth value for all semantic functions with the parameter vector close to the ray's direction while they both get similar values for perpendicular semantic function parameters. This limits separability to the use of the direction of the pixie vector since the magnitude correlates between all semantic functions in the same direction. In future work, this might be improved by the use of deeper semantic functions as proposed in this thesis or a move to a spherical latent space such as proposed by Xu and Durrett (2018).

Chapter 7

Summary and Conclusions

This thesis presented the goals of computational lexical semantics through a discussion of previous work and presented the key challenges. Furthermore, a substantial overview of the Functional Distributional Semantics framework was given with both linguistic and computational motivations for different aspects of the framework. Next, the computational models based on Functional Distributional Semantics were presented and analysed in order to motivate the proposed extension of their approach to continuous spaces. As a key contribution, a new continuous Gaussian Pixie Autoencoder model was proposed and explained. The proposed model was then implemented and evaluated against previous work. The proposed model was shown to be effective and on one metric outperformed its discrete predecessor. The proposed model also improves on the predecessor in terms of simplicity and speed which could be further improved by GPU integration that the model should now support. The move to continuous spaces should also ease integration with most modern NLP utilities based on continuous lexical representations thus being a great starting point for further work in the field.

Appendix A

Extended results

Run	MAP
1	0.0657527150605203
2	0.3407683938318077
3	0.2120301848203641
4	0.2729841924678909
Mean	**0.275260923707**

Table A.1: Performance of individual runs on the RELPRON evaluation task

Run	Separate	Averaged
1	-0.04116861023637214	-0.0722938384692256
2	0.11706847316929977	0.1443943458758427
3	0.10926838991068903	0.1221158694050367
4	0.17458944446937177	0.2210730272309125
Mean	**0.133642102516**	**0.162527747504**

Table A.2: Performance of individual runs on the GS2011 evaluation task

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Ilya Archakov and P. Hansen. A new parametrization of correlation matrices. *arXiv: Econometrics*, 2020.

Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings, 2014.

Jose Camacho-Collados and Mohammad Taher Pilehvar. From word to sense embeddings: A survey on vector representations of meaning, 2018.

Ting-Yun Chang and Yun-Nung Chen. What does this word mean? explaining contextualized embeddings with natural language definition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6064–6070, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1627. URL https://www.aclweb.org/anthology/D19-1627.

Ann Copestake. **Invited Talk:** slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 1–9, Athens, Greece, March 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E09-1001.

Ann A. Copestake, Guy Edward Toh Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and E. Muszyńska. Resources for building applications with dependency minimal recursion semantics. In *LREC*, 2016.

Paula Czarnowska, Guy Emerson, and Ann Copestake. Words are vectors, dependencies are matrices: Learning word embeddings from dependency graphs. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 91–102, Gothenburg, Sweden, May 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-0408. URL https://www.aclweb.org/anthology/W19-0408.

Donald Davidson. The logical form of action statements.". In Nicholas Rescher and

Alan Ross Anderson, editors, *The Logic of Decision and Action*. Pittsburgh]University of Pittsburgh Press, 1966.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Carl Doersch. Tutorial on variational autoencoders, 2021.

Guy Emerson. Autoencoding pixies: Amortised variational inference with graph convolutions for functional distributional semantics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3982–3995, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.367. URL https://www.aclweb.org/anthology/2020.acl-main.367.

Guy Emerson and Ann Copestake. Functional distributional semantics. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 40–52, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-1605. URL https://www.aclweb.org/anthology/W16-1605.

Guy Emerson and Ann Copestake. Semantic composition via probabilistic model theory. In *IWCS 2017 - 12th International Conference on Computational Semantics - Long papers*, 2017. URL https://www.aclweb.org/anthology/W17-6806.

Guy Edward Toh Emerson. *Functional Distributional Semantics: Learning Linguistically Informed Representations from a Precisely Annotated Corpus (Doctoral thesis)*. PhD thesis, 2018.

Katrin Erk. What is word meaning, really? (and how can distributional models help us describe it?). In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 17–26, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W10-2803.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. Wikiwoods: Syntacto-semantic annotation for english wikipedia. In *In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC10*, 2010.

