CLASP Papers in Computational Linguistics

Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017)

Simon Dobnik and Shalom Lappin (eds.)

Gothenburg, 12–13 June 2017

ISSN XXXX-XXXX

CLASP Papers in Computational Linguistics

Volume 1: Proceedings of the Conference on Logic and Machine Learning in Natural Language (LaML 2017), Gothenburg, 12–13 June 2017, edited by Simon Dobnik and Shalom Lappin

University of Gothenburg

2017-11-21

e-publication available at:
http://hdl.handle.net/2077/54911

Distribution:

Centre for Linguistic Theory and Studies
in Probability (CLASP)
Department of Philosophy, Linguistics and
Theory of Science (FLOV)
University of Gothenburg
Box 200, SE-405 30 Gothenburg
http://www.clasp.gu.se

CLASP Papers in Computational Linguistics

http://hdl.handle.net/2077/54899

LAML 2017 Website

http://goo.gl/YkXSKg

Acknowledgements

The conference and this volume was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at Department of Philosophy, Linguistics and Theory of Science (FLoV), University of Gothenburg.

Preface

The past two decades have seen impressive progress in a variety of areas of AI, particularly NLP, through the application of machine learning methods to a wide range of tasks. With the intensive use of deep learning methods in recent years this work has produced significant improvements in the coverage and accuracy of NLP systems in such domains as speech recognition, topic identification, semantic interpretation, and image description generation.

While deep learning is opening up exciting new approaches to longstanding, difficult problems in computational linguistics, it also raises important foundational questions. Specifically, we do not have a clear formal understanding of why multi-level recursive deep neural networks achieve the success in learning and classification that they are delivering. It is also not obvious whether they should displace more traditional, logically driven methods, or be combined with them. Finally, we need to explore the extent, if any, to which both logical models and machine learning methods offer insights into the cognitive foundations of natural language.

The aim of the Conference on Logic and Machine Learning in Natural Language (LAML) was to initiate a dialogue between these two approaches, where they have traditionally remained separate and in competition. It included invited talks by Marco Baroni (University of Trento and Facebook AI Research (FAIR)), Alexander Clark (King's College London), Devdatt Dubhashi (Chalmers Institute of Technology), Katrin Erk (University of Texas at Austin), Joakim Nivre (Uppsala University), Aarne Ranta (University of Gothenburg), and Mehrnoosh Sadrzadeh (Queen Mary University of London). In addition, there were 9 peer-reviewed contributing papers that were accepted for presentation. The present volume contains a selection of extended papers based on the talks from the conference.

Simon Dobnik and Shalom Lappin

Gothenburg, Sweden

November 2017

Programme Committee

Marco Baroni	University of Trento and Facebook AI Research (FAIR)
Islam Beltagy	University of Texas at Austin
Jean-Philippe Bernardy	University of Gothenburg
Gemma Boleda	Universitat Pompeu Fabra
Stergios Chatzikyriakidis	University of Gothenburg
Alexander Clark	Kings College London
Robin Cooper	University of Gothenburg
Simon Dobnik	University of Gothenburg
Devdatt Dubhashi	Chalmers Institute of Technology
Katrin Erk	University of Texas at Austin
Julian Hough	Bielefeld university
Christine Howes	University of Gothenburg
John D. Kelleher	Dublin Institute of Technology
Shalom Lappin	University of Gothenburg
Staffan Larsson	University of Gothenburg
Julian Michael	University of Washington
Joakim Nivre	Uppsala University
Stephan Oepen	University of Oslo
Barbara Plank	University of Groningen
Matthew Purver	Queen Mary University of London
Aarne Ranta	University of Gothenburg
Mehrnoosh Sadrzadeh	Queen Mary University of London
Anders Sgaard	University of Copenhagen
Charalambos Themistocleous	University of Gothenburg

Table of Contents

On Bridging Mechanistic and Phenomenological Models with Deep Neural Networks in Natural Language Processing

Simon Dobnik
CLASP and FLOV
University of Gotenburg, Sweden
simon.dobnik@gu.se

John D. Kelleher
ADAPT Centre for Digital Content Technology
Dublin Institute of Technology, Ireland
john.d.kelleher@dit.ie

Abstract

Natural language processing (NLP) can be done using either top-down (theory driven) and bottom-up (data driven) approaches, which we call *mechanistic* and *phenomenological* respectively. The approaches are frequently considered to stand in opposition to each other. Examining some recent approaches in deep learning we argue that deep neural networks incorporate both perspectives and, furthermore, that leveraging this aspect of deep learning may help in solving complex problems within language technology, such as modelling language and perception in the domain of spatial cognition.

1 Introduction

There are two distinct methodologies to build computational models of language or of world in general. The first approach can be characterised as qualitative, symbolic and driven by domain theory (we will call this a *top-down* or *mechanistic approach*), whereas the second approach may be characterised as quantitative, numeric and driven by data and computational learning theory (we will call this the *bottom-up* or *phenomenological approach*). In this context we are borrowing the terminology of *phenomenological model* from the literature on the Philosophy of Science where the term *phenomenological model* is sometimes used to describe models that are independent of theory (see for example (McMullin, 1968)), but more generally is used to describe models that focus on the observable properties (phenomena) of a domain (rather than explaining the hidden mechanisms relating these phenomena) (Frigg and Hartmann, 2017). For this paper we use the term phenomenological model to characterise models which are primarily driven by fitting to observable relationships between phenomena in a domain, as represented by correlations between features in a dataset sampled from the domain; as opposed to models that are derived from a domain theory of the interactions between domain features. The focus of this paper is to examine and frame the potentially synergistic relationship between these distinct analytic methods for natural language processing (NLP) in the light of recent advances in deep neural networks (DNNs) and deep learning.

In historic terms this discussion is recurrent throughout the history of NLP. For example, early approaches such as (Shieber, 1986; Alshawi, 1992) are mechanistic in nature as they are based on logic and other formal approaches such as features structures and unification which are tools that allow formalisation of domain theories. With the availability of large corpora in mid-1990s there was a shift to data-driven phenomenological approaches with a focus on statistical machine learning methods (Manning and Schütze, 1999; Turney et al., 2010). This inspired several discussions on the relation between the two approaches (e.g., (Gazdar, 1996; Jones et al., 2000)). We share the view of some that both approaches are in fact in a complimentary distribution with each other as shown in Table 1 (adapted from a slide by Stephen Pulman). Mechanistic approaches provide deep coverage but of a limited domain; outside a domain they prove brittle and therefore limited. On the other hand, phenomenological approaches are wide-coverage and robust to variation found in data but provide shallow representation of language.

Our desiderata is a wide-coverage system with deep analyses. It was considered that this could be achieved by a hybrid model but working out such a model has proven not a trivial task. Systems that used both approaches treated them normally as in-

tech/cov	wide	narrow
deep	**our goal**	symbolic
shallow	data-based	**useless**

Table 1: Properties of mechanistic and phenomenological approaches in NLP

dependent black-boxes organised in layers (e.g. (Kruijff et al., 2007)). However, the marked recent advances in the NLP based on *deep* (!) neural networks have made the question of how these two methodologies should be used, related and integrated in NLP research apposite.

The choice of a method depends on the goal of the task for which it is used. One goal for processing natural language is to develop useful applications that help humans in their daily life, for example machine translation and speech recognition. In application scenarios where a rough analysis is acceptable (e.g., a translation that provides the gist of the message) and large annotated and structured corpora are available, machine learning is the methodology of choice to address this goal. However, where precise analysis is required or where there is a scarcity of data, a machine learning approach may not be suitable. Furthermore, if the goal of processing language is rather motivated by the desire to better understand its cognitive foundations, than a machine learning methodology, particularly one based on an unconstrained, fully connected deep neural networks, is not appropriate. The criticisms of unconstrained neural network based models (typically characterised by fully-connected feed-forward multi-layer networks) in cognitive science has a long history (see (Massaro, 1988) *inter alia*) and often focuses on (i) the difficultly in analysing in a domain-theoretic sense how the model works, and (ii) the, somewhat ironic, scientific short-coming that neural networks are such powerful and general learning mechanisms that demonstrating the ability of a network to learn a particular mapping or a function is scientifically useless from a cognitive science perspective. In particular, as Massaro (1988) argues, a neural network model is so adaptable that given the appropriate dataset and sufficient time and computing power it is likely to be able to learn mappings that not only support a cognitive theory but also ones that contradict that theory. One approach to address this problem is to introduce domain relevant structural constraints into the model via the network architecture, early approaches include (Feldman et al., 1988; Feldman, 1989; Regier, 1996). Indeed, we argue in this paper that one of the important and somewhat overlooked factors driving the success of research in deep learning is the specificity and modularity of deep learning architectures to the tasks they are applied too.

Contribution: In this paper we evaluate the relation between mechanistic and phenomenological models and argue that although it appears that the former have lost their significance in computational linguistics and its applications they are still very much present in the form of formal language modelling that underlines most of the current work with machine learning. Moreover, we highlight that many of the recent advances in deep learning for NLP are not based on unconstrained neural networks but rather that these networks have task specific architectures that encode domain-theoretic considerations. In this light, the relationship between mechanistic and phenomenological models can be viewed as potentially more synergistic. Given that many logical theories are defined in terms of *functions* and *compositional* operations and neural networks learn and compose functions, a logic-based domain theory of linguistic performance can naturally inform the structural design of deep learning architectures and thereby merge the benefits of both in terms of model interpretability and performance.

Overview: In Section 2, we discuss recent developments in deep learning approaches in NLP and situate them within the current debate; then, in Section 3, we use the computational modelling of spatial language as an NLP case study to frame the possible synergies between formal models and machine learning and set out our thoughts for potential approaches to developing a more synergistic understanding of the formal models and machine learning for NLP research. In Section 4 we give our concluding thoughts.

2 Deep Learning: A New Synthesis?

In recent years deep learning (DL) models have improved or in some cases markedly improved the state of the art across a range of NLP tasks. Some of the drivers of DL success include: (i) the availability of large datasets, (ii) more powerful computers, and (iii) the power of learning and adapt-

ability of connectionist neural networks. However, another and less obvious driver of DL is the fact that (iv) DL network models often have architectures that are specifically tailored or structured to the needs of a specific domain or task. This fact becomes obvious when one considers the variety of DL architectures that have been proposed in the literature. For example, a schematic overview of neural network architectures can be found at at: http://www.asimovinstitute.org/neural-network-zoo/ (van Veen, 2016).

2.1 Modularity in Deep Learning Architectures

There are a large-number of network design parameters that may be driven by experimental results rather than domain theory. For example, (i) the size of the network, (ii) the depth of the layers, (iii) the size of the matrices passed between the layers, (iv) activation functions and (v) optimiser are all network parameters that are often determined through an empirical trial-and-error process that is informed by designer intuition (Jozefowicz et al., 2016). However, the diversity of current network architectures extends beyond differences in these parameters and this diversity of network architecture is not a given. For example, given the flexibility of neural networks, one approach to accommodating structure into the processing of a network is to apply minimal constraints on the architecture and to rely on the ability of the learning algorithm to induce the relevant structure constraints by adjusting the network's weights.

On the other hand, it has, however, long been known that pre-structuring a neural network by the careful design of its architecture to fit the requirements of the task results in better generalisation of the model beyond the training dataset (LeCun, 1989). Understood in this context, DL is assisted (or supervised!) by the task designer in terms of a priori background knowledge who decides what kind of networks they are going to build, the number of layers, what kind of layers, the connectivity between the layers and other parameters. DL is most frequently not using fully connected layers, instead several kinds of layered networks have been developed tailored to the task. In this respect DL models capture top-down domain informed specification that we have seen with the rule-based NLP systems. This flexibility of neural networks is ensured by their modular design which takes as a basis a single perceptron unit which can be thought of encoding a simple concept. When several units are organised and connected into larger collections of units, these may be given interpretations that we give to symbolic representations in rule-based systems. The level of conceptual supervision may thus vary from no-supervision when fully connected layers are used, to weak supervision that primes the networks to learn particular structures, to strong supervision where the structure is given and only parameters of this structure are trained.

An example of weak supervision are Recurrent Neural Networks (RNNs) that capture sequence learning required for language models. The design of current state-of-the-art RNN language models is informed by linguistic phenomena such as short- and long-distance dependencies between linguistic units. In order to improve the ability of RNNs to model long-distance dependencies, contemporary RNN language models use Long-Short Memory Units (LSTM) or Gated Recurrent Units (GRUs) which may be further augmented with attention mechanisms (Salton et al., 2017). The inputs and outputs of such networks can be either characters or words, the latter represented as word embeddings in vector spaces.

Another example of weakly supervised neural networks, in the sense that their design is informed by a domain, are Convolutional Neural Networks (CNNs) which have their origin in image processing (LeCun, 1989). In CNNs the convolutions are meant as filters that encode a region of pixels into a single neural unit which learns to respond to the occurrence of such pixels as a specific visual feature. Importantly, the weights associated with a specific convolution are shared across a group of neurons such that together the group of neurons check for the occurrence of the visual features across the full surface of the image. Additionally, as objects or entities may occur in different parts of image, to decrease the effects of spatial continuum, operations such as pooling are used that encode convolved representations from various parts of the image. In analogy to learning visual features, CNNs have also been used for language modelling to capture different patterns of characters in strings (Kim et al., 2016).

Specialised networks may be treated as modules which are sequenced after each other. For example, the current Neural Machine Transla-

tion (NMT) architecture is the encoder-decoder (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Kelleher, 2016). This architecture uses one RNN, known as the encoder, to fully process the input sentence and generate its vector based representation. This is passed to a second RNN, the decoder, which implements a language model of the target language which generates the translation word by word. Domain theoretic considerations have affected the design how the two language modelling networks are connected in a number of ways. For example, an understanding that different languages have different word orders lead to enabling the decoder to look both back and forward along the input sentence during translation. This is implemented by fully processing the input sequence with the first RNN before translation is generated by the second RNN. However, the understanding of the need for local dependencies between different sections of the translation and somewhat a contrary requirement to the need for a potentially global perspective on the input has resulted in the development of attention mechanisms within the NMT framework. This means that DL network architectures modules are not only sequenced but they are also stacked. A variant of the NMT encoder-decoder architecture that replaces the encoder RNN with a CNN has revolutionised the field of image captioning (Xu et al., 2015). Figure 1 gives a schematic representation of such image captioning systems. The CNN module learns to represent images as vector representations of visual features and the RNN module is a language model whose output is conditioned on the visual representations. We have already mentioned that CNNs are also used to generate word representations. These representations are then passed to a RNN model to predict the next word in the context of preceding words in the sequence (see (Kim et al., 2016)). The advantage of using a CNN module to learn word representation is that it enables to capture spelling variation of morphologically-rich languages or texts from social media that does not use standard spelling of words. This and also the preceding examples therefore illustrate how different levels of linguistic representations are modelled in modular DL architectures.

In summary, the design of a DL architectures, where DL networks are treated as composable modules, can constrain and guide a number of factors that are important in representing language and other modalities, in particular the hierarchical composition of features and the sequencing of the representations. Importantly, the neural representations that are used in these cases are inspired by rich work on top-down rule-based mechanistic natural language processing.

2.2 Phenomenological versus Mechanistic Models

The ability to treat neural networks as composable modules within an overall system architecture is a powerful one. This is because during training it is possible to back-propagate the error through each of the system's modules (networks) and train them in consort while permitting each module to learn its distinctive task in parallel with the other modules in the network. However, the power of this approach has led to some research being based on a relatively shallow understanding of domain theory and most of the work being spent on fitting the hyper-parameters of the training algorithm through a grid-search driven by experimental performance on gold-standard datasets. The domain theory is only used to inform the broad outlines of the system architecture. Using image-captioning as an example, and at the risk of presenting a caricature, this approach may be described as: "we are doing image-captioning so we need a CNN to encode the image and an RNN to generate the language and we will let the learning algorithm sort out the rest of the details".

This theory free, or at least, theory light approach to NLP research is primarily driven by performance on gold-standard datasets and lamentably frequently the analysis of the systems is limited to the presentation of system results relative to a state-of-the-art leader-board with relatively little reflection on the how the structure of the model reflects theoretic considerations. This focus on performance in terms of accurately modelling the empirical relationship between inputs and outputs and where the trained model is treated as a black box aligns with what we describe as the *phenomenological tradition* in machine learning. This can be contrasted with an alternative tradition within machine learning which is sometimes described as being based on *mechanistic models*. Mechanistic models presuppose a domain theory and the model is essentially a computational implementation of this domain theory.

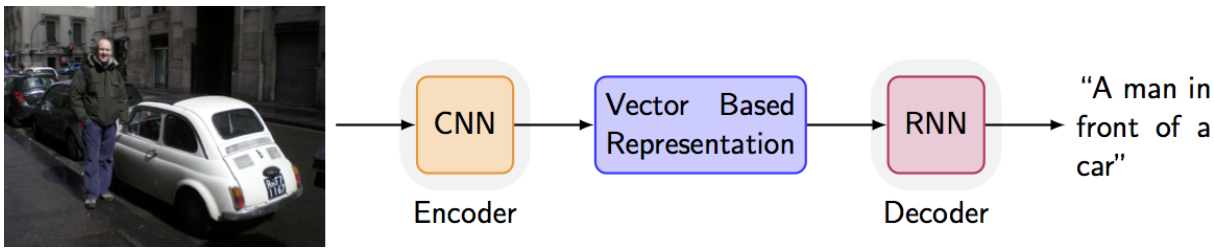


Figure 1: A schematic representation of DL image captioning architectures

To illustrate this difference, contrast for example the approach to training a support vector machine classifier where multiple kernels are tested until one with high performance on a dataset is found versus the approach to defining the topology of a Bayesian network in such a way that it mirrors a theory informed model of the causal relationships between relevant variables in the domain (Kelleher et al., 2015). Once the theoretical model has been implemented, the free parameters of the model can then be empirically fit to the data.

Consequently, mechanistic models are informed by both top-down theoretical considerations of a task designer but they are also sensitive to bottom-up empirical considerations, the training data. Mechanistic models have several advantages, for example: they can be used to test a domain theory. If the model is accurate, this provides evidence that the theory is correct. Assuming the theory is correct, they are likely to outperform phenomenological models in contexts where data is limited.[1] The top top-down approach provides background knowledge that restricts the size of the training search space.

Traditionally, neural networks have been considered the paradigmatic example of a phenomenological model. However, viewing neural networks as component modules within a larger deep-learning systems opens the door to sophisticated mechanistic deep-learning models. Such an approach to network design is, however, dependent on the system designer being informed by domain theory and is therefore strongly supervised in terms of background knowledge. An example of modular networks where each module is some configuration of neural units that are tailored to optimise parameters of a particular task is described in (Andreas et al., 2016) who work in the domain of question answering. The architecture learns how to map questions and visual or database representations to textual answers. In order to answer a question, the network learns a network layout of modules that are responsible for the individual steps required to answer the question. For example, to answer "What colour is the bird" the network applies the attention module to find the object from the question, followed by a module that identifies the colour of the attended region in the image. The possible sequences of modules are constrained by being represented as typed functions: in fact the modules translate to typed functional applications through which compositionality of linguistic meaning is ensured as in formal semantics (Blackburn and Bos, 2005). The system learns (using reinforcement learning) a layout model which predicts the sequence of modules to produce an answer for a question sentence and an execution module which learns how to ground a network layout in the image or database representation. An extension of this work is described in (Johnson et al., 2017) where both procedures rely on less background knowledge. For example, the system does not use a dependency parser to parse the input sentence but an LSTM language module and the modules use a more generic architecture.

The modular networks are in line with the *structured connectionism* of (Feldman et al., 1988) and *constrained connectionism* of Regier "in which complex domain-specific structures are built into the network, constraining its operation in clearly understandable and analysable ways" (Regier, 1996, p. 2). Regier's presentation of constrained connectionism is based on a case study on learning spatial relations and events. The case study describes the design and training of a neural network that receives short movies of 2 two-dimensional objects, a static rectangle and a circle which is either static or moving, as input and the model learns to predict the correct spatial term to describe the position and movement of the circle relative to the

[1] See discussion on generative versus discriminative models in (Kelleher et al., 2015).

rectangle. For example, a static circle might be described as *above* the rectangle, whereas a moving circle might move *out from under* the rectangle. A crucial aspect of this case study for Regier's argument is that the neural network's architecture is constrained in so far as it incorporates a number of structural devices that are motivated by neurological and psychological evidence concerning the human visual system, including motion buffers, angle and orientation computations components, and boundary and feature maps for objects in the input. Following (Regier, 1996), in the next section we will take spatial language as an NLP case-study and discuss how domain theory can be used to extend current deep-learning systems so as to move them further towards the mechanistic pole within the phenomenological versus mechanistic spectrum.

3 Spatial language

Our focus is computational modelling of spatial language, such as *the chair is to the left and close to the table* or *go down the corridor until the large painting on your right, then turn left*, which requires integration of different sources of knowledge that affect its semantics, including: (i) scene geometry, (ii) perspective and perceptual context, (iii) world knowledge about dynamic kinematic routines of objects, and (iv) interaction between agents through language and dialogue and with the environment through perception. Below we describe these properties in more detail:

Scene geometry is described within a two-dimensional or three-dimensional coordinate frame in which we can represent locations of objects as geometric shapes as well as angles and distances between them. Over a given area we can identify different degrees of applicability of a spatial description, for example with spatial templates (Logan and Sadler, 1996; Dobnik and Åstbom, 2017). A spatial template may be influenced by perceptual context through the presence of other objects in the scene known as distractors (Kelleher and Kruijff, 2005b; Costello and Kelleher, 2006), occlusion (Kelleher and van Genabith, 2006; Kelleher et al., 2011), and attention (Regier and Carlson, 2001).

Directionals such as *to the left of* require a model of *perspective* or *assignment of a frame of reference* (Maillat, 2003) which includes a viewpoint parameter. The viewpoint may be defined linguistically *from your view* or *from there* but it is frequently left out. Ambiguity with respect to the intended perspective of a reference can affect the grounding of spatial terms in surprising ways (Carlson-Radvansky and Logan, 1997; Kelleher and Costello, 2005). However, frequently the intended perspective can be either inferred from the perceptual context (if only one interpretation is possible, see for example the discussion on contrastive versus relative meanings in (Kelleher and Kruijff, 2005a)) or it may be linguistically negotiated and aligned between conversational partners in dialogue (Dobnik et al., 2014, 2015, 2016).

As mentioned earlier, spatial descriptions do not refer to the actual objects in space but to conceptual geometric representations of these objects, which may be points, lines, areas and volumes. The representation depends on how we view the scene, for example *under the water* (water $\approx$ surface) and *in the water* (water $\approx$ volume). The influence of *world knowledge* goes beyond object conceptualisation. Some prepositions are more sensitive to the way the objects interact with each (their dynamic kinematic routines) while other are more sensitive to the way the objects relate geometrically (Coventry et al., 2001).

Finally, because situated agents are located within dynamic linguistic and perceptual environments they must continuously adapt their understanding and representations relative to these context. On the language side they must maintain language coordination with dialogue partners (Clark, 1996; Fernández et al., 2011; Schutte et al., 2017; Dobnik and de Graaf, 2017). A good example of adaptation of contextual meaning through linguistic interaction is the coordinated assignment of frame of reference mentioned earlier.

In summary, the meaning of spatial descriptions is dynamic, dependent on several sources of contextually provided knowledge which provide a challenge for its computational modelling because of its contextual underspecification and because it is difficult to provide and integrate that kind of knowledge. On the other hand, a computational system taking into account these meaning components in context would be able to understand and generate better, more human-like, spatial descriptions and engage in more efficient communication in the domain of situated agents and humans. Furthermore, it could exploit the synergies between different knowledge sources to compensate miss-

ing knowledge in one source from another (Steels and Loetzsch, 2009; Skočaj et al., 2011; Schutte et al., 2017).

An example of a mechanistic neural model of spatial descriptions is described in (Coventry et al., 2005). Their system processes dynamic visual scenes containing three objects: a teapot pouring water into a cup and the network learns to optimise, for each temporal snapshot of a scene, the appropriateness score of a spatial description obtained in subject experiments. The idea behind these experiments is that descriptions such as *over* and *above* are sensitive to a different degree to geometric and functional properties of a scene, the latter arising from the interactions between objects as mentioned earlier. The model is split into three modules: (i) a vision processing module that deals with detection of objects from image sequences that show the interaction of objects, the tea pot, the water and the cup, using an attention mechanism, (ii) an Elman recurrent network that learns the dynamics of the attended objects in the scene over time, and (iii) a dual feed-forward vision and language network to which representations from the hidden layer of the Elman network are fed and which learns how to predict the appropriateness score of each description for each temporal configuration of objects. Each module of this network is dedicated to a particular task: (i) to recognition of objects, (ii) to follow motion of attended objects in time and (iii) to integration of the attended object locations with language to predict the appropriateness score, factors that have been identified to be relevant for computational modelling of spatial language and cognition through previous experimental work (Coventry et al., 2001). The example shows the effectiveness of representing networks as modules and their possibility of joint training where individual modules constrain each other.

The model could be extended in several ways. For example, contemporary CNNs and RNNs could be used which have become standard in neural modelling of vision and language due to their state-of-the-art performance. Secondly, the approach is trained on a small dataset of artificially generated images of a single interactive configuration of three objects.[2] An open question is how the model scales on a large corpus of image descriptions (Krishna et al., 2017) where considerable noise is added. There will be several objects, their appearance and location may be distorted by the angle at which the image is taken, there are no complete temporal sequences of objects and the corpora typically does not contain human judgement scores on how appropriate a description is given an image. Finally, Coventry et al.'s model integrates three modalities used in spatial cognition, but as we have seen there are several others. An important aspect is grounded linguistic interaction and adaptation between agents. For example, (Lazaridou et al., 2016) describe a system where two networks are trained to perform referential games (dialogue games performed over some visual scene) between two agents. In this context, the agents develop their own language interactively. An open research question is whether parameters such frame of reference intended by the speaker of a description could also be learned this way. Note that this is not always overtly specified, e.g. *from my left*.

Sometimes a mechanistic design of the network architecture constrains what a model can learn in undesirable ways. For example, Kelleher and Dobnik (2017) (in this volume) argue that contemporary image captioning networks as in Figure 1 have been configured in a way that they capture visual properties of objects rather than spatial relations between them. Consequently, within the captions generated by these systems the relation between the preposition and the object is not grounded in geometric representation of space but only in the linguistic sequences through the decoder language model where the co-occurrence of particular words in a sequence is estimated. (Dobnik and Kelleher, 2013, 2014) show that a language model is predictive of functional relations between objects that spatial relations are also sensitive to but in this case the geometric dimension is missing. This indicates that the architecture of these image-captioning systems, although modular, ignores important domain theoretic considerations and hence are best understood as close to the phenomenological (black-box) than the mechanistic (grey-box) network design philosophy this paper advocates.

In summary, it follows that an appropriate computational model of spatial language should consist of several connected modalities (for which individual neural network architectures are specified) but also of a general network that con-

[2] To be fair to the authors, their intention was not to build an image captioning system but to show that modular networks can optimise human experimental judgements.

nects these modalities, thus akin to the specialised regions and their interconnections in the brain (Roelofs, 2014). The challenge of creating and training such a system is obviously significant, however one feature of neural network training that may make this task easier is that it is possible to back-propagate through a pre-trained network. This opens the possibility of pre-training networks as modules (sometimes even on different datasets) that carry out specific theory-informed tasks and then training larger systems that represent the full-theory by including these pre-trained modules components within the system and training other modules and/or integration layers while keeping the weights of the pre-trained modules frozen during training.

4 Conclusion and future research

DNNs provide a platform for machine learning that permits great flexibility in combining top-down specification (in terms of hand-designed structures and rules) and data driven approaches. Designers can tailor the network structures to each individual learning problem and therefore effectively reach the goal of combining mechanistic and phenomenological approaches: a problem that has been investigated in NLP for several decades. The strength of DNNs is in the compositionality of perceptrons or neural units, and indeed networks themselves, which represent individual classification functions that can be combined in novel ways. This was not possible with other approaches in machine learning to the same degree with a consequences that these worked more as black boxes. Finally, although we are not advocating that there is a direct similarity between DNNs and human cognition, it is nonetheless the case that DNNs are inspired by neurons and connectionist organisation of human brain and hence at some high abstract level they share some similarities, for example basic classification units combine to larger structures, the structures get specialised to modules to perform certain tasks, and training and classification is performed across several modules. Therefore, this might be a possible explanation that DNNs have been so successful in computational modelling of language and vision, the surface manifestations of the underlying human cognition, as at some abstract level they represent a similar architecture to human cognition.

Acknowledgements

The research of Dobnik was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at Department of Philosophy, Linguistics and Theory of Science (FLoV), University of Gothenburg.

The research of Kelleher was supported by the ADAPT Research Centre. The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Funds.

References

Hiyan Alshawi. 1992. *The Core Language Engine*. ACL-MIT Press series in natural language processing. MIT Press, Cambridge, Mass.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT 2016*. Association for Computational Linguistics, San Diego, California, pages 1545–1554.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *arXiv preprint*. International Conference on Learning Representations, arXiv:1409.0473v7 [Cs.CL], pages 1–15.

Patrick Blackburn and Johan Bos. 2005. *Representation and inference for natural language. A first course in computational semantics*. CSLI Publications.

L.A. Carlson-Radvansky and G.D. Logan. 1997. The influence of reference frame selection on spatial template construction. *Journal of Memory and Langauge* 37:411–437.

Herbert H. Clark. 1996. *Using language*. Cambridge University Press, Cambridge.

Fintan Costello and John D. Kelleher. 2006. Spatial prepositions in context: The semantics of *Near* in the presense of distractor objects. In *Proceedings of the 3rd ACL-Sigsem Workshop on Prepositions*. pages 1–8.

Kenny R. Coventry, Angelo Cangelosi, Rohanna Rajapakse, Alison Bacon, Stephen Newstead, Dan Joyce, and Lynn V. Richards. 2005. Spatial prepositions and vague quantifiers: Implementing the functional geometric framework. In Christian Freksa, Markus Knauff, Bernd Krieg-Brückner, Bernhard Nebel, and Thomas Barkowsky, editors,

Spatial Cognition IV. Reasoning, Action, Interaction, Springer Berlin Heidelberg, volume 3343 of *Lecture Notes in Computer Science*, pages 98–110.

Kenny R. Coventry, Mercè Prat-Sala, and Lynn Richards. 2001. The interplay between geometry and function in the apprehension of Over, Under, Above and Below. *Journal of Memory and Language* 44(3):376–398.

Simon Dobnik and Amelie Åstbom. 2017. (Perceptual) grounding as interaction. In Volha Petukhova and Ye Tian, editors, *Proceedings of Saardial – Semdial 2017: The 21st Workshop on the Semantics and Pragmatics of Dialogue*. Saarbrücken, Germany, pages 17–26.

Simon Dobnik and Erik de Graaf. 2017. KILLE: a framework for situated agents for learning language through interaction. In Jörg Tiedemann and Nina Tahmasebi, editors, *Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa)*. Northern European Association for Language Technology (NEALT), Association for Computational Linguistics, Gothenburg, Sweden, pages 162–171.

Simon Dobnik, Christine Howes, Kim Demaret, and John D. Kelleher. 2016. Towards a computational model of frame of reference alignment in Swedish dialogue. In Johanna Björklund and Sara Stymne, editors, *Proceedings of the Sixth Swedish language technology conference (SLTC)*. Umeå University, Umeå, pages 1–3.

Simon Dobnik, Christine Howes, and John D. Kelleher. 2015. Changing perspective: Local alignment of reference frames in dialogue. In Christine Howes and Staffan Larsson, editors, *Proceedings of goDIAL – Semdial 2015: The 19th Workshop on the Semantics and Pragmatics of Dialogue*. Gothenburg, Sweden, pages 24–32.

Simon Dobnik and John Kelleher. 2014. Exploration of functional semantics of prepositions from corpora of descriptions of visual scenes. In *Proceedings of the Third V&L Net Workshop on Vision and Language*. Dublin City University and the Association for Computational Linguistics, Dublin, Ireland, pages 33–37.

Simon Dobnik and John D. Kelleher. 2013. Towards an automatic identification of functional and geometric spatial prepositions. In *Proceedings of PRECogSsci 2013: Production of referring expressions - bridging the gap between cognitive and computational approaches to reference*. Berlin, Germany, pages 1–6.

Simon Dobnik, John D. Kelleher, and Christos Koniaris. 2014. Priming and alignment of frame of reference in situated conversation. In Verena Rieser and Philippe Muller, editors, *Proceedings of DialWatt – Semdial 2014: The 18th Workshop on the Semantics and Pragmatics of Dialogue*. Edinburgh, pages 43–52.

J. A. Feldman, M. A. Fanty, and N. H. Goodard. 1988. Computing with structured neural networks. *Computer* 21(3):91–103.

Jerome A. Feldman. 1989. Structured neural networks in nature and in computer science. In Rolf Eckmiller and Christoph v.d. Malsburg, editors, *Neural Computers*, Springer, Berlin, Heidelberg, pages 17–21.

Raquel Fernández, Staffan Larsson, Robin Cooper, Jonathan Ginzburg, and David Schlangen. 2011. Reciprocal learning via dialogue interaction: Challenges and prospects. In *Proceedings of the IJCAI 2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*. Barcelona, Catalonia, Spain.

Roman Frigg and Stephan Hartmann. 2017. Models in science. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy (Spring 2017 Edition)*, Metaphysics Research Lab, Stanford University.

Gerald Gazdar. 1996. Paradigm merger in natural language processing. In Ian Wand and Robin Milner, editors, *Computing Tomorrow*, Cambridge University Press, New York, NY, USA, pages 88–109.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Fei-Fei Li, C. Lawrence Zitnick, and Ross B. Girshick. 2017. Inferring and executing programs for visual reasoning. In *arXiv preprint*. arXiv:1705.03633v1 [cs.CV], pages 1–13.

Karen I. B. Spärck Jones, Gerald J. M. Gazdar, and Roger M. Needham. 2000. Introduction: combining formal theories and statistical data in natural language processing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 358(1769):1227–1238.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. In *arXiv preprint*. arXiv:1602.02410v2 [cs.CL], pages 1–11.

John D. Kelleher. 2016. Fundamentals of machine learning for neural machine translation. In *Proceedings of the Translating Europen Forum 2016: Focusing on Translation Technologies*. European Commission Directorate-General for Translation. https://doi.org/10.21427/D78012.

John D. Kelleher and Fintan J. Costello. 2005. Cognitive representations of projective prepositions. In *Proceedings of the Second ACL-SIGSEM workshop on the linguistic dimensions of prepositions and their use in computational linguistics formalisms and applications*. Association for Computational Linguistics, University of Essex, Colchester, United Kingdom, pages 119–127.

John D. Kelleher and Simon Dobnik. 2017. What is not where: the challenge of integrating spatial representations into deep learning architectures. In *CLASP Papers in Computational Linguistics: Proceedings of the Conference on Logic and Machine Learning in*

Natural Language (LaML 2017). Gothenburg, Sweden, volume 1, pages 42–53.

John D. Kelleher and Geert-Jan M. Kruijff. 2005a. A context-dependent algorithm for generating locative expressions in physically situated environments. In Graham Wilcock, Kristiina Jokinen, Chris Mellish, and Ehud Reiter, editors, *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*. Association for Computational Linguistics, Aberdeen, Scotland, pages 1–7.

John D. Kelleher and Geert-Jan M. Kruijff. 2005b. A context-dependent model of proximity in physically situated environments. In *Proceedings of the Second ACL-SIGSEM workshop on the linguistic dimensions of prepositions and their use in computational linguistics formalisms and applications*. Association for Computational Linguistics, University of Essex, Colchester, United Kingdom.

John D Kelleher, Brian Mac Namee, and Aoife D'Arcy. 2015. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.

John D. Kelleher, Robert Ross, Colm Sloan, and Brian Mac Namee. 2011. The effect of occlusion on the semantics of projective spatial terms: a case study in grounding language in perception. *Cognitive Processing* 12(1):95–108.

John D. Kelleher and Josef van Genabith. 2006. A computational model of the referential semantics of projective prepositions. In P. Saint-Dizier, editor, *Syntax and Semantics of Prepositions*, Kluwer Academic Publishers, Dordrecht, The Netherlands, Speech and Language Processing.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. Phoenix, Arizona USA, pages 2741–2749.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123(1):32–73.

Geert-Jan M. Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. 2007. Situated dialogue and spatial organization: what, where... and why? *International Journal of Advanced Robotic Systems* 4(1):125–138.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2016. Multi-agent cooperation and the emergence of (natural) language. In *arXiv preprint*. arXiv:1612.07182v2 [cs.CL], pages 1–11.

Yann LeCun. 1989. Generalization and network design strategies. Technical report CRG-TR-89-4, Department of Computer Science, University of Toronto.

Gordon D. Logan and Daniel D. Sadler. 1996. A computational analysis of the apprehension of spatial relations. In Paul Bloom, Mary A. Peterson, Lynn Nadel, and Merrill F. Garrett, editors, *Language and Space*, MIT Press, Cambridge, MA, pages 493–530.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*. Lisbon, Portugal, pages 1412–1421.

Didier Maillat. 2003. *The semantics and pragmatics of directionals: a case study in English and French*. Ph.D. thesis, University of Oxford: Committee for Comparative Philology and General Linguistics, Oxford, United Kingdom.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. The MIT Press.

Dominic Massaro. 1988. Some criticisms of connectionist models of human performance. *Journal of Memory and Language* 27:213–234.

Ernan McMullin. 1968. What do physical models tell us? In Bob van Rootselaar and Johan Frederik Staal, editors, *Logic, Methodology and Science III: Proceedings of the Third International Congress for Logic, Methodology and Philosophy of Science, Amsterdam 1967*, North-Holland Publishing Company, pages 385–396.

Terry Regier. 1996. *The human semantic potential: Spatial language and constrained connectionism*. MIT Press.

Terry Regier and Laura A. Carlson. 2001. Grounding spatial language in perception: an empirical and computational investigation. *Journal of Experimental Psychology: General* 130(2):273–298.

Ardi Roelofs. 2014. A dorsal-pathway account of aphasic language production: The WEAVER++/ARC model. *Cortex* 59:33–48.

Giancarlo Salton, Robert Ross, and John D. Kelleher. 2017. Attentive language models. In *Proceedings of the 8th International Joing Conference on Natural Language Processing (IJCNLP)*. Taipei, Taiwan, pages 441–450.

Niels Schutte, Brian Mac Namee, and John D. Kelleher. 2017. Robot perception errors and human resolution strategies in situated human–robot dialogue. *Advanced Robotics* 31(5):243–257.

Stuart Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Publications, Stanford.

Danijel Skočaj, Matej Kristan, Alen Vrečko, Marko Mahnič, Miroslav Janíček, Geert-Jan M. Kruijff, Marc Hanheide, Nick Hawes, Thomas Keller,

Michael Zillich, and Kai Zhou. 2011. A system for interactive learning in dialogue with a tutor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2011*. San Francisco, CA, USA.

Luc Steels and Martin Loetzsch. 2009. Perspective alignment in spatial language. In Kenny R. Coventry, Thora Tenbrink, and John. A. Bateman, editors, *Spatial Language and Dialogue*, Oxford University Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Curran Associates, Inc., pages 3104–3112.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37(1):141–188.

Fjodor van Veen. 2016. The neural network ZOO. The Asimov Institute Blog posted on September 14. http://www.asimovinstitute.org/neural-network-zoo.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *arXiv preprint*. arXiv:1502.03044v3 [cs.LG], pages 1–22.

Neural TTR and possibilities for learning

Robin Cooper
University of Gothenburg
cooper@ling.gu.se

Abstract

One of the claims of TTR (Type Theory with Records) is that it can be used to model types learned by agents in order to classify objects and events in the world, including speech events. That is, the types can be represented by patterns of neural activation in the brain. This claim would be empty if it turns out that the types are in principle impossible to represent on a finite network of neurons. We will discuss how to represent types in terms of neural events on a network and present a preliminary implementation that maps types to events on a network. The kind of networks we will use are closely related to the transparent neural networks (TNN) discussed by Strannegård.

1 Introduction

Work on TTR, Type Theory with Records, (Cooper and Ginzburg, 2015; Cooper, 2017; Cooper, in prep) claims that it can be used to model types learned by agents in order to classify objects and events in the world.

In contrast to *traditional type theories* used in classical approaches to formal semantics (Montague, 1973; Montague, 1974), TTR is a *rich type theory* inspired by work developing from Martin-Löf (1984), called "modern type theory" by Luo (2010),Luo (2011). Traditional type theories provide types for basic ontological classes (e.g., for Montague: entities, truth values, time points, possible worlds and total functions between these objects) whereas rich type theories provide a more general collection of types, e.g. in our type theory, categories of objects such as *Tree*, types of situations such as *Hugging of a dog by a boy*.

Central to a rich type theory is the notion of *judgement* as in

> (An agent judges that) object a is of type T.

in symbols, $a : T$. We say that a is a *witness* for T. We build on this notion to put a cognitive spin on type theory, and say that perception involves a judgement that an object (possibly an event or, more generally, a situation) belongs to a certain type. Perception is constrained by the types to which an agent is attuned. This relates to ideas about visual perception proposed by Gibson (1986) which were influential in the development of situation semantics (Barwise and Perry, 1983).

We relate this simple minded view of perception to the kind of natural language interpretation which main stream semantics has taught us about and propose a view of linguistic evolution which roots linguistic ability in basic cognitive ability. The larger project is to do this in a way that incorporates results we have obtained from mainstream formal semantics but also in a way that can provide useful applications in robotic systems, including learning theories.

If we adopt this cognitive view of type theory, it seems that we should accept that the types can be represented by patterns of neural activation in the brain. The claim of the cognitive view would appear to be empty if it turns out that the types are in principle impossible to represent on a finite network of neurons. The aim of this paper is to suggest a way in which types could be represented neurally in principle rather than to make precise claims about the nature of their representation in agents' brains. Nevertheless it is our hope that the kind of techniques we are sketching are ultimately neurally plausible in the sense that they could be made consistent with what we know about biological brains.