Gottlob Frege. Sense and reference. *The Philosophical review*, 57(3):209–230, 1948. ISSN 0031-8108.

Daniel Fried, Tamara Polajnar, and Stephen Clark. Low-rank tensors for verbs in compositional distributional semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 731–736, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2120. URL https://www.aclweb.org/anthology/P15-2120.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html.

Jesús González-Rubio, Daniel Ortiz-Martínez, José-Miguel Benedí, and Francisco Casacuberta. Interactive machine translation using hierarchical translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 244–254, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1025.

Edward Grefenstette and Mehrnoosh Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D11-1129.

Stevan Harnad. The symbol grounding problem. *Physica. D*, 42(1-3):335–346, 1990. ISSN 0167-2789.

Zellig S. Harris. Distributional structure. *¡i¿WORD¡/i¿*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL https://doi.org/10.1080/00437956.1954.11659520.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1544–1555, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1163. URL https://www.aclweb.org/anthology/D14-1163.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and R. Navigli. Sensembed: Learning sense embeddings for word and relational similarity. In *ACL*, 2015.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, Aug 2016. ISSN 1573-4951. doi: 10.1007/s10822-016-9938-8. URL http://dx.doi.org/10.1007/s10822-016-9938-8.

Yoon Kim. Convolutional neural networks for sentence classification, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

Kevin Lund and Curt Burgess. Producing high-dimensional semantic space from lexical co-occurence. *Behavior Research Methods Instruments Computers*, 28:203–208, 06 1996. doi: 10.3758/BF03204766.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 100–111, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1012. URL https://www.aclweb.org/anthology/K17-1012.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013a.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, pages 746–751, Atlanta, Georgia, June 2013b. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N13-1090.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1079. URL https://www.aclweb.org/anthology/D14-1079.

G. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, 1995.

George Miller, R. Beckwith, Christiane Fellbaum, Derek Gross, and K. Miller. Wordnet: An on-line lexical database. *Communications of the ACM*, 38, 07 2008.

Kevin Murphy. *Machine Learning: A Probabilistic Perspective*, volume 58. 01 2012.

Kevin P. Murphy. Multivariate gaussians, 2007. URL https://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall07/reading/gauss.pdf.

Roberto Navigli and Federico Martelli. An overview of word and sense similarity. *Natural Language Engineering*, 25(6):693–714, 2019. doi: 10.1017/S1351324919000305.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

Tamara Polajnar, Laura Rimell, and Stephen Clark. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-2701. URL https://www.aclweb.org/anthology/W15-2701.

François Recanati. Compositionality, flexibility, and context-dependence. In Wolfram Hinzen, Edouard Machery, and Markus Werning, editors, *Oxford Handbook of Compositionality*, pages 175–191. Oxford University Press, 2012.

Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. RELPRON: A Relative Clause Evaluation Data Set for Compositional Distributional Semantics. *Computational Linguistics*, 42(4):661–701, 12 2016. ISSN 0891-2017. doi: 10.1162/COLI_a_00263. URL https://doi.org/10.1162/COLI_a_00263.

John R. Searle. *Minds, Brains and Science*. Harvard University Press, 1984.

Kevin Swersky, Daniel Tarlow, Ilya Sutskever, Ruslan Salakhutdinov, Richard S. Zemel, and Ryan P. Adams. Cardinality restricted boltzmann machines. In *Advances in Neural Information Processing Systems 25*, Advances in Neural Information Processing Systems, pages 3293–3301, 2012. ISBN 9781627480031. Copyright: Copyright 2013 Elsevier B.V., All rights reserved.; 26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012 ; Conference date: 03-12-2012 Through 06-12-2012.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. A tensor-based factorization model of semantic compositionality. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1142–1151, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N13-1134.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding, 2015.

Jiacheng Xu and Greg Durrett. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1480. URL https://www.aclweb.org/anthology/D18-1480.

Jonathan Yedidia, William Freeman, and Yair Weiss. *Understanding belief propagation and its generalizations*, volume 8, pages 239–269. 01 2003. ISBN 1558608117.