The project we are engaged in here can be called *neuroscience fiction*. We cannot yet hope to observe brain activity corresponding to single types as conceived of in TTR. Available techniques such as FMRI do not have fine enough resolution and there is too much noise of other brain activity to easily identify exactly which neural activity corresponds to the perception of a situation where, for example, a boy hugs a dog. We see this work as an attempt to consider what a top-down approach to the neuroscience of perception and classification would be as opposed to the bottom-up approach which is available in current neuroscience. Basically the idea is this: in the bottom-up approach you might show a subject a picture of a boy hugging a dog and see what is common in brain activity over a large number of trials; on the top-down approach you create a theoy which makes a prediction of brain activity corresponding to a boy hugging a dog and you then test the prediction in subjects shown a picture of a boy hugging a dog.

It will be central to our discussion here that what is involved in representing types neurally is a neural event rather than a piece of neural architecture. We will present a preliminary implementation (see `nu.ipynb` on `https://github.com/GU-CLASP/pyttr`) that maps types to types of events on a network. The kind of networks we will use are closely related to the transparent neural networks (TNN) discussed by Strannegård and Nizamani (2016). It may be helpful to emphasize some of the things we are *not* doing with this particular implementation: we are not engaging in a machine learning exercise but rather addressing the theoretical question of how types could be represented in a neurologically plausible network; we are not addressing the question of recognizing witnesses for types but just initially the representation of the types themselves. The question of learning to make judgements that certain situations in the world are of these types *is* something where models of machine learning might be helpful and we will have some suggestions for how this could be approached later.

We will make some basic assumptions about neurons which seem to correspond to basic facts about typical neurons. A neuron consists of a body which carries out a computation on the basis of several inputs received on a number of dendrites connected to the body. A neuron has a single axon on which signals can be sent on the basis of the computation performed on the input received on the neuron's dendrites. While the neuron only has a single axon this axon may be connected to a large number of dendrites of other neurons by means of a number of axon terminals branching from the axon. The connection between an axon terminal and a dendrite is known as a synapse and the synapse itself may have some computational power. The input on a dendrite can correspond to a real number whereas the output on an axon (based on a computation of dendritic input) is boolean: either the neuron fires or it does not. We can think of the computation carried out by a synapse as converting a boolean to a real number. A simplified representation of a neural state is a characterization of which neurons have active axons, that is, which neurons have an output of 1. For a neuron-scientist this description of what is going on in the brain may seem like oversimplification to the point of falsity. However, it will enable us to address some of the basic formal problems associated with representing types as neural activation.

2 The binding problem

In TTR 'hug(a,b)' is known as a *ptype* (a type constructed from a predicate together with its arguments). Intuitively it is a type of situation in which a hugs b. The binding problem refers to making sure that one can distinguish between the type of events where a hugs b, the ptype 'hug(a,b)' in TTR, and the type of events where b hugs a, 'hug(b,a)' (Shastri, 1999; Kiela, 2011). A minimal solution to this is to designate an event involving the activation of a single neuron to represent each of 'hug', a and b. A more realistic encoding would most likely be events involving several neurons for each of these but the activation of a single neuron will be sufficient for the purposes of this discussion. An initial proposal for the neural representation of the ptype might be a neural event in which each of the neural events associated with the predicate and the arguments occurs in turn. In the neural TTR implementation this is displayed as the history of activation on a network as in:

```
a       0  0  1  0  0
b       0  0  0  1  0
hug_n   0  1  0  0  0
```

Here each row indicates a neuron given a convenient label in the first column. Subsequent columns indicate whether the neuron is firing at

successive time-steps. However, we do not wish to rely on neurons firing in a certain order but rather rely on the phasing of neurons with other neurons in a way similar to that originally suggested by Shastri (1999). This means that we add neurons that will correspond to predicate and argument roles and also a neuron that will be active throughout the neural event coding that the three separate neural events group together. The pattern of activity on the network thus looks like this:

a	0	0	1	0	0
b	0	0	0	1	0
hug_n	0	1	0	0	0
ptype2	*	1	1	1	0
rel	*	1	0	0	0
arg0	*	0	1	0	0
arg1	*	0	0	1	0

Here the activation of the neuron labelled 'ptype2' encodes that a two-argument ptype is represented from time-steps 2–4 with the relation at time-step two and the two arguments at the subsequent time-steps. Thus while we are exploiting the fact that the ptype is encoded as an event over several time-steps in order to solve the binding problem, it is no longer important exactly which order the events occur in. The 4th–7th rows in this display correspond to what we might call "book-keeping" neurons which are used to indicate the structure of represented types, as opposed to the "content" neurons represented in the first three rows. If the system discovers during the course of a computation that not enough book-keeping neurons are available it will create those needed in order to carry out the computation. In the implementation this represents an expansion of the number of neurons in the network and it is indicated in this display by the occurrences of '*' in the first column indicating that these neurons did not exist at the first time-step. While neurogenesis (structural plasticity) is a known phenomenon — see Maguire et al. (2000), Maguire et al. (2006) for a discussion of the relative sizes of the hippocampus in London taxi drivers as compared with London bus drivers — it does not seem reasonable to assume that human brains actually grow during the course of a computation in this way, but we might take this expansion of the network to model a use of previously unused neurons in order to carry out a novel computation.

From this simple example, three potential basic principles of neural representation emerge:

- *neural events* (with phasing) are important for neural representation (rather than just neural architecture or snapshots of the network at a single time-step)

- *neural event types* can be *realized differently* on different networks, *cf.* Fedorenko and Kanwisher (2009), Fedorenko and Kanwisher (2011). Which neurons are dedicated to a particular purpose can vary from network to network and depends in part on the order in which things are presented to the network.

- We can expect a kind of *compositionality* in neural representations. For example, whatever pattern of activation a network uses to represent 'hug' (firing of a single neuron or multiple neurons), that pattern of activation will occur in phase with a 'rel' pattern of activation in representing a ptype with 'hug'.

3 The recursion problem

As a simple illustration of the kind of recursion needed by linguistic representations we will show examples where ptypes can occur as arguments within ptypes as in:

believe(c, hug(a,b))

know(d, believe(c, hug(a,b)))

One aspect of this recursion that can be challenging for neural representation is that there is in princple no upper limit on the depth of embedding that can be obtained. Another challenge is that various components may be repeated at various points in the structure as in:

believe(a, believe(b, hug(a,b)))

Thus the phasing of neurological events in a representation of this has to be such that a single object can play several distinct roles in the representation. The technique we developed for coding ptypes as neurological events in order to solve the binding problem is in fact adequate to deal with the recursion problem as well. Here is a trace of a network event representing believe(c, hug(a,b)):

a	0	0	0	0	1	0	0	0
b	0	0	0	0	0	1	0	0
hug_n	0	0	0	1	0	0	0	0
ptype2	0	1	1	1	1	1	1	0
rel	0	1	0	0	0	0	0	0
arg0	0	0	1	0	0	0	0	0
arg1	0	0	0	1	1	1	1	0
c	0	0	1	0	0	0	0	0
believe_n	0	1	0	0	0	0	0	0
ptype2	*	0	0	1	1	1	0	0
rel	*	0	0	1	0	0	0	0
arg0	*	0	0	0	1	0	0	0
arg1	*	0	0	0	0	1	0	0

This is the first time that this network has seen an embedding of a ptype within a ptype and it therefore adds an additional set of book-keeping neurons for a two-place ptype. Note that the ptype2 neuron represented in row 4 is active from time-step 2 to time-step 7 whereas the ptype2 neuron in row 10 is active from time-step 4 to time-step 6 within the period of activation of the arg1 neuron represented in row 7. What we have here is thus a rather straightforward encoding of structure in a two-dimensional binary matrix. Given that the network is capable of growing in order to accommodate greater depths of embedding there is in principle no limit on the depth of embedding that it can handle except for (in the case of the implementation) available memory in the computer or (in the case of a natural brain) availability of neurons that can be dedicated to book-keeping. This is in contrast to the kind of neural network representation of recursion provided by, for example, Christiansen and Chater (1999) which is limited to a finite number of embeddings. On the other hand we have only looked at representation and said nothing about learning. This makes it difficult to make any meaningful comparison with the literature on neural networks at this point.

The importance of recursion and the compositional approach to neural representation is further illustrated by the treatment of dependent types as functions which return a type, for example a ptype. Such functions can be of arbitrary depth (e.g. functions which return a function which returns a type and so on). Also we treat generalized quantifiers in terms of ptypes whose arguments are dependent types. Thus we can have a situation where we have a ptype within which is a function and within the function is a ptype construction. This is the kind of recursion which is common in linguistic structure. To illustrate how this works consider how we can create a dependent type which returns a ptype in `pyttr`.

```
T = DepType('v',Ind,
            PType(hug,['v','b']))
print(show(T))
```

This returns:

```
lambda v:Ind . hug(v, b)
```

Thus we have created a function from objects of type *Ind* (individual) to the ptype of situations where that individual hugs b. A neural event which represents this function has a neural event representing a ptype ('ptype2') temporally included in a neural event representing a function ('lambda'):

b	0	0	0	0	1	0	0
hug_n	0	0	1	0	0	0	0
ptype2	0	0	1	1	1	0	0
rel	0	0	1	0	0	0	0
arg0	0	0	0	1	0	0	0
arg1	0	0	0	0	1	0	0
lambda	*	1	1	1	1	1	0
dom	*	1	0	0	0	0	0
var	*	1	0	1	0	0	0
rng	*	0	1	1	1	1	0

We can represent the type of situation in which every dog runs as the ptype:

```
every(lambda x:Ind . dog(x),
      lambda x:Ind . run(x))
```

This type will be correspond to an neural event as illustrated in Figure 1.

4 Memory – a simple kind of learning

We have argued above that a good way to deal with the binding problem and the recursion problem in representing types in neural networks is to let the representations be neural events rather than pieces of neural architecture. The simple-minded idea is that when an agent classifies an object or event as being of a certain type the corresponding neural event will occur in the agent's brain. While there

every_n	0	1	0	0	0	0	0	0	0	0	0	0	0
dog_n	0	0	0	1	0	0	0	0	0	0	0	0	0
run_n	0	0	0	0	0	0	0	0	1	0	0	0	0
Ind_n	0	0	1	0	0	0	0	1	0	0	0	0	0
ptype2	*	1	1	1	1	1	1	1	1	1	1	1	0
rel	*	1	0	0	0	0	0	0	0	0	0	0	0
arg0	*	0	1	1	1	1	1	0	0	0	0	0	0
arg1	*	0	0	0	0	0	0	1	1	1	1	1	0
lambda	*	0	1	1	1	1	0	1	1	1	1	0	0
dom	*	0	1	0	0	0	0	1	0	0	0	0	0
var	*	0	1	0	1	0	0	1	0	1	0	0	0
rng	*	0	0	1	1	1	0	0	1	1	1	0	0
ptype1	*	0	0	1	1	0	0	0	1	1	0	0	0
rel	*	0	0	1	0	0	0	0	1	0	0	0	0
arg0	*	0	0	0	1	0	0	0	0	1	0	0	0

Figure 1: "every dog runs"

seem to be good reasons to think of the representations as events rather than architecture, it seems initially puzzling how such an agent could store a type in memory. In the TTR literature we talk of agents as having types available as resources which can be used to make judgements about objects and situations. In particular Cooper et al. (2015) talk about estimating probabilistic judgements based on previous judgements. How could such judgements be stored in memory if they are just represented as neural events?

Our proposed solution uses an idea from TNN where a single neuron which is *top-active* in the sense of TNN (Strannegård and Nizamani, 2016) can be regarded as encoding a concept since it is triggered by a complex activity corresponding to that concept. Here we will turn the idea around and create a single *memory* neuron which when excited will trigger a neural event representing a type. This is a simple way of "freezing" a neural event in a network in architectural terms. The memory neuron must be connected to other neurons in the network in a way so that its activation will occasion an orderly progression of neural events in sequence with the correct phasing. This is achieved by introducing *delay* neurons (Strannegård et al., 2015) which can be used to delay passing on a signal an arbitrary number of timesteps. For an interesting account of delay circuitry in nature see Schöneich et al. (2015). As an illustration Figure 2 shows the trace of a network with a memory neuron (labelled 'every dog runs' in the display) for the type

$$\text{every}(\lambda x{:}\textit{Ind} \,.\, \text{dog}(x),\ \lambda x{:}\textit{Ind} \,.\, \text{run}(x))$$

Delay neurons, like other bookkeeping neurons, are added as required in the process of creating the memory. Notice that our treatment of quantification in terms of generalized quantifiers where the *every* is a predicate holding between two properties means that we can reuse our method for encoding ptypes in this more complex example involving quantification. Currently memories of judgements are implemented by activating a neuron represented by an object in phase with the representation of a type, though we suspect that something more like the method used for ptypes will ultimately be necessary. Figure 3 shows an example where a particular event 'e' is judged to have the type in Figure 1. Note that while the neuron labelled 'e:every(dog,run)' could be said to encode an *Austinian proposition* in memory in something like the sense discussed by Cooper et al. (2015) it says absolutely nothing about what has to happen in the world (or in the agent's peceptual apparatus) in order for this memory to be formed.

5 Prospects for more complex learning

One way of incorporating learning into this system is to interface it to a conventional machine learning system. We currently plan to create an interface to the KILLE system (de Graaf and Dobnik, 2015) which learns classifications of objects and spatial relations between objects on the basis of visual input through Kinect (`https://developer.microsoft.`

every_n	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
dog_n	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
run_n	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Ind_n	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
ptype2	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
rel	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
arg0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
arg1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
lambda	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0
dom	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
var	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0
rng	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0
ptype1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0
rel	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
arg0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
every dog runs	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 2: Running the memory "every dog runs"

com/en-us/windows/kinect) and linguistic input. It seems that it would be straightforward to map the final linguistic output from KILLE to a TTR type that could be represented on a network in the way that we have suggested. More interesting perhaps would be to map lower level outputs from this system directly to the activity patterns which neural TTR associates with a type for a given network. Below is a small example of an activity pattern generated by neural TTR for a judgement that a is an individual, that is, a : *Ind*:

```
[[(0, 'Ind_n', 1), (1, 'a', 1)],
 [(0, 'Ind_n', 0), (1, 'a', 0)]]
```

An activity pattern is a list of lists of triples. The first member of the triple is a unique identifier for a neuron on a given network. The second member is the intuitive label for the neuron which is provided only for the sake of human readibility and the third boolean value indicates whether the neuron should be turned on or off. Each list of tuples represents one time-step. It should be a straightforward exercise to learn a mapping from Kinect output to such triples which can then be realized on the network. This would then be a two-level system which uses conventional machine learning possibly involving non-transparent networks for low level learning and a transparent network of the kind we have described for high level representation. Such systems would raise the question of how far down it would be possible or desirable to go before converting to the high level representations using neural TTR.

Another approach we are considering is to build basic learning strategies using techniques based on reinforcement learning into the transparent neural TTR network. Here the idea would be to simulate the embodiment of a network in a body for which some actions stimulate a pleasure circuit in the network whereas other actions stimulate a pain

every_n	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
dog_n	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
run_n	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Ind_n	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
e	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
ptype2	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0
rel	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
arg0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
arg1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
lambda	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
dom	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
var	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0
rng	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0
ptype1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0
rel	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
arg0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
e:every(dog,run)	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
Delay	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Figure 3: Judgement that every dog runs in e

circuit. The network would evolve in a way that would avoid painful actions and seek pleasurable ones.

We suspect that a combination of both of these strategies might ultimately be useful.

6 Conclusion

In this paper we have suggested a way in which types as discussed in TTR could be represented as neural events in a network. Representing types as neural events rather than neural architecture enabled us to give simple-minded solutions to the problem of binding and the problem of recursion where the fact that the network can grow (or adjust itself) during the course of computation seems important for the latter. We also suggested a way in which this event approach to representation can be made compatible with storage in memory by introducing memory neurons which when activated will give rise to appropriate events. The introduction of delay circuitry was important for this.

This proposal, rather like formal semantics, does not say anything about the way in which representations of such types could be grounded in actual experience. In the final section we suggested a couple of strategies for addressing this and relating it to machine learning and we plan to explore this in future work.

Acknowledgments

This work is supported by VR project 2013-4873, Networks and Types.

References

Jon Barwise and John Perry. 1983. *Situations and Attitudes*. Bradford Books. MIT Press, Cambridge, Mass.

Morten H Christiansen and Nick Chater. 1999. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23(2):157 – 205.

Robin Cooper and Jonathan Ginzburg. 2015. Type theory with records for natural language semantics. In Shalom Lappin and Chris Fox, editors, *The Handbook of Contemporary Semantic Theory*, pages 375–407. Wiley-Blackwell, second edition.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. 2015. Probabilistic Type Theory and Natural Language Semantics. *Linguistic Issues in Language Technology*, 10(4):1–45.

Robin Cooper. 2017. Adapting type theory with records for natural language semantics. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, *Modern Perspectives in Type-Theoretical Semantics*, number 98 in Studies in Linguistics and Philosophy, pages 71–94. Springer.

Robin Cooper. in prep. Type theory and language: from perception to linguistic communication. Draft of book chapters available from `https://sites.google.com/site/typetheorywithrecords/drafts`.

Erik de Graaf and Simon Dobnik. 2015. KILLE: Learning Objects and Spatial Relations with Kinect. In *Proceedings of goDIAL - Semdial 2015: The 19th Workshop on the Semantics and Pragmatics of Dialogue*.

E. Fedorenko and N. Kanwisher. 2009. Neuroimaging of language: Why hasn't a clearer picture emerged? *Language and Linguistics Compass*, 3:839–65.

E. Fedorenko and N. Kanwisher. 2011. Functionally localizing language-sensitive regions in individual subjects with fMRI: A reply to Grodzinsky's critique of Fedorenko & Kanwisher (2009). *Language and Linguistics Compass*, 5(2):78–94.

James J. Gibson. 1986. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates.

Douwe Kiela. 2011. Variable Binding in Biologically Plausible Neural Networks. Master's thesis, Universiteit van Amsterdam.

Zhaohui Luo. 2010. Type-theoretical semantics with coercive subtyping. In *Proceedings of SALT 20*, pages 38–56.

Zhaohui Luo. 2011. Contextual Analysis of Word Meanings in Type-Theoretical Semantics. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *Logical Aspects of Computational Linguistics: 6th International Conference, LACL 2011*, number 6736 in Lecture Notes in Artificial Intelligence, pages 159–174. Springer.

Eleanor A. Maguire, David G. Gadian, Ingrid S. Johnsrude, Catriona D. Good, John Ashburner, Richard S. J. Frackowiak, and Christopher D. Frith. 2000. Navigation-related structural change in the hippocampi of taxi drivers. *Proceedings of the National Academy of Sciences*, 97(8):4398–4403.

Eleanor A. Maguire, Katherine Woollett, and Hugo J. Spiers. 2006. London taxi drivers and bus drivers: A structural mri and neuropsychological analysis. *Hippocampus*, 16(12):1091–1101.

Per Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis, Naples.

Richard Montague. 1973. The Proper Treatment of Quantification in Ordinary English. In Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 247–270. D. Reidel Publishing Company, Dordrecht.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven. ed. and with an introduction by Richmond H. Thomason.

Stefan Schöneich, Konstantinos Kostarakos, and Berthold Hedwig. 2015. An auditory feature detection circuit for sound pattern recognition. *Science Advances*, 1(8):e1500325–e1500325, sep.

Lokendra Shastri. 1999. Advances in SHRUTI – a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11(1):79–108.

Claes Strannegård and Abdul Rahim Nizamani. 2016. Integrating symbolic and sub-symbolic reasoning. In *International Conference on Artificial General Intelligence*, pages 171–180. Springer.

Claes Strannegård, Simone Cirillo, and Johan Wessberg. 2015. Emotional Concept Formation. In *Proceedings of the Eighth Conference on Artificial General Intelligence*, pages 166–176. Springer.

Subregular Complexity and Deep Learning

Enes Avcu
Department of Linguistics and Cognitive Science
University of Delaware
enesavc@udel.edu

Chihiro Shibata
School of Computer Science
Tokyo University of Technology
shibatachh@stf.teu.ac.jp

Jeffrey Heinz
Department of Linguistics
Institute of Advanced Computational Science
Stony Brook University
jeffrey.heinz@stonybrook.edu

Abstract

This paper argues that the judicial use of formal language theory and grammatical inference are invaluable tools in understanding how deep neural networks can and cannot represent and learn long-term dependencies in temporal sequences.

Learning experiments were conducted with two types of Recurrent Neural Networks (RNNs) on six formal languages drawn from the Strictly Local (SL) and Strictly Piecewise (SP) classes. The networks were Simple RNNs (s-RNNs) and Long Short-Term Memory RNNs (LSTMs) of varying sizes. The SL and SP classes are among the simplest in a mathematically well-understood hierarchy of subregular classes. They encode local and long-term dependencies, respectively. The grammatical inference algorithm Regular Positive and Negative Inference (RPNI) provided a baseline.

According to earlier research, the LSTM architecture should be capable of learning long-term dependencies and should outperform s-RNNs. The results of these experiments challenge this narrative. First, the LSTMs' performance was generally worse in the SP experiments than in the SL ones. Second, the s-RNNs out-performed the LSTMs on the most complex SP experiment and performed comparably to them on the others.

1 Investigating Deep Learning

This paper argues that formal language theory and grammatical inference can provide a systematic way to better understand the kinds of patterns deep learning networks (Goodfellow et al., 2016) are able to learn. The main ideas are illustrated with experiments testing how well two types of Recurrent Neural Networks (RNNs) can learn different kinds of simple, subregular formal languages with a grammatical inference algorithm serving as a baseline.

Using formal languages to investigate the learning capabilities of neural networks is not without precedent. Much earlier research also used formal languages to probe the learning capabilities of neural networks; Schmidhuber (2015, sec. 5.13) provides a review. Section 2 highlights some of this work and makes clear our own contribution.

Long-term dependencies in temporal sequences have a distinguished history in the development of neural network learning models and in generative linguistics. Bengio et al. (1994) define long-term dependencies this way: "A task displays long-term dependencies if prediction of the desired output at time t depends on input presented at an earlier time $\tau \ll t$." Many examples of such long-term dependencies abound in nature and engineering. For example, generative linguists, beginning with Chomsky (1956, 1957), have studied the grammatical basis of long-term dependencies in natural languages and have raised the question of how such dependencies are learned (Chomsky, 1965).

We test simple RNNs (s-RNNs) (Elman, 1990) and Long Short-Term Memory RNNs (LSTMs) (Hochreiter and Schmidhuber, 1997) on simple regular languages which encode local and long-term dependencies. Readers are referred to Goodfellow et al. (2016) and Goldberg (2017) for details of these two types of networks.

A common narrative in the deep learning literature is that LSTMs are a solution to learning long-term dependencies, which are problematic for s-RNNs. For example, Schmidhuber's (2015) review, which received the first Best Paper Award

ever issued by the journal *Neural Networks*, explains that "Typical deep NNs suffer from the now famous problem of vanishing or exploding gradients." He calls this "the fundamental deep learning problem of gradient descent." It is these vanishing or exploding gradients that prevent neural networks like s-RNNs from learning long-term dependencies. Schmidhuber explains how much subsequent research was dedicated to overcoming this problem and writes "LSTM-like networks ... alleviate the problem through a special architecture unaffected by it."

Similarly, writing in *Nature*, LeCun et al. (2015, p. 442) say "Although [RNNs] main purpose is to learn long-term dependencies, theoretical and empirical evidence shows that it is difficult to learn to store information for very long." They go on to write "LSTM networks have subsequently proved to be more effective than conventional RNNs" because LSTMs "use special hidden units, the natural behaviour of which is to remember inputs for a long time."

Therefore, we were particularly interested in understanding how well LSTMs can learn long-term dependencies within temporal sequences. We developed training and test data sets for formal languages drawn from the Strictly Local (SL) and Strictly Piecewise (SP) classes of formal languages. As will be explained in more detail in §3, SL and SP languages are simple regular languages which only encode local and certain types of long-term dependencies, respectively.

These formal languages are drawn from well-understood subclasses of the regular languages which form a complexity hierarchy (McNaughton and Papert, 1971; Rogers et al., 2010; Rogers and Pullum, 2011). These hierarchies measure the complexity of formal languages not in terms of automata-theoretic measures, such as the size of the minimal deterministic automaton, but instead on a model-theoretic basis (Enderton, 2001; Rogers et al., 2013). In other words, the complexity of a formal language is determined by the kind of logic and model-theoretic representation needed to specify it (Rogers and Pullum, 2011). As Rogers et al. (2013) explain, these classes also have a cognitive interpretation.

In the experiments, there were six target languages to learn: three SL and three SP. For each language, three training sets were prepared, and for each training set two test sets were prepared, for a total of 36 test sets. The training and test data was also controlled for word length so we could assess the networks' ability to generalize to strings longer than the ones in the training sample. The LSTMs and s-RNNs were trained on both positive and negative examples. We conducted several experiments, systematically varying the vector sizes in the networks. These experimental details are explained in §4.

The results, presented in §5, are unexpected given the narrative outlined above. The narrative would suggest that the s-RNNs and LSTMs may perform comparably on the SL experiments, but that s-RNN performance would be worse than LSTM performance on the SP experiments due to the presence of long-term dependencies. Furthermore, since the LSTMs are "unaffected" by the "fundamental deep learning problem," we may expect that the LSTM performance on the SP experiments to be comparable to the ones on the SL experiments.

Neither of these expectations were borne out. While the RNNs performed above chance in all of our experiments, they struggled learning the two most complex SP languages as compared to the matched SL languages. Furthermore, the s-RNNs performed comparably to the LSTMs in many of the SP experiments, and in fact outperformed the LSTMs on the most complex SP learning task. Also, both LSTMs and s-RNNs did relatively poorly on the simplest SL experiment.

When learning fails, it is natural to ask whether the training data was sufficiently rich for it to be reasonable for correct inference to take place. For this reasons, we also ran the grammatical inference algorithm Regular Positive and Negative Inference (RPNI) (Oncina and Garcia, 1992) on the test sets and examined its output. RPNI provably infers any regular language, provided the training data is sufficient. Readers are referred to de la Higuera (2010) for details on RPNI. When RPNI is successful, it means there is enough information in the training sample for correct inference to occur, at least for learning algorithms which only consider regular languages as targets. RPNI's results suggest that training data was sufficient in almost all of the SP experiments, but only in one-third of the SL experiments. This one-third includes the simple SL experiments where the RNNs struggled. The analysis with RPNI makes it harder to explain away the poor performance of the RNNs

on the grounds that the data was insufficient. Section 6 discusses this, and other aspects of the results in more detail.

Our conclusion is that the there is much more to be learned about the how RNNs represent and learn long-term dependencies in sequences. We believe that understanding how RNNs generalize from their training data will follow from connecting the behavior of RNNs to classes of formal languages like the ones here.

2 Motivation and background

In the 1990s, many studies aimed to learn formal languages with neural networks. When the aim was to predict the next symbol of a string drawn from a regular language, first-order RNNs were used (Casey, 1996; Smith, A.W., 1989). The target languages here were based on the Reber grammar (Reber, 1967). When the aim was to decide whether a string is grammatical, second-order RNNs were used (Pollack, 1991; Watrous and Kuhn, 1992; Giles et al., 1992). Here the target languages were the regular languages studied by Tomita (Tomita, 1982). Later research targeted nonregular languages (Schmidhuber et al., 2002; Chalup and Blair, 2003). One striking result established that LSTMs can learn some context-sensitive formal languages exhibiting long-distance dependencies with uncanny precision (Prez-Ortiz et al., 2003).

The reasons for making formal languages the targets of learning are as valid today as they were decades ago. First, the grammars generating the formal languages are known. Therefore training and test data can be generated as desired. Thus, the scientist can run controlled experiments to see whether particular generalizations are reliably acquired under particular training regimens.

Importantly, the relative complexity of different formal languages may provide additional insight. If it is found that formal languages of one type are more readily learned than formal languages of another type in some set of experiments then the difference between these classes may be said to meaningfully capture some property unavailable to the RNNs in the experiments. Subsequent work may lead to proofs and theorems about which properties of RNNs lead to the reliable inference of formal languages from certain classes and which do not. It may also lead to new network architectures which overcome identified hurdles.

There are two important differences between the present paper and past research, beyond the development in neural networks. First, the regular languages chosen here are known to have certain properties. The Reber grammars and Tomita languages were not understood in terms of their abstract properties or pattern complexity. While it was recognized some encoded long-term dependencies and some did not, there was little recognition of the computational nature of these formal languages beyond that. In contrast, the formal languages in this paper are much better understood. While *subregular* distinctions had already been studied by the time of that research (McNaughton and Papert, 1971), it went unrecognized how that branch of computer science could inform neural network learning.

The second difference is the advances in grammatical inference over the past few decades. The development of RPNI (Oncina and Garcia, 1992) essentially solved the problem of efficiently learning regular languages from positive and negative data. Other results addressed the learning of subregular classes from positive data only (Garcia et al., 1990; García and Ruiz, 2004; Heinz, 2010b; Heinz et al., 2012). Like the work on subregular complexity, what these analytical approaches to learning formal languages offered neural network researchers went unrecognized.

3 Subregular Complexity

Figure 1 shows proper inclusion relationships of well-studied classes of subregular languages. The Strictly Local (SL), Locally Testable (LT), and Non-Counting (NC) classes were studied by (McNaughton and Papert, 1971). The Locally Threshold Testable (LTT) class was introduced and studied by (Thomas, 1982). The Piecewise Testable (PT) class was introduced and studied by (Simon, 1975). The Strictly Piecewise (SP) class was studied by (Rogers et al., 2010). As many authors discuss, these classes are natural because they have multiple characterizations in terms of logic, automata, regular expressions, and abstract algebra. Cognitive interpretations of these classes also exist (Rogers and Pullum, 2011; Rogers et al., 2013).

From the perspective of natural language processing, SL is the formal language-theoretic basis of n-gram models (Jurafsky and Martin, 2008) and SP models aspects of phonology (Heinz, 2010a).

Next we define these classes, focusing on the

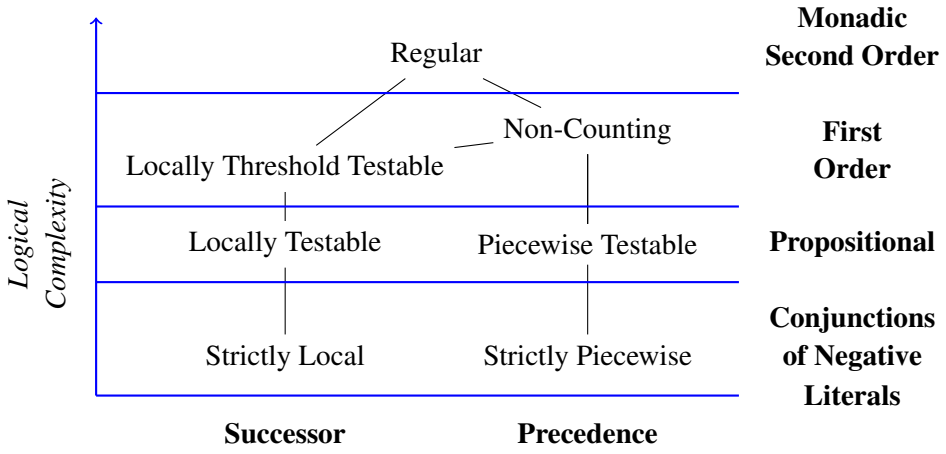


Figure 1: Subregular language classes with inclusion shown from the top down.

SL and SP classes since languages belonging to them form the learning targets in the experiments described in §4.

3.1 Mathematical Notation

Let Σ denote a finite set of symbols, the alphabet, and Σ^* the set of elements of the free monoid of Σ under concatenation. We refer to these elements both as *strings* and as *words*. The ith symbol in word w is denoted w_i. Left and right word boundary markers ($\rtimes$ and $\ltimes$, respectively) are symbols not in Σ. A stringset (also called formal language) is a subset of Σ^*.

If u and v are strings, uv denotes their concatenation. Similarly, if S_1, S_2 are stringsets then S_1S_2 denotes their concatenation and is equal to $\{uv \mid u \in S_1, v \in S_2\}$.

For all $u, v, w, x \in \Sigma^*$, if $x = uwv$ then then w is a *substring* of x. If $x \in \Sigma^* w_1 \Sigma^* w_2 \Sigma^* \dots w_n \Sigma^*$ then w is a *subsequence* of x. A substring (subsequence) of length k is called a k-factor (k-subsequence). Let $\texttt{factor}_k(w)$ denote the set of substrings of w of length k. Let $\texttt{subseq}_k(w)$ denote the set of subsequences of w up to length k. The domains of these functions extend to languages in the normal way.

3.2 Strictly Local Stringsets

A stringset L is Strictly k-Local (SL_k) iff whenever there is a string x of length $k-1$ and strings $u_1, v_1, u_2, v_2 \in \Sigma^*$, such that $u_1xv_1, u_2xv_2 \in L$ then $u_1xv_2 \in L$. We say L is *closed under suffix substitution*. L is SL if $L \in \text{SL}_k$ for some k (Rogers and Pullum, 2011).

As discussed in (McNaughton and Papert, 1971; Rogers and Pullum, 2011), SL_k languages can also be characterized by a finite set of k-factors as follows. Observe first that for each k, $\texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$ is finite. Let a SL_k grammar be a set of k-factors $G \subseteq \texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$. The language of G is the stringset $L(G) = \{w \mid \texttt{factor}_k(\rtimes w \ltimes) \subseteq G\}$. Thus, the grammar G is the set of *permissible* k-factors. Any k-factor w in $\texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$ which is not in G is thus *forbidden* and consequently all strings containing w as a substring are not in $L(G)$. Consequently, SL stringsets can also be defined with grammars that only contain finitely many forbidden k-factors as we do in §4. From a logical perspective, SL stringsets can thus be expressed as the conjunction of negative literals where literals correspond to a model-theoretic representation of strings where the order of the elements is given by a successor relation (Rogers and Pullum, 2011).

3.3 Strictly Piecewise Stringsets

A stringset L is Strictly k-Piecewise (SP_k) iff $\texttt{subseq}_k(w) \subseteq \texttt{subseq}_k(L)$ implies $w \in L$. L is SP if there is a k such that it belong to SP_k; equivalently, L belongs to SP iff L is closed under subsequence (Rogers et al., 2010). SP_k stringsets can also be defined with a finite set of k-subsequences (Rogers et al., 2010). In fact the parallel to SL_k is near perfect.

Observe the set $\texttt{subseq}_k(\Sigma^*)$ is finite. Let a

SP_k grammar be a set of k-subsequences $G \subseteq \texttt{subseq}_k(\Sigma^*)$. The language of G is the stringset $L(G) = \{w \mid \texttt{subseq}_k(w) \subseteq G\}$. The grammar G is the set of *permissible* k-subsequences. Any k-subsequence w in $\texttt{subseq}_k(\Sigma^*)$ which is not in G is thus *forbidden* so strings containing w as a subsequence are not in $L(G)$. Since for each k, $\texttt{subseq}_k(\Sigma^*)$ is finite, SP stringsets can also be defined with grammars containing forbidden k-subsequences as in §4. From a logical perspective, SP stringsets can thus be expressed as the conjunction of negative literals where literals correspond to a model-theoretic representation of strings where the order of the elements is given by the precedence relation (Rogers et al., 2013).

3.4 Locally and Piecewise Testable Classes

A stringset L is Locally k-Testable (LT_k) iff for all $w, v \in \Sigma^*$, it is the case that if $\texttt{factor}_k(\rtimes w \ltimes) = \texttt{factor}_k(\rtimes v \ltimes)$ then either $w, v \in L$ or $w, v \notin L$. In other words, membership of a string w in any LT_k stringset is determined solely by the set of k-factors in w. Similarly, a stringset L is Piecewise k-Testable (PT_k) iff for all $w, v \in \Sigma^*$, it is the case that if $\texttt{subseq}_k(\rtimes w \ltimes) = \texttt{subseq}_k(\rtimes v \ltimes)$ then either $w, v \in L$ or $w, v \notin L$. Here, membership of a string w in any PT_k stringset is determined solely by the set of k-subsequences in w. Stringsets are LT (PT) if there is some k such that they are LT_k (PT_k), respectively.

It can be proven that LT (PT) languages are those equivalent to a Boolean combination of finitely many SL (SP) formal languages. LT (PT) are also parameterized by a k value, which corresponds to the largest k-value among the SL languages they are a Boolean combination of.

From a logical perspective, the LT and PT classes can be defined with propositional statements over relational structures representing sequences where the order relation is given as successor and precedence, respectively, as shown in Figure 1 (Rogers et al., 2013).

3.5 Locally Threshold Testable, NonCounting, and Regular Classes

A stringset L is Locally Threshold Testable iff there are two numbers k and t such that for all strings $u, v \in \Sigma^*$ and k-factors $x \in \texttt{factor}_k(\{\rtimes\}\Sigma^*\{\ltimes\})$ if the number of times x occurs in u is the same as the number of times x occurs in v whenever this number is less than t or occurs at least t times in both u and v then either $u, v \in L$ or $u, v \notin L$. In other words, membership of a string w in any $\text{LTT}_{t,k}$ stringset is determined solely by the number of occurences each k factor occurs in w, counting them only up to some threshold t. The LT_k class equals $\text{LTT}_{1,k}$.

A stringset L is NonCounting iff there is a k such that for all $w, u, v \in \Sigma^*$, if $wuv \in L$ then $wu^{k+1}v \in L$. McNaughton and Papert (1971) prove languages in the NonCounting class are exactly those definable with star-free generalized regular expressions and exactly those obtained by closing LT stringsets under concatenation.

Languages in the LTT and NC classes can be defined with first-order statements over relational structures representing sequences where the order relation is given as successor and precedence, respectively (Thomas, 1982; McNaughton and Papert, 1971; Rogers et al., 2013). LTT is thus properly included in NC because successor is first-order definable with precedence but precedence is not first-order definable with successor.

Informally, a stringset L is regular if the resources required to decide whether a string u belongs to L is independent of the length of u. They can be defined as those formal languages recognizable by finite-state acceptors. Büchi (1960) showed these are exactly the stringsets definable with weak monadic second-order logic with the order relation given as successor (or precedence, since the precedence relation is MSO-definable from successor and vice versa). Stringsets that are regular but not NonCounting typically count modulo some n. For example, the stringset which contains all and only strings with an even number of `a`s is not NonCounting, but regular.

3.6 Further Comment

SL, SP, LT, PT, and LTT classes form infinite hierarchies of language classes based on k (Rogers and Pullum, 2011; Rogers et al., 2010). In particular for all $k \in \mathbb{N}$, $\text{SL}_k \subsetneq \text{SL}_{k+1}$, $\text{SP}_k \subsetneq \text{SP}_{k+1}$, $\text{LT}_k \subsetneq \text{LT}_{k+1}$, $\text{PT}_k \subsetneq \text{PT}_{k+1}$ and $\text{LTT}_{t,k} \subsetneq \text{LTT}_{t,k+1}$. It is also true that $\text{LTT}_{t,k} \subsetneq \text{LTT}_{t+1,k}$. Consequently, for any SL (SP/LT/PT/LTT) stringset, the smallest k value for which it is SL_k (SP_k/LT_k/PT_k/$\text{LTT}_{t,k}$) is another measure of its complexity.

We analyzed the Reber (1967) and Tomita (1982) languages and concluded the following. The Reber grammar is SL_3 and the embedded Reber grammar is LT_3. Tomita languages 1, 2, 3, 4,

5, 6, and 7 are SL_1, SL_2, regular, SL_3, regular, regular, and SP_4, respectively.

The subregular hierarchies provide a much more fine-grained meaning to the term *long-term dependency*. For example, SL dependencies are effectively bounded by a window of size k, but none of the other classes are limited in this way. Consequently the distinctions between them distinguish different *kinds* of long-term dependencies. This is why we are optimistic that much these subregular properties can meaningfully inform how RNNs represent and learn long-term dependencies.

4 The Experiments

Here we describe the target languages, the training data, the test sets, the neural network architectures and RPNI.

We implemented the RNNs with `Chainer` (`http://chainer.org`). RPNI was implemented using Matlab and the `gi-toolbox` `https://code.google.com/archive/p/gitoolbox` (Akram et al., 2010). The files we used with `Chainer`, `gi-toolbox`, and the ones we used to prepare the training and test sets are available online at `https://github.com/enesavc/subreg_deeplearning`.

4.1 Target Languages

In this study, six formal target languages were defined in order for training and testing purposes. In each case, we let $\Sigma = \{a, b, c, d\}$.

Table 1 defines the six formal languages we used in this study. Each of the SL languages was defined with four banned substrings and each of the SP languages was defined with one banned subsequence. We refer to these six languages by the class they belong to: SL2, SL4, SL8, SP2, SP4, and SP8.

These grammars were implemented as finite-state machines using `foma`, a publicy available, open-source platform (Hulden, 2009). The number of states in the minimal deterministic automaton recognizing these six languages are shown in Table 1.

4.2 Training data

Training data was generated with `foma`. For each language L we generated three training data sets, which we call 1k, 10k, and 100k because they contained 1,000, 10,000, and 100,000 words, respectively. Half of the words in each training set were positive examples (so they belonged to L) and half were negative examples (so they did not belong to L). Training words were between length 1 and 25. For the positive examples there were 20, 200, and 2,000 words of each length. For the negative examples, we wanted to provide 20, 200, and 2,000 words of each length, respectively. However, as the k value increases, there is no negative data for shorter words since all shorter words belong to L. In this case, we generated 20×k, 200×k, and 2,000×k of words of length k, and also 20, 200, and 2,000 words for the lengths between $k+1$ and 25. Training words were generated randomly using `foma`, so training sets contained duplicates.

4.3 Test sets

For each language L and each training regimen T for L, we developed two test sets, which we call Test1 and Test2. Test1 and Test2 contain 1,000, 10,000, or 100,000 words depending on whether T is 1k, 10k, or 100k, respectively. Half of the test words belong to L and half do not. Test1 and Test2 only contain novel words. Novel positive words belong to L but do not belong to the positive examples in T. Novel negative words neither belong to L nor to the negative examples in T.

The difference between the two test sets has to do with word length. Test1 words are no longer than 25. Test2 words are of length between 26 and 50. Thus while both sets test for generalization, Test2 importantly checks to what extent the generalizations are independent of word length.

For Test2, we used `foma` to randomly generate 20, 200, and 2,000 positive words and negative words for each length between 26 and 50.

For Test1, we used `foma` to randomly generate words whose length was less than 26. Again, we wanted to generate 20, 200, and 2,000 positive words and negative words for each length. However, there may not be positive or negative words for some of the shorter lengths. This is because either they do not exist for the target language L, or because they do exist for L but they also exist in the training data in which case they would not be novel. For positive (negative) words, we did the following. If there was at least one word of a particular length, we randomly generated 20, 200, or 2,000 words of that length depending on T. We also generated extra words of length 25 (specifically 200, 2,000, or 20,000). We concatenated

Table 1: The six target stringsets with $\Sigma = \{a, b, c, d\}$.

Language Class	Forbidden k-factors in target stringsets	Minimal DFA size
SL2	⋊b, aa, bb, a⋉	3
SL4	⋊bbb, aaaa, bbbb, aaa⋉	7
SL8	⋊bbbbbbb, aaaaaaaa, bbbbbbbb, aaaaaaa⋉	15
Language Class	Forbidden k-subsequences in target stringsets	
SP2	ab	2
SP4	abba	4
SP8	abbaabba	8

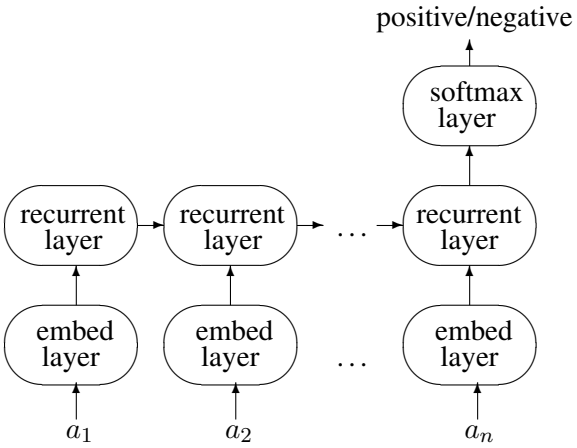


Figure 2: An unrolled RNN for predicting membership.

these together ordered by length into one file and then selected the first 500, 5,000, or 50,000 words to be the positive (negative) words in Test1. In other words, we effectively padded the Test1 with words of length 25 when there were no words of shorter lengths.

The order of the words in the test sets was randomized.

4.4 RNN Architectures

For the LSTMs and s-RNNs, we constructed simple networks to test the capability of the networks themselves. First we describe properties of the architectures shared by both RNN types (the LSTMs and s-RNNs) and then we discuss specific aspects of the LSTMs.

For each input string, the RNNs can be represented as a connected graph as shown in Figure 2. The output of the network is the probability of the input word belonging to the target language.

The output of the embed layer in Figure 2 is known as the distributed representation of the symbol, which is equivalent to a linear layer (with no bias) that maps the one-hot vector to a real-valued vector. The outputs of the recurrent layer corresponding to each symbol are ignored except for the last one. The output corresponding to the last symbol is mapped to a two dimensional vector through the softmax layer, whose elements represent the positive and negative probabilities.

Each value of the weights in the embed layer were independently initialized according to the normal distribution.

For all the RNNs in this study the vector sizes of the recurrent layer and the embed layer were made identical. For each type of RNN, we examined three networks of varying sizes, which we call v10, v30, and v100 because the vectors at each layer are of size 10, 30, and 100, respectively.

For the LSTMs, the standard architecture with forget gates and without peepholes was used among the various possible modifications (Greff et al., 2015). The recurent layer of the LSTMs includes four fully-connected layers called the input gate, the block input, the forget gate, and the output gate. The weights of those layers are initialized in the same manner as the embed layer except for the weights of the forget gate, which were initialized according to the normal distribution with mean 1 and variance 1.

4.5 RNN Training

When training the RNNs, the training data was divided into batches, and the gradient is calculated for each batch. The strings in each batch are usually chosen uniformly and randomly to avoid biased gradients. However, calculations become inefficient if strings are chosen completely randomly due to large differences in the lengths of the strings. Thus, we sort all strings first in order of their lengths, re-order partially, block them and choose those blocks randomly. The RNNs pro-

cessed the training data 100 times or 'epochs.'

Other training parameters are as follows. The batch size is 128. The L2 norm of the gradient is clipped with 1.0. The lengths of strings in each batch are aligned through padding an additional symbol whose embedding is fixed to the zero vector. The optimization algorithm called Adam (Kingma and Ba, 2014) is applied.

4.6 RPNI

RPNI takes as input a training set of positive and negative examples and outputs a deterministic finite-state acceptor (DFA), which accepts and rejects strings.

RPNI itself is deterministic. In other words, if RPNI is run on the same training data multiple times, the variance in its performance on the test sets will be zero. This is because for a given training set, it will always output the same DFA.

RPNI is a state-merging algorithm. It first builds a DFA which explicitly accepts and rejects only the positive and negative examples in training. This DFA is known as a prefix-tree because its states are in one-to-one correspondence with the prefixes of the strings in the training set. It then conducts a breadth-first traversal of the prefix tree, attempting to successively merge pairs of states. Two states are merged provided the resulting DFA is consistent with training set. Readers are referred to de la Higuera (2010) for more details on RPNI and to Heinz et al. (2015) for more general discussion of state-merging algorithms.

Oncina and Garcia (1992) proved that for each regular language L there is a finite set S^+ of positive examples and a finite set S^- of negative examples such that RPNI, when given any training set including S^+ and S^- as input, will output a minimal DFA A recognizing L. Furthermore, RPNI runs in cubic time with respect to the size of $S^+ \cup S^-$. Additionally, the size of $S^+ \cup S^-$ is also bounded by a polynomial with respect to the size of A. Readers are referred to de la Higuera (1997) and Eyraud et al. (2016) for more details on analyses of efficiency in grammatical inference.

5 Results

After each epoch, the RNNs were tested on the test sets. Inspection of the accuracy trajectories indicate that accuracy had stabilized in all cases well before the 100th epoch. Therefore, accuracy results are reported after the 100th epoch of training.

The networks were run 10 times and we report the mean accuracy results on each test set, along with the standard deviation. RPNI, being deterministic, was only run once. These results for the LSTMs and the s-RNNs and RPNI are given in Table 2 for the SL targets and in Table 3 for the SP targets. In these tables, each row corresponds to one test set with one training set for one target stringset. The mean value is given in each cell to three significant digits. The standard deviation is given in parantheses to two significant digits.[1] Boldfaced numbers in each row indicate the best performance. Chance performance is 50%.

RPNI successfully output the target grammar in 10 of the 18 training regimens. These were the SL2 10k, SL2 100k, SL4 100k, SP8 100k, and all of the SP2 and SP4 training sets.[2]

First we explain the results generally and then we make more specific comparisons. In all experiments the LSTMs and s-RNNs scored above chance indicating some learning took place. However, their mean results only outperformed RPNI in 9 of the 36 test sets. These are Test1 and Test2 in the SL4 1k, SL8 1k, SP8 1k and SP8 10k experiments, in addition to Test2 in the SL8 10k experiment.

Also, the accuracies on Test1 and Test2 are nearly the same for the LSTMs and s-RNNs in almost all experiments except for the SP8 experiments.[3] On every SP8 experiment, the LSTM and s-RNN mean performance on Test1 is at least 10% higher than the mean performance on Test2 (except for the s-RNNs on the SP8 100k experiments). Thus with few important exceptions, the generalizations acquired by the LSTMs and s-RNNs in training extended to longer words with a similar degree of accuracy.

However, in the SP8 experiments, the notable drop in accuracy in Test2 as compared to Test1 indicates that the networks failed to generalize to longer words. In the SP8 1k and 10k experiments, this failure may be excusable because the RPNI results are similar. More precisely, the failure of

[1] This means of course that values close, but not equal to, one and zero may be shown as one and zero respectively. We confirm this happens in a few instances.

[2] Even though the SL8 100k experiments shows RPNI with an accuracy of 1.000, this is due to the rounding error. The DFA RPNI output was not identical to the minimal target DFA.

[3] The s-RNN performance is worse on Test2 than Test1 in the SL4 1k experiment.

Table 2: Accuracy on Target SL Stringsets after 100 Epochs

Training		Test	LSTM			s-RNN			RPNI
			10	30	100	10	30	100	
SL2	1k	1	0.772 (0.09)	0.717 (0.08)	0.711 (0.02)	0.766 (0.11)	0.761 (0.11)	0.762 (0.10)	**0.855**
		2	0.758 (0.09)	0.696 (0.10)	0.685 (0.02)	0.757 (0.15)	0.784 (0.17)	0.768 (0.15)	**0.844**
	10k	1	0.773 (0.17)	0.616 (0.01)	0.666 (0.01)	0.682 (0.15)	0.660 (0.11)	0.649 (0.11)	**1.000**
		2	0.772 (0.19)	0.602 (0.01)	0.650 (0.01)	0.675 (0.16)	0.650 (0.12)	0.639 (0.12)	**1.000**
	100k	1	0.684 (0.15)	0.615 (0.03)	0.644 (0.01)	0.700 (0.14)	0.723 (0.16)	0.620 (0.01)	**1.000**
		2	0.669 (0.16)	0.596 (0.02)	0.624 (0.01)	0.689 (0.16)	0.718 (0.18)	0.601 (0.01)	**1.000**
SL4	1k	1	0.902 (0.01)	0.907 (0.07)	0.884 (0.06)	0.913 (0.01)	0.956 (0.01)	**0.968 (0.01)**	0.918
		2	0.836 (0.01)	0.890 (0.04)	0.901 (0.02)	0.844 (0.01)	0.896 (0.01)	**0.911 (0.01)**	0.813
	10k	1	0.840 (0.15)	0.856 (0.12)	0.942 (0.08)	0.934 (0.12)	0.982 (0.00)	0.977 (0.01)	**0.995**
		2	0.836 (0.16)	0.852 (0.13)	0.938 (0.08)	0.938 (0.12)	**0.993 (0.00)**	0.991 (0.00)	0.978
	100k	1	0.975 (0.05)	0.917 (0.12)	0.898 (0.10)	0.905 (0.16)	0.989 (0.00)	0.986 (0.00)	**1.000**
		2	0.981 (0.04)	0.923 (0.12)	0.903 (0.10)	0.916 (0.16)	0.995 (0.00)	0.994 (0.00)	**1.000**
SL8	1k	1	0.981 (0.02)	0.976 (0.04)	0.995 (0.00)	0.989 (0.01)	**0.999 (0.00)**	**0.999 (0.00)**	0.991
		2	0.976 (0.02)	0.965 (0.03)	0.983 (0.01)	0.991 (0.00)	0.992 (0.00)	**0.996 (0.00)**	0.966
	10k	1	0.931 (0.09)	0.979 (0.02)	0.964 (0.03)	0.995 (0.01)	**0.998 (0.00)**	0.997 (0.01)	**0.998**
		2	0.980 (0.04)	0.998 (0.00)	**0.999 (0.00)**	0.998 (0.00)	0.998 (0.00)	0.997 (0.01)	0.994
	100k	1	0.909 (0.11)	0.864 (0.12)	0.849 (0.11)	0.995 (0.01)	0.997 (0.00)	0.997 (0.00)	**1.000**
		2	0.976 (0.05)	0.986 (0.02)	0.980 (0.03)	0.999 (0.00)	**1.000 (0.00)**	**1.000 (0.00)**	**1.000**

RPNI in these cases indicates that the training data itself was plausibly insufficient for accurate learning to take place. However, this failure of the networks in the SP8 100k experiment cannot be so excused because RPNI scored 100% on both test sets indicating that the training data was sufficiently rich for proper inferences to occur.

We are also specifically interested in the LSTM performance in the SL versus SP experiments. With the exception of SL2 and SP2 experiments, there is generally a drop in accuracy for LSTMs on the SP experiments. This is most evident comparing the SP8 experiments with the SL8 experiments. In particular, LSTM mean accuracy on the SL4 and SL8 test sets were generally better than their mean accuracy on the SP4 and SP8 test sets. (The one exception is the comparison of the SL8 1k to the SP8 1k experiments.) This poorer performance in the SP experiments challenges the narrative in the deep learning literature that LSTMs solve the problem of learning long-term dependencies.

We are also interested in comparing the s-RNN performance between the SL and SP experiments. Like the LSTMS, the s-RNNs generally performed worse in the SP 4 and 8 experiments than the SL 4 and 8 experiments.

Next we compare the performance of the LSTMs to the performance of the s-RNNs in the SP experiments. Given the narrative that the LSTM architecture addresses a known problem for s-RNNs, we expected that the s-RNN performance would be worse than the LSTM performance on the SP experiments. However this was not the case. In fact, the s-RNN results are about the same, and in some cases superior, perhaps most notably in the SP8 100k experiments. Consequently, this result also challenges the deep learning narrative regarding long-term dependencies.

We conclude this section by mentioning an additional striking and unexpected result. The mean accuracy of the LSTMS and s-RNNs on the SL2 experiments, which are arguably the simplest patterns to learn, were among the worst results in all the experiments. While the accuracy in the SL2 1k experiments could be due to insufficient training data (as indicated by the RPNI result), this rationale is not available in the 10k and 100k experiments. Their performance here suggests that even the ability of recurrent networks to learn local dependencies in formal languages needs further study.

6 Discussion

The results above indicate the the neural networks had the most difficulty in the SL2 and SP8 experiments. Generally when learning fails, the possible culprits are either a deficiency in the data or

Table 3: Accuracy on Target SP Stringsets after 100 Epochs

Training		Test	LSTM 10	LSTM 30	LSTM 100	s-RNN 10	s-RNN 30	s-RNN 100	RPNI
SP2	1k	1	0.847 (0.06)	0.935 (0.07)	0.952 (0.07)	0.910 (0.05)	0.999 (0.00)	0.999 (0.00)	**1.000**
		2	0.873 (0.10)	0.951 (0.08)	0.947 (0.08)	0.976 (0.01)	**1.000 (0.00)**	**1.000 (0.00)**	**1.000**
	10k	1	0.734 (0.12)	0.673 (0.04)	0.720 (0.03)	0.937 (0.13)	0.960 (0.08)	0.972 (0.06)	**1.000**
		2	0.723 (0.12)	0.656 (0.04)	0.701 (0.03)	0.934 (0.13)	0.960 (0.08)	0.972 (0.06)	**1.000**
	100k	1	0.680 (0.08)	0.707 (0.10)	0.732 (0.07)	0.974 (0.07)	0.974 (0.07)	0.982 (0.04)	**1.000**
		2	0.665 (0.09)	0.697 (0.12)	0.716 (0.08)	0.977 (0.07)	0.974 (0.08)	0.986 (0.04)	**1.000**
SP4	1k	1	0.883 (0.06)	0.885 (0.08)	0.775 (0.05)	0.890 (0.05)	0.969 (0.02)	0.988 (0.01)	**1.000**
		2	0.943 (0.04)	0.840 (0.09)	0.749 (0.06)	0.885 (0.06)	0.975 (0.01)	0.985 (0.01)	**1.000**
	10k	1	0.862 (0.13)	0.880 (0.14)	0.853 (0.08)	0.696 (0.05)	0.840 (0.15)	0.903 (0.12)	**1.000**
		2	0.862 (0.14)	0.877 (0.15)	0.843 (0.08)	0.686 (0.05)	0.841 (0.16)	0.900 (0.12)	**1.000**
	100k	1	0.842 (0.13)	0.791 (0.14)	0.720 (0.09)	0.884 (0.14)	0.828 (0.17)	0.895 (0.12)	**1.000**
		2	0.831 (0.13)	0.785 (0.13)	0.716 (0.08)	0.900 (0.15)	0.827 (0.17)	0.902 (0.13)	**1.000**
SP8	1k	1	0.844 (0.04)	0.863 (0.05)	**0.901 (0.01)**	0.871 (0.01)	0.885 (0.02)	0.878 (0.01)	0.817
		2	0.699 (0.08)	0.627 (0.05)	0.692 (0.03)	**0.719 (0.02)**	0.663 (0.06)	0.668 (0.03)	0.587
	10k	1	0.827 (0.15)	0.798 (0.11)	0.804 (0.04)	0.818 (0.12)	0.856 (0.10)	**0.979 (0.02)**	0.873
		2	0.654 (0.11)	0.672 (0.10)	0.638 (0.05)	0.566 (0.05)	0.646 (0.05)	**0.811 (0.08)**	0.634
	100k	1	0.880 (0.10)	0.927 (0.08)	0.904 (0.08)	0.893 (0.14)	0.978 (0.04)	0.988 (0.01)	**1.000**
		2	0.760 (0.12)	0.802 (0.13)	0.739 (0.09)	0.825 (0.15)	0.909 (0.11)	0.907 (0.09)	**1.000**

some deficiency in the learning mechanism. The RPNI results exclude the former rationale for the SL2 10k, SL2 100k, and SP8 100k experiments.

As for a deficiency in the neural networks, there are many possible modifications that would arguably improve outcomes. For instance, if the results are due the networks overfitting then the networks could be augmented with the dropout method or could be told to stop training before 100 epochs provided some condition is reached ("early stopping").

We explored the possibility of early stopping and the results are shown in Tables 4 and 5. These tables repeat the RPNI results for easier comparison. Training for the networks were halted at the epoch which maximized the accuracy of the networks on the validation set. The validation set was a held-out fraction (10%) of the training set that was randomly selected from the training set. Because the training set contained duplicates, it is not the case that every test item in the validation set is novel, as is the case with Test1 and Test2 items.

These experiments show that the SL2 results do improve, but not to the level of the other experimental conditions. There is also improvement in the SP8 experiments though the difference between Test1 and Test2 results remains large. We also experimented with the dropout method (not shown). Like early stopping, this improved the results, but not the observed trends described above.

Of course there are other ways to improve the "learning power" of recurrent neural networks. The vector sizes can be increased, multiple layers can be included in the recurrent components, Kalman filters could be used, and so on. So this is one avenue of future research.

However, these methods go hand-in-hand with varying the complexity of the target languages at different levels of abstraction, such as adding more forbidden strings to the grammars, increasing the size of the alphabet, increasing k, or moving up the subregular hierarchy to more complex classes. The goal here is not one-upmanship, but to instead get a better understanding of how properties of RNNs relate to properties of formal language classes, like the well-understood subregular ones presented in section 3.

As such, these experiments showed something very clearly. They showed naive LSTMs have difficulty with learning a formal language defined by a forbidden subsequence of length 8 but little to no difficulty with learning a formal language defined by forbidden substrings of length 8. The difference between substring and subsequence—which reduces logically to the question of whether order is represented in strings with the successor or precedence relation (Rogers et al., 2013)—is thus significant for naive LSTMs.

Table 4: Accuracy on Target SL Stringsets Early Stopping

Training		Test	LSTM 10	LSTM 30	LSTM 100	s-RNN 10	s-RNN 30	s-RNN 100	RPNI
SL2	1k	1	0.818 (0.03)	0.843 (0.05)	0.923 (0.02)	0.848 (0.06)	0.904 (0.03)	**0.930 (0.03)**	0.855
		2	0.780 (0.06)	0.820 (0.07)	0.905 (0.04)	0.871 (0.10)	0.980 (0.02)	**0.992 (0.01)**	0.844
	10k	1	0.925 (0.07)	0.851 (0.04)	0.875 (0.04)	0.936 (0.05)	0.884 (0.07)	0.729 (0.12)	**1.000**
		2	0.919 (0.09)	0.836 (0.06)	0.835 (0.10)	0.964 (0.07)	0.868 (0.11)	0.753 (0.15)	**1.000**
	100k	1	0.737 (0.14)	0.711 (0.14)	0.730 (0.03)	0.869 (0.15)	0.767 (0.17)	0.625 (0.01)	**1.000**
		2	0.727 (0.15)	0.698 (0.15)	0.711 (0.04)	0.885 (0.17)	0.766 (0.19)	0.605 (0.01)	**1.000**
SL4	1k	1	0.898 (0.01)	0.939 (0.01)	0.945 (0.01)	0.908 (0.01)	0.945 (0.01)	**0.958 (0.01)**	0.918
		2	0.829 (0.01)	0.888 (0.01)	0.887 (0.00)	0.840 (0.01)	0.883 (0.01)	**0.898 (0.01)**	0.813
	10k	1	0.953 (0.05)	0.956 (0.04)	**0.997 (0.00)**	0.976 (0.03)	0.982 (0.00)	0.981 (0.00)	0.995
		2	0.934 (0.06)	0.932 (0.05)	**0.995 (0.01)**	0.973 (0.04)	0.989 (0.00)	0.990 (0.00)	0.978
	100k	1	0.994 (0.00)	0.973 (0.07)	**1.000 (0.00)**	0.990 (0.00)	0.990 (0.00)	0.987 (0.00)	**1.000**
		2	0.996 (0.00)	0.975 (0.07)	**1.000 (0.00)**	0.996 (0.00)	0.996 (0.00)	0.995 (0.00)	**1.000**
SL8	1k	1	0.966 (0.02)	0.983 (0.01)	**0.995 (0.00)**	0.962 (0.02)	0.969 (0.01)	0.971 (0.01)	0.991
		2	0.971 (0.01)	0.980 (0.02)	**0.994 (0.00)**	0.969 (0.02)	0.974 (0.01)	0.977 (0.01)	0.966
	10k	1	0.990 (0.01)	0.996 (0.00)	**0.999 (0.00)**	0.998 (0.00)	**0.999 (0.00)**	**0.999 (0.00)**	0.998
		2	0.994 (0.00)	0.995 (0.00)	**0.998 (0.00)**	0.997 (0.00)	**0.998 (0.00)**	**0.998 (0.00)**	0.994
	100k	1	0.993 (0.01)	0.998 (0.00)	**1.000 (0.00)**	0.999 (0.00)	**1.000 (0.00)**	**1.000 (0.00)**	**1.000**
		2	0.994 (0.01)	0.999 (0.00)	**1.000 (0.00)**	0.999 (0.00)	**1.000 (0.00)**	**1.000 (0.00)**	**1.000**

The successor/precedence difference is also challenging for s-RNNs, but they are not any more challenging for s-RNNs than they are for LSTMs. In fact, s-RNNs outperform LSTMs on many of the SP experiments.

From these results, it is hard to see how LSTMs are a solution to learning long-term dependencies, which are problematic for s-RNNs.

Rodriguez (2001) cautions against underestimating s-RNNs. In this article he argues he shows "a range of language tasks in which an [s-RNN] develops solutions that not only count but also copy and store counting information. In one case, the network stores information like an explicit storage mechanism. In other cases, the network stores information more indirectly..." In short, despite the well-known exploding and vanishing gradient problems, s-RNNs can learn to store information long-term in some circumstances.

So what then explains the unexpected performance of s-RNNs? One possibility is the Adam optimization method (Kingma and Ba, 2014) which the s-RNNs in these experiments used. This optimization technique is relatively recent and replaces the method of stochastic gradient descent (SGD), which was used in the decades prior to Adam's introduction. The narrative regarding long-term dependencies surrounding LSTMs and s-RNNs was developed in the context of SGD. So one hypothesis is that the narrative is conditioned on the use of SGD as the optimization method, but that once Adam is used in its place, the capacity of s-RNNs to learn long-term dependencies is much improved. In a sense, maybe Adam itself goes some distance in resolving the vanishing and exploding gradient problems.

We have begun to test this hypothesis by running the experiments with SGD instead of Adam. While we are not yet able to provide a full report, the result of a single experiment with s-RNNs using SGD as the optimization method shows a serious decline its performance as shown in Table 6 Obviously, this is an area of current and future research.

Another possibility is highlighted by the fact that there are different *kinds* of long-term dependencies. It may be that LSTMs outperform s-RNNs on long-term dependencies unlike the Strictly Piecewise ones tested here. As mentioned, the subregular hierarchies (Figure 1) are particularly good at distinguishing different types of long-term dependencies. One contribution of this paper is the more fine-grained classification of long-term dependencies that formal language theory and the subregular language classes provide researchers.

Finally, we return to the discussion of interpreting the performance of the neural networks

Table 5: Accuracy on Target SP Stringsets Early Stopping

Training		Test	LSTM			s-RNN			RPNI
			10	30	100	10	30	100	
SP2	1k	1	0.871 (0.04)	0.954 (0.05)	0.992 (0.00)	0.910 (0.05)	0.994 (0.01)	0.992 (0.01)	**1.000**
		2	0.960 (0.03)	0.989 (0.02)	0.998 (0.00)	0.976 (0.01)	0.998 (0.01)	**1.000 (0.00)**	**1.000**
	10k	1	0.890 (0.07)	0.941 (0.04)	0.977 (0.02)	0.995 (0.01)	0.981 (0.05)	0.999 (0.00)	**1.000**
		2	0.979 (0.02)	0.990 (0.01)	0.994 (0.01)	**1.000 (0.00)**	0.984 (0.05)	**1.000 (0.00)**	**1.000**
	100k	1	0.833 (0.14)	0.819 (0.12)	0.890 (0.08)	0.997 (0.01)	0.999 (0.00)	0.997 (0.00)	**1.000**
		2	0.838 (0.16)	0.805 (0.13)	0.872 (0.09)	**1.000 (0.00)**	**1.000 (0.00)**	**1.000 (0.00)**	**1.000**
SP4	1k	1	0.881 (0.06)	0.946 (0.04)	0.963 (0.03)	0.887 (0.05)	0.966 (0.02)	0.979 (0.01)	**1.000**
		2	0.950 (0.03)	0.960 (0.03)	0.983 (0.01)	0.883 (0.05)	0.975 (0.01)	0.979 (0.01)	**1.000**
	10k	1	0.899 (0.11)	0.958 (0.07)	0.991 (0.01)	0.935 (0.08)	0.968 (0.04)	0.999 (0.00)	**1.000**
		2	0.926 (0.09)	0.971 (0.05)	0.991 (0.01)	0.954 (0.07)	0.984 (0.02)	**1.000 (0.00)**	**1.000**
	100k	1	0.943 (0.08)	0.940 (0.08)	0.920 (0.06)	0.942 (0.09)	0.958 (0.09)	0.973 (0.07)	**1.000**
		2	0.928 (0.08)	0.930 (0.09)	0.911 (0.07)	0.951 (0.09)	0.962 (0.08)	0.974 (0.08)	**1.000**
SP8	1k	1	0.884 (0.02)	0.884 (0.02)	**0.903 (0.02)**	0.861 (0.01)	0.878 (0.02)	0.857 (0.02)	0.817
		2	**0.733 (0.03)**	0.643 (0.06)	0.688 (0.04)	0.730 (0.01)	0.681 (0.06)	0.625 (0.04)	0.587
	10k	1	0.934 (0.05)	0.921 (0.05)	0.959 (0.03)	0.908 (0.02)	0.952 (0.03)	**0.991 (0.00)**	0.873
		2	0.637 (0.08)	0.659 (0.10)	0.704 (0.11)	0.600 (0.08)	0.640 (0.10)	**0.837 (0.05)**	0.634
	100k	1	0.977 (0.04)	0.975 (0.04)	0.980 (0.02)	0.964 (0.05)	0.990 (0.03)	**1.000 (0.00)**	**1.000**
		2	0.881 (0.11)	0.865 (0.13)	0.864 (0.08)	0.890 (0.08)	0.942 (0.09)	0.984 (0.03)	**1.000**

in light of the grammatical inference algorithm RPNI. We argued it was useful to use RPNI to evaluate the quality of the data. However, this argument has one potential flaw. RPNI can be said to measure the sufficiency of the data, at least for learning mechanisms which are targeting *regular* stringsets. If the learning mechanisms are capable of learning nonregular languages then they may need *more* data to learn some regular languages (in order to distinguish them from some context-free language for example).

RPNI in a sense "knows" that it is learning regular stringsets because it only ever builds finite-state acceptors. There is evidence that RNNs, on the other hand, can learn some non-regular stringsets (Rodriguez, 2001; Schmidhuber et al., 2002). Furthermore, it is not known how to build this kind of a priori knowledge into neural networks. In other words, the use of RPNI to understand the quality of the training data is suggestive but not probative. Nonetheless, in the absence of theoretical learning results of RNNs on nonregular formal languages, using grammatical inference algorithms which provably learn large classes of languages (like RPNI) may be the best anyone can do.

7 Conclusion

In this paper, we developed controlled experiments using formal languages for investigating the ability of RNNs to learn long-term dependencies. The results are difficult to understand in light of the dominant narrative in the deep learning literature regarding the efficacy of LSTMs over s-RNNs in this learning task.

More generally, the experiments presented here help show how formal language theory can reveal the advantages and disadvantages of various RNN models more clearly than testing with real-world datasets. The primary reason is that we control the nature of the target patterns and their complexity. This was illustrated here with the comparison of SL and SP languages which encode local and long-term dependencies, respectively. From a logical perspective, the only difference between the SL and SP classes is the way in which order is represented in strings: SL classes use the successor relation and SP classes use the precedence relation.

Controlled experiments can also include carefully designed test sets. Here, through Test1 and Test2, we could control the test data to better understand how the RNNs generalize to words longer than the ones found in training.

We argued these controlled experiments show there is still some distance to go to understand

Table 6: s-RNN with SGD Accuracy on Target Stringsets after 100 epochs

		sRNN with SGD		
	Training	v10	v30	v100
	Regimen	Test1 Test2	Test1 Test2	Test1 Test2
SL2	1k	0.7641 0.7228	0.8015 0.7336	0.7362 0.6898
	10k	0.7194 0.6983	0.7752 0.7169	0.9745 0.9981
	100k	0.8034 0.7417	0.9713 0.9984	0.9482 0.9968
SL4	1k	0.9304 0.8683	0.9072 0.8359	0.8789 0.8134
	10k	0.8234 0.7870	0.8347 0.7977	0.8588 0.8187
	100k	0.4738 0.5005	0.8957 0.8492	0.5567 0.5641
SL8	1k	0.8191 0.8265	0.8402 0.8564	0.8065 0.8064
	10k	0.5076 0.4951	0.5076 0.4951	0.6441 0.6409
	100k	0.7830 0.7789	0.7830 0.7789	0.8079 0.8031
SP2	1k	0.7346 0.7331	0.8739 0.9174	0.8370 0.9277
	10k	0.8371 0.8449	0.8400 0.8264	0.9998 1.0000
	100k	0.6071 0.5902	0.5696 0.5645	1.0000 1.0000
SP4	1k	0.7625 0.7361	0.5744 0.5306	0.6668 0.5991
	10k	0.7270 0.6808	0.6167 0.6331	0.7270 0.6398
	100k	0.7038 0.6287	0.5626 0.5582	0.6643 0.5845
SP8	1k	0.8160 0.7702	0.1984 0.2882	0.8393 0.7372
	10k	0.6520 0.6201	0.8364 0.6677	0.4504 0.5204
	100k	0.8251 0.7505	0.8117 0.7645	0.4409 0.5027

how RNNs, including LSTMs, represent and learn long-term dependencies in sequential patterns. We tentatively hypothesized that the optimization technique Adam may alleviate some difficulties that s-RNNs may have with long-term dependencies. If correct, this means the source of the problem was not the s-RNNs per se, but SGD.

Finally, we believe that more controlled experiments of the sort presented here with more complex formal languages and less naive LSTMs and other kinds of RNNs, in conjunction with learning algorithms from grammatical inference, are critical to better understanding the capacity of neural networks to represent and learn long-term dependencies.

Acknowledgements

The authors acknowledge the support of the Japan Society for the Promotion of Science (JSPS) KAKENHI grant 26730123 to CS and the support of the National Institute of Health (NIH) grant R01HD087133-01 to JH.

We are grateful to the feedback of the audiences of the LAML conference, the LearnAut workshop and the IACS seminar. We also thank Jim Rogers for assistance in determining the complexity of the embedded Reber grammar.

References

Hasan Ibne Akram, Colin de la Higuera, Huang Xiao, and Claudia Eckert. 2010. Grammatical inference algorithms in matlab. In José M. Sempere and Pedro García, editors, *Proceedings of the 10th International Colloquium of Grammatical Inference: Theoretical Results and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 262–266.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

J. Richard Büchi. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly* 6(1-6):66–92.

Mike Casey. 1996. The dynamics of discrete-time computation with application to recurrent neural networks and finite state machine extraction. *Neural computation* 8(6):1135–1178.

Stephan K. Chalup and Alan D. Blair. 2003. Incremental training of first order recurrent neural networks to predict a context-sensitive language. *Neural Networks* 16(7):955–972.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory* page 113124. IT-2.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton & Co., Printers, The Hague.

Noam Chomsky. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.

Colin de la Higuera. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning* 27(2):125–138.

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14(2):179–211.

Herbert B. Enderton. 2001. *A Mathematical Introduction to Logic*. Academic Press, 2nd edition.

Rémi Eyraud, Jeffrey Heinz, and Ryo Yoshinaka. 2016. Efficiency in the identification in the limit learning paradigm. In Jeffrey Heinz and José Sempere, editors, *Topics in Grammatical Inference*, Springer-Verlag Berlin Heidelberg, pages 25–46.

Pedro García and José Ruiz. 2004. Learning k-testable and k-piecewise testable languages from positive data. *Grammars* 7:125–140.

Pedro Garcia, Enrique Vidal, and José Oncina. 1990. Learning locally testable languages in the strict sense. In *Proceedings of the Workshop on Algorithmic Learning Theory*. pages 325–338.

C L Giles, C B Miller, D Chen, H H Chen, G Z Sun, and Y C Lee. 1992. Learning and Extracting Finite State Automata with 2nd-Order Recurrent Neural Networks. *Neural Computation* 4(3):393–405.

Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Morgan and Claypool Publishers.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *CoRR* abs/1503.04069.

Jeffrey Heinz. 2010a. Learning long-distance phonotactics. *Linguistic Inquiry* 41(4):623–661.

Jeffrey Heinz. 2010b. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 897–906.

Jeffrey Heinz, Colin de la Higuera, and Menno van Zaanen. 2015. *Grammatical Inference for Computational Linguistics*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. 2012. Learning with lattice-structured hypothesis spaces. *Theoretical Computer Science* 457:111–127.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 29–32.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, Upper Saddle River, NJ, 2nd edition.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press.

Jose Oncina and Pedro Garcia. 1992. Identifying regular languages in polynomial time. In *Advances In Structural And Syntactic Pattern Recognition, Volume 5 Of Series In Machine Perception And Artificial Intelligence*. World Scientific, pages 99–108.

Jordan B. Pollack. 1991. The Induction of Dynamical Recognizers. *Machine Learning* 7(2):227–252.

Juan Antonio Prez-Ortiz, Felix A. Gers, Douglas Eck, and Jrgen Schmidhuber. 2003. Kalman filters improve {LSTM} network performance in problems unsolvable by traditional recurrent nets. *Neural Networks* 16(2):241 – 250.

Arthur S. Reber. 1967. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior* 6(6):855–863.

Paul Rodriguez. 2001. Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural Computation* 13(9):2093–2118.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*. Springer, volume 6149 of *Lecture Notes in Artifical Intelligence*, pages 255–265.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In Glyn Morrill and Mark-Jan Nederhof, editors, *Formal Grammar*. Springer, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108.

James Rogers and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20:329–342.

J. Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85–117.

J. Schmidhuber, F. Gers, and D. Eck. 2002. Learning nonregular languages: A comparison of simple recurrent networks and lstm. *Neural Computation* 14:2039–2041.

Imre Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975*. Springer, pages 214–222.

Zipser D. Smith, A.W. 1989. Encoding sequential structure: experience with the real-time recurrent learning algorithm. *Neural Networks* pages 0–4.

Wolfgang Thomas. 1982. Classifying regular events in symbolic logic. *Journal of Computer and Systems Sciences* 25:370–376.

Masaru Tomita. 1982. Learning of construction of finite automata from examples using hill-climbing. *Proc. Fourth Int. Cog. Sci. Conf.* pages 105 – 108.

Raymond L Watrous and G M Kuhn. 1992. Induction of Finite-State Automata Using Second-Order Recurrent Networks. *Advances in Neural Information Processing Systems* (10):309–316.

Can neural networks learn logical reasoning?

Sara Veldhoen
Cognition, Language & Computation lab
Institute for Logic, Language and
Computation, University of Amsterdam

Willem Zuidema
Cognition, Language & Computation lab
Institute for Logic, Language and
Computation, University of Amsterdam
zuidema@uva.nl

Abstract

We replicate a result from Bowman, Potts & Manning (2015), showing success on a task where neural networks learn logical reasoning. We study the solutions the neural networks find using visualisation & dimensionality reduction techniques. This allows us to start addressing a key methodological worry: that without an understanding of how the networks solve the task, we cannot determine whether the networks have learned a general solution or a large number of 'local' approximations. We moreover look in some detail into the way Bowman et al. constructed their train and test set, and find that the random train–test set split does not allow us to distinguish between the local and global solutions. Finally, we identify an alternative test, that, if past, would demonstrate that the networks have learned a global solution. We find, however, that recursive neural networks (the simplest of the models studied) fail this more stringent test. We conclude that the question whether neural networks can learn to reason logically is still open, and deserves a nuanced, graded answer rather than a binary one.

1 Introduction

Can neural networks learn to reason logically? To study this question, Bowman et al. (2015) trained networks on a task involving pairs of simple sentences like 'all reptiles walk' and 'some turtles move'. In their setup, two recursive neural networks (Tree-RNNs) are used to compute representations for each of the sentences, which in turn enter in a simple feed-forward network (with a softmax output layer) trained to predict the logical relation between the sentences[1]. Bowman et al. find that their networks can indeed learn to correctly classify the pairs of sentences from their artificial data set.

Does this show that neural networks and their associated learning rules are expressive enough to learn to perform logical reasoning? This would be an important conclusion, in light of the debate about the necessity to assume 'symbolic' machinery to account for such cognitive tasks. However, not only are both the logic and the neural networks chosen not very representative for their domains, but there are also a number of methodological questions that need to be answered before conclusions can be drawn about the actual success of Tree-RNNs on the specific task.

In this paper we first replicate the results from Bowman et al., and then study the solutions the neural networks find in some detail using visualisation & dimensionality reduction techniques. This allows us to start addressing a key methodological worry: that without an understanding of how the networks solve the task, we cannot determine whether the networks have learned a general solution or a large number of 'local' approximations. We moreover look in some detail into the way Bowman et al. constructed their train and test set, and find that the random train–test set split does not allow us to distinguish between the local and global solutions. Finally, we identify an alternative test, based on identifying logical equivalence, that if past would demonstrate that the networks have learned a global solution. As it turns out, the Tree-RNNs we study in this paper fail this more stringent test. We conclude that the question whether neural networks can learn to reason logically is still open, and, moreover, that this question deserves a nuanced, graded answer rather than a binary one.

2 Replication

The task is to predict logical relations that hold between pairs of sentences that involve monotonicity reasoning and quantification. Rather than classifying relations as entailment, contradiction or neither, as is commonly done, a more informative inventory of

[1] For the given pair of sentences, the correct logical relation is 'forward entailment', because turtles are a subset of the reptiles, walking is a type of moving, and 'all X Y' entails 'some V W' if V entails W and Y entails W.

seven semantic relations by (MacCartney and Manning, 2009) is used. These relations correspond to five commonly used set-theoretic relations between non-empty sets (is subset of; equals; is superset of; intersects with; does not intersect with – depicted on the top and middle row of Figure 1), and two additional ones (like 'intersects with' and 'does not intersect with', but with the extra condition that the union of the two sets is the universe – depicted on the bottom row of Figure 1).

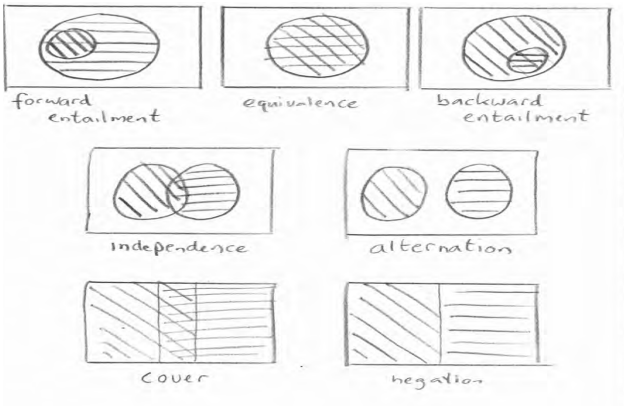


Figure 1: The seven logical relations by (MacCartney and Manning, 2009)

All sentences are of the form `quantifier (not) noun (not) verb`. Only five nouns and four verbs occur, together with ten quantifiers, resulting in 800 possible sentences. Of each sentence pair, the logical relation is determined using natural logic calculus, which we describe in Section 4. In Table 1, a few training examples are presented[2].

A Recursive Neural Network (henceforth: *Tree-RNN*) is a neural network that can be shaped after the syntactic structure of a sentence (Goller and Kuchler, 1996; Socher et al., 2013; Le and Zuidema, 2014). We follow the architecture and training procedure proposed by (Bowman et al., 2015), who applied this Tree-RNN to the logical inference task described above.

The training set-up is displayed in Figure 2. A Tree-RNN is instantiated for both sentences. The weights and word embeddings are shared between them. The networks yield sentence representations, which are concatenated and fed into a so-called *comparison layer*. This is a higher-dimensional layer that can be thought of as a representation of the relation between the two sentences. From this representation, a softmax classifier predicts the logical relation.

After training, by backpropagating the error from the softmax layer all the way back to the word embeddings, classification accuracy on the training data is 100%. On held-out data, performance varies from 98.9 to 99.5%, over different data splits. These results are on par with those by (Bowman et al., 2015), who reported 99.6 and 99.2% accuracy on train and test set respectively. This is promising, but what have the networks really learned? As noted by (Bowman et al., 2015), "interesting analytical questions remain about *how* these models encode the underlying logics."

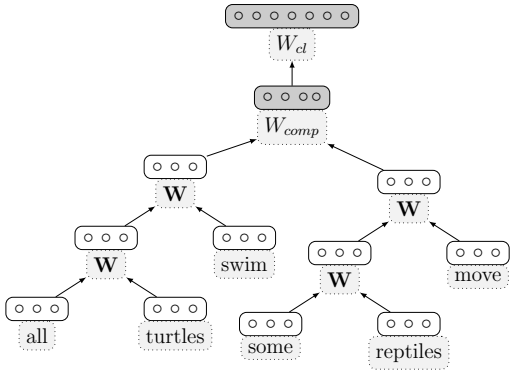


Figure 2: Training set-up

3 What have the networks really learned?

There are two sets of learned parameters in the Tree-RNN: the word embeddings, and the composition function. Moreover, there are weights to connect the sentence representations to the comparison layer, and weights that transform the contents of the comparison layer to a distribution over the possible classes. For our analysis we focus on the sentence representations that emerge, reflecting the combined effects of the word embeddings and the composition function.

3.1 Dimensionality Reduction

The representations of words and phrases/ sentences in the Tree-RNNs are 25-dimensional, making them hard to inspect manually. We applied two dimensionality reduction techniques to the data, one linear and the other non-linear:

PCA (principal components analysis) determines linear components that best explain the distribution of the data. By projecting high-dimensional data along their first principal components, we can visually inspect the data. An advantage of this technique is that an explicit mapping exists from the original space to the principal components. That is, the analysis can be applied to a certain set of data-points, for instance all 800 possible sentence representations. The result can be used to project any data-point in the same space, such as a word embedding, onto the same principal

[2]We used the original data, courtesy of Bowman et al., available at `https://github.com/sleepinyourhat/vector-entailment/releases/tag/W15-R2`.

((lt_two warthogs) move)	⌣	((two (not turtles)) move)
((no mammals) swim)	|	((all mammals) move)
((no turtles) walk)	⊐	((some (not warthogs)) (not move))

Table 1: Example pairs of sentences from the artificial dataset of (Bowman et al., 2015)

components.

t-SNE (t-Distributed Stochastic Neighbor Embedding; (van der Maaten and Hinton, 2008)) is not restricted to linear projections. It fits a lower-dimensional (e.g., 2D) manifold to the data-points. This yields a visualization that may better explain the relations between the data-points, but there is no mapping that can be applied to new data-points after the manifold has been fitted.

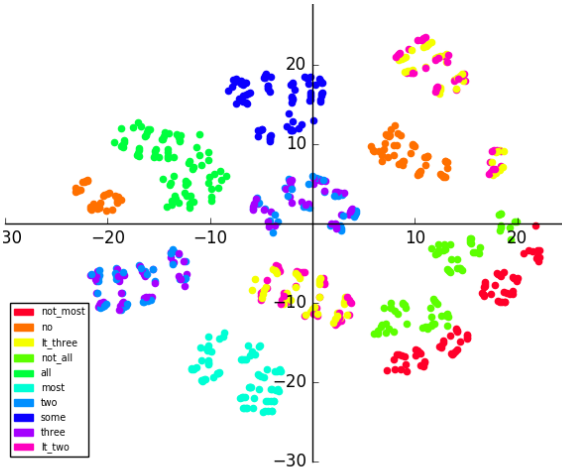
Whichever dimensionality reduction is used, it is important to remember that we are not looking at the real data directly. The dimensionality reduction is always applied using a certain set of data, e.g. the word embeddings or a set of sentence embeddings. In what follows, we will specify this set for each diagram.

3.2 Sentence Embeddings

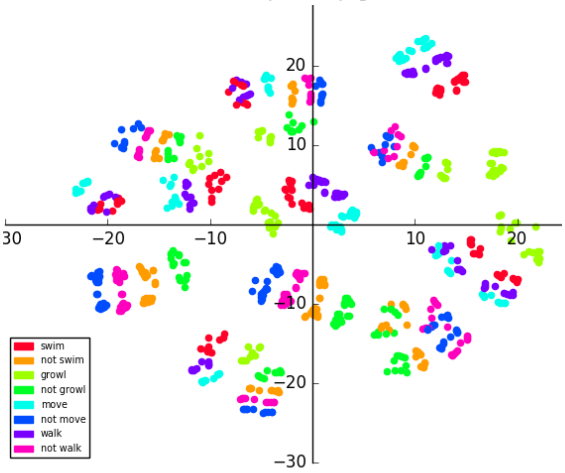
We applied both dimensionality reductions to the representations of all (800) possible sentences. The t-SNE reduction is shown in Figure 3; the PCA reduction (here using 1st and 4th principal component) in Figure 4. These plots demonstrate three interesting features of the way the space of sentence embeddings gets organized when optimizing for logical reasoning:

(1) From 3a, it is clear that the quantifiers are the most prominent feature: they determine the primary clustering of the data in this 2D reduction. **(2)** The clusters for `two` and `three`; and for `lt_two` and `lt_three`, largely overlap. This is understandable: the semantic relations and projectivity behavior of those respective quantifier pairs is almost identical. Notably, the clusters corresponding to those pairs as well as the points belonging to the quantifier `no` actually end up in two separate clusters, whereas for the other quantifiers there are single convex clusters. This qualitative difference is observed in some (but not all) other runs too. Interestingly, this division corresponds to whether the verb of the sentence is negated, as we can see in Figure 3b. Here, the same reduced data-points are displayed with the coloring corresponding to verb identity, using different colors to distinguish a predicate and its negation. (Plotting the same data-points using the identity of the noun for coloring did not yield any useful information. There is some organization, but no systematic pattern over clusters or over different runs.)

(3) In some cases, such as for the cluster of `some`, negation of the verb correspond to a fixed linear



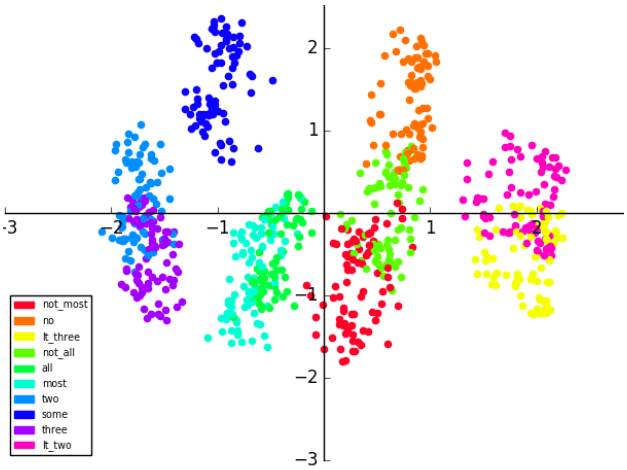
(a) Colors distinguishing quantifiers



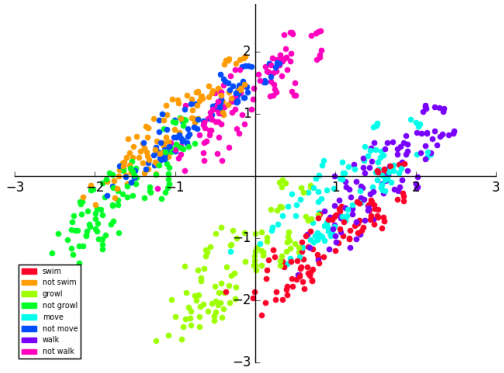
(b) Colors distinguishing verbs

Figure 3: t-SNE applied to all 800 sentence representations

shift (the observed order is swim and walk - move and growl - not swim and not growl - not walk - not move). In other cases, such as for the cluster of 'not_most', the negation has a mirroring effect. (the order is: not_growl - not_swim - not_move and not_walk - move and walk - swim - growl). A similar mirroring effect of negation of the verb exists in the PCA projection (not shown here). Negation is in these case thus represented as a *geometric operation*, a local mirroring effect (reflectional symmetry), that allows the networks to generalize it to unseen data. The network has discovered something general about the negation of predicates.



(a) Fourth against first component, which exhibit the most prominent clustering of quantifiers.



(b) Second and third component, which show the most prominent clustering of verbs.

Figure 4: PCA applied to all 800 sentence representations

3.3 Logical Relations as linear shifts in the Representation Space

For every two sentences in the dataset, the logical relation between them is given. When we study in the PCA representation pairs of sentences for which each of the 7 logical relations hold, we find that these logical relations are instantiated with a variety of different geometric relations in the 2-dimensional projection. Although this observation is difficult to interpret (given the high-dimensionality of the sentence vector space, and the non-linearity of its projections into the comparison layer), it is in any case not providing us with evidence in favor of the hypothesis that the network has learned a unified notion of entailment, equivalence or contradiction. Figure 5 illustrates this point. Here a new PCA was applied to the *difference vectors* of 6818 sentence pairs (the development set). In the figure, they are projected along their first two components. We can certainly distinguish some tendency to cluster in this image, but we do not find a single cluster per relation. This is similarly the case in each of the other combinations of the first five principle components.

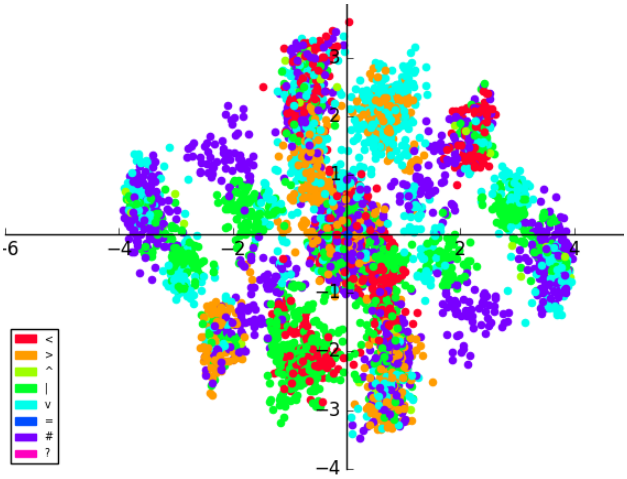


Figure 5: Second against first components of PCA applied to the difference vectors between representations of 6818 sentence pairs. The symmetry in the origin emerges from the fact that for almost every $\langle A, B\rangle$ also $\langle B, A\rangle$ is in the dataset.

Our hypothesis is that each of the clusters corresponds to a certain reasoning *pattern*. Rather than learning a single transformation that belongs to a logical relation, the network seems to have learned a set of such transformations. For instance, one transformation for sentence pairs `all x V` and `some y V`.

These visualisations thus suggests that the networks compute sentence representations such that every sentence gets assigned a different vector, and that some local generalizability but no global generalizability arises in the geometric relations between sentences. This leads to a prediction that the networks only reach their high accuracy, because for every sentence pair in the test set there are one or multiple similar sentence pairs in the train set with a known logical relation between them. For instance, the network can predict the logical relation between sen-

tences in the pair <'All turtles walk', 'Some reptiles move'>, only because it has seen the label for the pair <'All warthogs walk', 'Some mammals move'>, or for < 'Some reptiles move','All turtles walk',>, or for <'Some mammals move', 'All warthogs walk'> etc. Given this possibility we can see that a random split between train and test set, where most test set pairs will have such a 'near neighbour', is not the right approach to distinguish between *local* generalization and more *global* generalization.

Can we falsify this prediction by evaluating whether the network can also generalize to logical relations for which no similar sentence pairs have been observed? It turns out that the dataset of (Bowman et al., 2015) does allow for this possibility, but it requires us to first have a deeper look into the logic they used.

4 A deeper look into the used logic

The task that the network was applied to, is created in the tradition of an approach to reasoning called Natural Logic (Lakoff, 1970; Benthem, 1986). This tradition views reasoning as something that applies directly to the surface form of language, rather than to a so-called 'language of thought'. In practice, this means that inference relations are computed between sentences, without mapping them to a different semantic representation, using a calculus. In this section, we describe the calculus that was used to create the dataset. Furthermore, we point out some limitations of this system that may have affected the dataset.

4.1 Calculus

A calculus is described by (MacCartney and Manning, 2009) to compute the relation that holds between two utterances p (the premise) and h (the hypothesis). The following steps are taken to compute the inference relation: **(1)** Determine a sequence of atomic edits $\langle e_1, \ldots, e_n \rangle$ that transforms $x_0 = p$ into $x_n = h$ (substitution, deletions, and insertions) where $x_i = e_i(x_{i-1})$. **(2)** For each atomic edit e_i: (a) determine the lexical relation $\beta(e_i)$; (b) project it through the tree to determine the relation with previous step $\beta(x_{i-1}, e_i)$ (In this case, only quantifiers and 'not' can influence projection); (c) Join the atomic semantic relations with the previous results to obtain $\beta(x_0, x_i)$.

The information that is required by the calculus to fulfill the task can be specified in a number of tables. For step 2(a), the relations between lexical items that can substitute each other need to be determined. That is, a table saying that hippos are mammals, and that barking entails making sound. In step (b), the logical relation is projected through the tree. For the dataset at hand, projection has an effect if a relation is under the scope of a negation, or under a quantifier.

4.2 Limitations

The Natural Logic is not claimed to be a proper logical system – but for Bowman et al. (2015) it presumably formed a convenient starting point. The calculus described here is an extension of the so-called monotonicity calculus, and can explain more examples of entailment. Still, it is much weaker than the logics used in state-of-the-art natural language inference systems (Mineshima et al., 2016, a.o.), and even has less deductive strength than first-order logic (MacCartney, 2009). The dataset of (Bowman et al., 2015) is based on this natural logic calculus, and only contains sentence pairs for which this calculus yields an unambiguous statement. We here describe why the calculus cannot always obtain such a statement, whereas a sound and complete logical system would in some cases. Partly, this is related to the order of the edits made. Moreover, there are reasoning patterns that are undeniably valid, that the calculus cannot establish: de Morgan's laws for quantifiers.

Edit order The principle of the calculus is to look at semantic relations that belong to local edits, that are later joined. This sequential process comes with a drawback. In the calculus, the order in which the edits are processed can influence the result of the calculation. That is, in some cases the result of the calculation is a weak statement (a combination of possible relations), whereas it would be possible to obtain a stronger result if the edits were made in a different order.

De Morgan's laws Of course, equivalence holds trivially for pairs of identical sentences. However, there are other pairs of sentences conceivable that would satisfy equivalence. Here, we list four such patterns, known as de Morgan's laws for quantifiers. Note that there may be other equivalence patterns conceivable in the grammar.

```
                N V  ≡   No N not V
          N not V  ≡   No N V
      Not all N V  ≡   Some N not V
  Not all N not V  ≡   Some N V
```

Where `N` and `V` stand for the same noun and verb in a pair of sentences. However obvious these relations may seem, the natural logic calculus in its current form cannot account for this outcome (MacCartney, 2009). Regardless of the order in which the edits are applied to form the hypothesis from the premise, the end result is too weak: to go from *all hippos bark* to *no hippos not bark* the calculus has to go through either *no hippos bark* or *all hippos not bark*, and in both

cases looses crucial information (see Table 2).

i	x_i	e_i	$\beta(e_i)$	$\beta(x_{i-1}, e_i)$	$\beta(x_0, x_i)$
0	all hippos bark				
1	no hippos bark	SUB(all,no)	|	|	|
2	no hippos not bark	INS(not)	^	|	$\equiv \sqsubset \sqsupset \smile \#$
0	all hippos bark				
1	all hippos not bark	INS(not)	^	|	|
2	no hippos not bark	SUB(all,no)	|	|	$\equiv \sqsubset \sqsupset \smile \#$

Table 2: The calculus fails to predict equivalence.

4.3 Logical Equivalence

Since the natural logic calculus is unable to handle the de Morgan's laws, the dataset does not cover cases of equivalence other than identical sentences. This represents an opportunity to test the extent to which the networks may generalize. We have generated all the 160 sentence pairs in the grammar that instantiate these laws. Using the trained TreeRNN's and comparison layer, we find that the networks do not classify these sentence pairs as 'equivalent' above chance levels (nor do they classify them as 'forward' or 'backward entailment', Michael Repplinger, p.c.).

The sentence representations and linear shifts associated to these sentences are displayed in Figure 6. What this graph (and projections on combinations of other principal components not shown) show, is that 'equivalence' is not a unified geometric relation in the sentence vector space: **(1)** Equivalent but non-identical sentences are not mapped to the same or similar points; **(2)** Each of the 4 equivalence patterns (de Morgan laws) corresponds to its own approximate linear shift in this space (in the projection of figure 6, with PC4 against PC3, laws 1 and 4 run more or less parallel, as do laws 2 and 3; however, in a projection with PC4 against PC1, it is rather laws 1 and 2 and laws 3 and 4). There doesn't seem to be global generalizability of the equivalence relation; **(3)** Within each of these 4 patterns, the shift for a sentence pair with one combination of a noun and a verb is quite similar to the shift for a pair with a different combination – the equivalence relation might be locally generalizable.

5 Conclusions

We have in this paper replicated the results by (Bowman et al., 2015), showing that a Tree-RNN can learn to solve a logical reasoning task. We analyzed the learned solution and showed that the Tree-RNN learns to organize its semantic space such that negations of verbs are often implemented as a local mirroring and shifting (i.e., a geometric relation that allows local generalizability), and that the linear shifts between sentence representations tend to cluster together if the same logical relation holds. However, there are no unique clusters per logical relation.

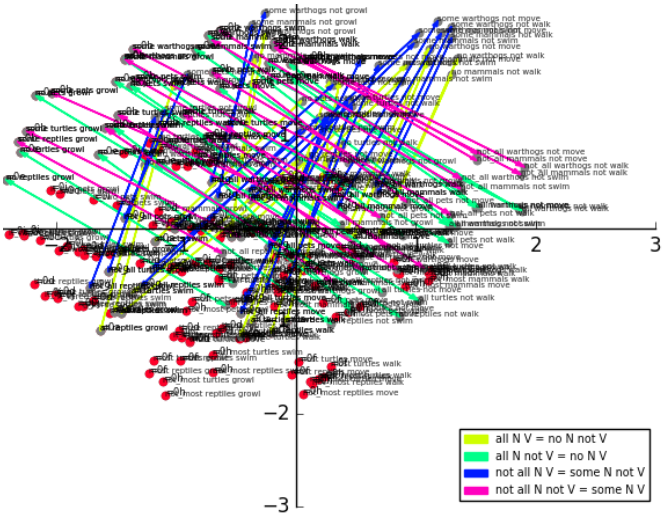


Figure 6: PCA applied to the representations of all possible sentences. Difference vectors corresponding equivalence patterns drawn as colored lines.

So, can neural networks learn logical reasoning? The answer is unfortunately not an unambiguous 'yes' or 'no', but something much more nuanced. First, we must get clear that 'learning logical reasoning', understood as learning to derive new knowledge from previously established facts using previously established valid reasoning patterns, is all about *generalization*. The crucial question for any given learning system is which generalizations it has learned and which ones it hasn't. Has it learned that there is a systematic relationship between mammals and warthogs, between reptiles and turtles, between moving and walking (superset-subset relations)? Has it learned that there is a systematic relationship between moving and not moving, between pets and not pets? The answer to these questions seems to be 'yes', as evidenced by the high performance of the classifier on the test set and the fact that we identified for some of these logical relations (in particular, negation of verbs) a corresponding geometric relationship in the sentence vector space.

Has the system also learned the semantics of the quantifiers, and thus the way the semantics of a quantified expression depends on the semantics of its constituents (Montague, 1973; Benthem, 1986)? Has it learned that 'all' is upward monotone in its first argument, but downward monotone in its second argument (such that for 'all X Y' to entail 'all V W' it must be the case that X entails V – same order – but that W entails Y – reversed order)? It appears the answer to this question is 'probably not', at least for the TreeRNN system that we studied in this paper. We found no ev-

idence that suggests that the networks have learned the systematic relationship between different quantifiers (e.g., that everything entailed by 'some X Y' must also be entailed by 'all X Y', because they have the same monotonicity profile). Rather, it seems that the networks have learned relations between quantifiers on a case by case basis - they generalize locally (nouns and verbs) but not globally (quantifiers). Further investigating this conclusion by selectively leaving out classes of sentences from the training data (rather than sampling at random from all possible sentences) is part of our future work.

Acknowledgements

W.Z. is funded by the Gravitation Program 'Language in Interaction' of the Netherlands Organization for Scientific Research.

References

Johan van Benthem. 1986. *Essays in logical semantics*, volume 29. D. Reidel Pub. Co., Dordrecht.

Samuel R Bowman, Christopher Potts, and Christopher D. Manning. 2015. Recursive neural networks can learn logical semantics. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, page 12.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1):151–271.

Phong Le and Willem Zuidema. 2014. Inside-outside semantics: A framework for neural models of semantic composition. *NIPS 2014 Workshop on Deep Learning and Representation Learning*.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*, pages 140–156. Association for Computational Linguistics.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Citeseer.

Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2016. Building compositional semantics and higher-order inference system for a wide-coverage japanese ccg parser. In *EMNLP*, pages 2236–2242.

R. Montague. 1973. The proper treatment of quantification in ordinary English. In K. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*, Synthese Library. Reidel, Dordrecht.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

L.J.P. van der Maaten and G.E. Hinton. 2008. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

What is not where: the challenge of integrating spatial representations into deep learning architectures

John D. Kelleher
ADAPT Centre for Digital Content Technology
Dublin Institute of Technology, Ireland
john.d.kelleher@dit.ie

Simon Dobnik
CLASP and FLOV
University of Gotenburg, Sweden
simon.dobnik@gu.se

Abstract

This paper examines to what degree current deep learning architectures for image caption generation capture spatial language. On the basis of the evaluation of examples of generated captions from the literature we argue that systems capture *what* objects are in the image data but not *where* these objects are located: the captions generated by these systems are the output of a language model conditioned on the output of an object detector that cannot capture fine-grained location information. Although language models provide useful knowledge for image captions, we argue that deep learning image captioning architectures should also model geometric relations between objects.

1 Introduction

There is a long-traditional in Artificial Intelligence (AI) of developing computational models that integrate language with visual information, see *inter alia.*: (Winograd, 1973; McKevitt, 1995; Kelleher, 2003; Gorniak and Roy, 2004; Kelleher and Kruijff, 2005a; Brenner et al., 2007; Dobnik, 2009; Tellex, 2010; Sjöö, 2011; Dobnik and Kelleher, 2016; Schütte et al., 2017). The goal of this paper is to situate and critically examine recent advances in computational models that integrate visual and linguistic information. One of the most exciting developments in AI in recent years has been the development of *deep learning* (DL) architectures (LeCun et al., 2015; Schmidhuber, 2015). Deep learning models are neural network models that have multiple hidden layers. The advantage of these architectures is that these models have the potential to learn high-level useful features from raw data. For example, Lee et al. (2009) report how their *convolutional deep belief network* "learns useful high-level visual features, such as object parts, from unlabelled images of objects and natural scenes". In brief, Lee et al. show how a deep network trained to perform face recognition learns a hierarchical sequence of feature abstractions: neurons in the early layers in the network learn to act as edge detectors, neurons in later layers react to the presence of meaningful parts of a face (e.g., nose, eye, etc.), and the neurons in the last layers of the network react to sensible configurations of body parts (e.g., nose and eyes and sensible (approximate) offsets between them).

Deep learning models have improved on the start-of-the-art across a range of image and language modelling tasks. The typical deep learning architecture for image modelling is a *convolutional neural network* (CNN) (Lecun et al., 1998) and for language modelling is a *recurrent neural network* (RNN), often using *long short-term memory* (LSTM) units (Hochreiter and Schmidhuber, 1997). However, from the perspective of research into the interface between language and vision perhaps the most exciting aspect of deep learning is the fact that all of these models (both language and vision processing architectures) use a vector based representation. A consequence of this is that deep learning models have the potential to learn multi-modal representations that integrate linguistic and visual information. Indeed, inspired by sequence-to-sequence neural machine translation research (Sutskever et al., 2014), deep learning image captioning systems have been developed that use a CNN to process and encode image data and then pass this vector based encoding of the image to an RNN that generates a caption for the image. Figure 1 illustrates the components and flow of data in an encoder-decoder CNN-RNN image captioning architecture.

The performance of these deep learning image

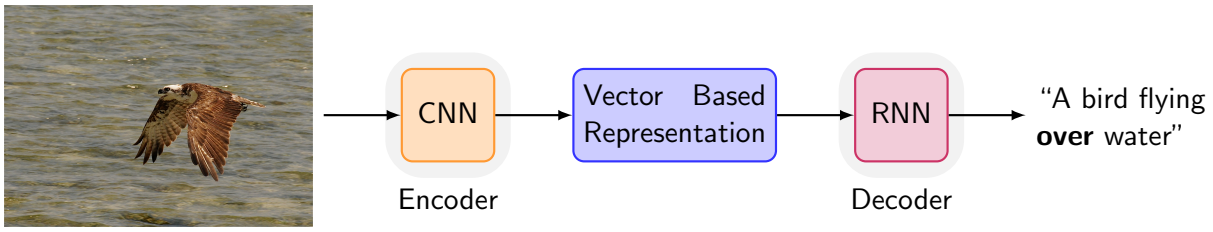


Figure 1: A schematic of a typical encoder-decoder deep learning image captioning architecture in (Xu et al., 2015). The photo is different from but similar to the photo in this example. The current photo is by Jerry Kirkhart (originally posted to Flickr as Osprey Hunting) and is sourced via Wikimedia Commons. It is used here under the Creative Commons Attribution 2.0 generic licence.

captioning systems is impressive. However, the question posed by this paper is whether these systems are actually grounding the semantics of the entire linguistic caption in the image, and in particular whether these systems ground the semantics of spatial relations in the image. The rest of the paper is structured as follows: Section 2 introduces the components of deep learning image captioning architecture in more detail; following this, Section 3 reviews the challenges of grounding language in perception, with a particular focus on spatial language; the paper concludes in Section 5 by posing the question of whether deep learning image captioning architectures as currently constituted are capable of doing justice to the complexity and diversity of factors that affect the production and interpretation of spatial language in visually situated dialogues.

2 The Standard DL Image Captioning Architecture

As mentioned in Section 1 there are two major components within current standard deep-learning image captioning systems, a CNN that processes the image input and encodes some of the information from the image as a vector, and an RNN that takes the vector representation of the image as an input and generates a caption for the image. This section provides an explanation for how each of these components works: Section 2.1 introduces the basic architecture of a CNN and Section 2.2 introduces the basic architecture of an RNN and explains how they can be used to create visually grounded language models.

2.1 Convolutional Neural Networks

CNNs are specifically designed for image recognition tasks, such as handwritten digit recognition (Le Cun, 1989). A well-recognised approach to image recognition is to extract local visual features and combine these features to form higher-order features. A local feature is a feature whose extent within a image is constrained to a small set of neighbouring pixels. For example, for face recognition a system might first learn to identify features such as patches of line or curve segments, and then learn patterns across these low level features that correspond to features such as eyes or a mouth, and finally learn how to combine these body-part features to identify a face.

A key challenge in image recognition is creating a model that is able to recognise if a visual feature has occurred in the image irrespective of the location of the feature in the image:

> "it seems useful to have a set of feature detectors that can detect a particular instance of a feature anywhere on the input plane. Since the precise location of a feature is not relevant to the classification, we can afford to loose some position information in the process" (Le Cun, 1989, p.14)

For example, a face recognition network should recognise the shape of an eye whether the eye is in the top right corner of the image or in the centre of the image. CNNs achieve this translation invariant detection of local visual features using two techniques:

1. weight (parameter) sharing and
2. pooling.

Recall that each neuron in a network learns a function that maps from a set of inputs to an output activation. The function is defined by the set of weights the neuron applies to the inputs it receives and learning the function involves updating the

weights from a set of random initialised values to a set of values that define a function that the network found useful during training in terms of predicting the correct output value. In the context of image recognition a function can be understood as a feature detector which takes a set of pixel values as input and outputs a high-activation score if the visual feature is present in the set of input pixels and a low-activation score if the feature is not present. Furthermore, neurons that share (or use) the same weights implement the same function and hence implement that same feature detector.

Given that a set of weights for a neuron defines a feature detector and that neurons with the same weights implement the same feature detector, it is possible to design a network to check whether a visual feature occurs anywhere in an image by making multiple neurons share the same set of weights but have each of these neurons inspect different portions of the image in such a way so that together the neurons cover the whole image.

For example, imagine we wish to train a network to identify digits in images of 10×10 pixels. In this scenario we may design a network so that one of the neurons in the network inspects the pixels in the top-left corner of the figure to check if a visual feature is present. The image at the top of Figure 2 illustrates such a neuron. This neuron inspects the pixels $(0,0),\ldots,(2,2)$ and applies the function defined by the weight vector $< w_0,\ldots,w_8 >$. This neuron will return a high activation if the appropriate pixel pattern is present in the pixels input to the function and low otherwise. We can now create a copy of this neuron that uses the same weights $< w_0,\ldots,w_8 >$ but which inspects a different set of pixels in the image: the image at the bottom of Figure 2 illustrates such a neuron, this particular neuron inspects the pixels $(0,1),\ldots,(2,3)$. If the visual feature that the function defined by the weight vector $< w_0,\ldots,w_8 >$ occurs in either of the image patches inspected by these two neurons, then one of the neurons will fire.

Extending the idea of a set of neurons with shared weights inspecting a full image results in the concept of a feature map. In a CNN a feature map consists of a group of neurons that share the same set of weights on their inputs. This means that each group of neurons that share their weights learns to identify a particular visual feature and each neuron in the group acts as a detector for that

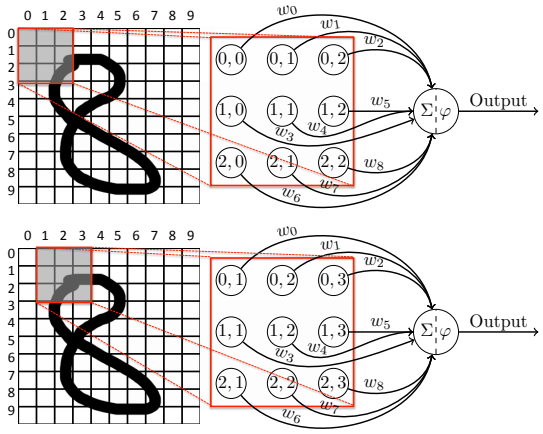


Figure 2: Two neurons connected to different areas in the input image (i.e., with different receptive fields) but which shared the same weights: $w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8$

feature. In a CNN the neurons within each group are arranged so that each neuron examines a different local region in the image. The set of pixels that each neuron in the feature map inspects is known as the receptive field of that neuron. The neurons and the related receptive fields are arranged so that together the receptive fields cover the entire input image. Consequently, if the visual feature the group detects occurs anywhere in the image one of the neurons in the group will identify it. Figure 3 illustrates a feature map and how each neuron in the feature map has a different receptive field and how the neurons and fields are organised so that taken together the receptive fields of the feature map cover the entire input image. Note that the receptive fields of neighbouring neurons typically overlap. In the architecture illustrated in Figure 3 the receptive fields of neighbouring neurons will overlap by either two columns (for horizontal neighbours) or by two rows (for vertical neighbours). However, an alternative organisation would be to reduce the number of neurons in the feature map and reduce the amount of overlap in the receptive fields. For example, if the receptive fields only overlapped by one row or one column then we would only need half the number of neurons in the feature map to cover the entire input image. This would of course result in a "two-to-one under-sampling in each direction" (Le Cun, 1989).

The idea of applying the same function repeatedly across an input space by defining a set of neurons where each neuron applies the function to a

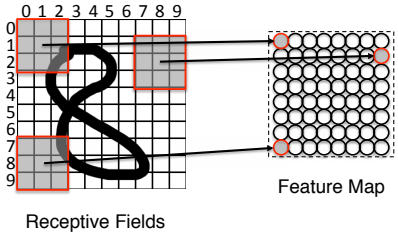


Figure 3: A feature map

different part of the input is very general and can be used irrespective of the type of function being applied. CNNs networks often use the same technique to under-sample the output from a feature map. The motivation for under-sampling is to *discard locational information in favour of generalising the network's ability to identify visual features in a shift invariant manner*. The standard way to implement under-sampling on the output of a feature map is to use a pooling layer, so called as it pools information from a number of neurons in a feature map. Each neuron in a pooling layer inspects the outputs of a subset of the neurons in a feature map, in a very similar way to the way the neuron in the feature map each has a receptive field in the input. Often the function used by neurons in a pooling layer is the *max* function. Essentially, a max pooling neuron outputs the maximum activation value of any of the neurons in the preceding layer that it inspects. Figure 4 illustrates the extension of the feature map in Figure 3 with a pooling layer. The output of the highlighted neuron in the pooling layer is simply the highest activation across the 4 neurons in the feature map that it inspects. Pooling obviously discards locational information at a local level, after pooling the network knows that a visual feature occurred in a region of the image but does not know where precisely within the region the feature occurred.

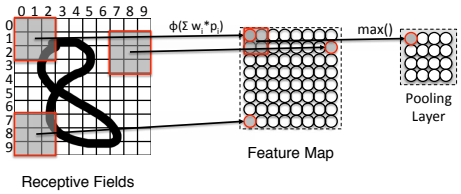


Figure 4: Applying pooling to a feature map

A CNN network is not restricted to only one feature map or one pooling layer. A CNN network can consist of multiple feature maps and pooling layers working in parallel where the outputs of these different streams of processing are finally merged to one or more fully connected layers (see Figure 5). Furthermore, these basic building blocks of feature maps and pooling layers can be sequenced in many different ways: the output of one feature map layer can be used as the input to another feature map layer, and the output of a pooling layer may be the input to a feature map layer. Consequently, a CNN architecture is very flexible and can be composed of multiple layers of feature maps and pooling layers. For example, a CNN could include a feature map that is fed into a pooling layer which in turn acts as the input for a second feature map layer which itself is down-sampled using another pooling layer, and so on, until the outputs of a layer are eventually fed into a fully-connected feed-forward layer where the final prediction is calculated: feature map $\rightarrow$ pooling $\rightarrow$ feature map $\rightarrow$ pooling $\rightarrow$ *dots* $\rightarrow$ fully-connected layer. Obviously with each extra layer of pooling the network discards more and more location information.

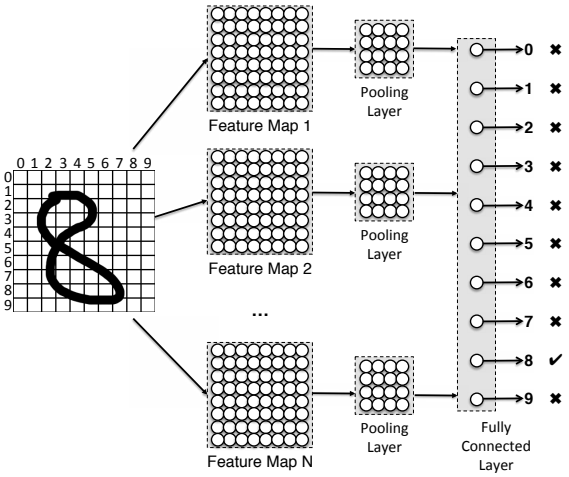


Figure 5: A CNN architecture containing N parallel streams of feature maps and pooling layers feeding into a single fully connected feed-forward layer

2.2 Recurrent Neural Network Language Models

Recurrent Neural Networks (RNN)[1] are an ideal neural network architecture for processing *sequential* data such as language. Generally, RNN models are created by extending a feed-forward neu-

[1]This introduction to Recurrent Neural Networks is based on (Kelleher, 2016).

ral network that has just one hidden layer with a memory buffer, as shown in Figure 6a.

RNNs process sequential data one input at a time. In an RNN the outputs of the neurons in the hidden layer of the network for one input are feed back into the network as part the next input. Each time an input from a sequence is presented to the network the output from the hidden units for that input are stored in the memory buffer overwriting whatever was in the memory (Figure 6b). At the next time step when the next data point in the sequence is considered, the data stored in the memory buffer is merged with the input for that time step (Figure 6c). Consequently, as the network moves through the sequence there is a recurrent cycle of storing the state of the network and using that state at the next time step (Figure 6d).

In order to simplify the following figures we do not draw the individual neurons and connections but represent each layer of neurons as a rounded box and show the flow of information between layers with arrows. Also, we refer to the input layer as x_t, the hidden layer as h_t, the output layer as y_t, and the memory layer as h_{t-1}. Figure 7a illustrates the use of this schematic representation of layers of neurons and the flow of information through an RNN and Figure 7b shows the same network using the shorter naming convention.

Figure 8 demonstrates the flow of information through an RNN as it processes a sequence of inputs. An interesting thing to note is that there is a path connecting each h (the hidden layer for each input) to all the previous hs. Thus, the hidden layer in an RNN at each point in time is dependent on its past. In other words, the network has a memory so that when it is making a decision at time step t it can remember what it has seen previously. This allows the model to take into account data that depends on previous data, for example in sequences. This is the reason why an RNN is useful for language processing: having a memory of the previous words that have been observed in a sequence of a sentence is predictive of the words that follow them.

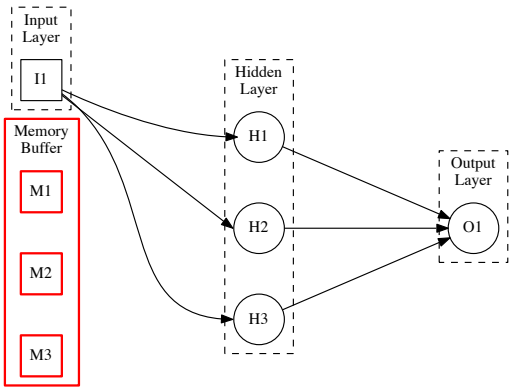
A language model is a computational model that takes a sequence of words as input and returns a probability distribution from which the probability of each vocabulary word being the next word in the sequence can be predicted. An RNN language model can be trained to predict the next word in a sequence. Figure 9 illustrates how information flows through an RNN language model as it processes a sequence of words and attempts to predict the next word in the sequence after each input. The * indicates the next word as predicted by the system. All going well $*Word_2 = Word_2$ but if the system makes a mistake this will not be the case.

When we have trained a language model we can make it to "hallucinate" or generate language by giving it an initial word and then inputting the word that the language model predicts as the most likely next word as the following input word into the model, etc. Figure 10 shows how we can use an RNN language model to generate text by feeding the words the language model predicts back into the model.

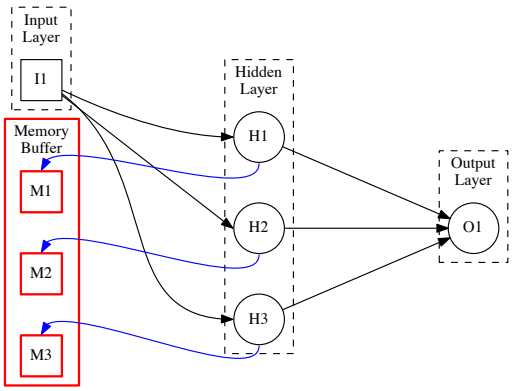
3 Grounding Spatial Language in Perception

The *symbol grounding problem* is the problem of how the meaning of a symbol can be grounded in anything other than other meaningless symbols. Harnad (1990) argues the symbolic representations must be grounded bottom-up from two forms of non-symbolic sensor representations: *iconic representations* which can be understood as sensory experience of objects and events, and *categorial representations* which are feature detectors that are triggered by invariant features of objects and events within these sensory experiences. Given these two foundational non-symbolic representations, a grounded symbolic system can be built up with the elementary symbols of this system being the symbolic names or labels of the object and event categories that are distinguished within the categorical representations of the agent. Essentially, the meaning of an elementary symbol is the categorisation of sensor grounded experience.

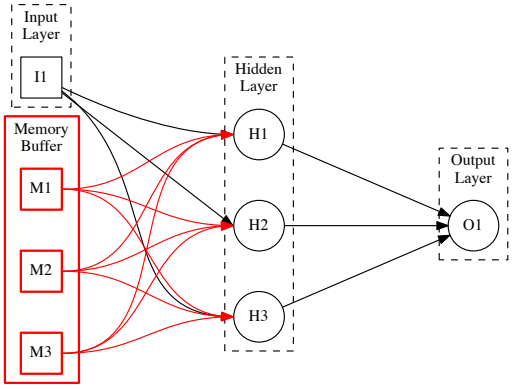
The description "meaningless" in the definition above (originally made by (Harnad, 1990)) should be discussed in relation to the work in distributional semantics (Turney et al., 2010) which has been used very successfully for computational representation of meaning. The reason why distributional semantic representations work is that word contexts capture indirectly latent situations that co-occuring words are all referring to. Distributional semantic models are built from grounded language (which is therefore not "meaningless") it is only that grounded representations are not included in the model. Grounding is expressed indi-



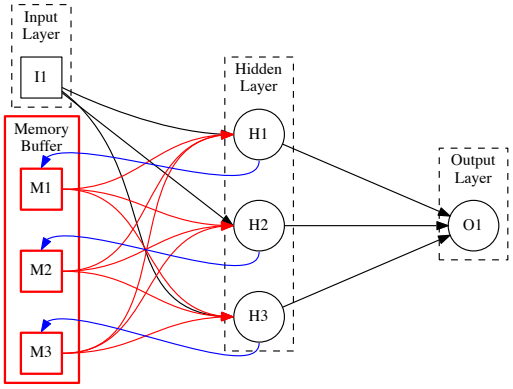
(a) Adding a memory buffer to a feed-forward neural network with one hidden layer



(b) Writing the activation of the hidden layer to the memory buffer after processing the input at time t



(c) Merging the memory buffer with the next input at $t+1$



(d) The cycle of writing to memory and merging with the next input as the network processes a sequence

Figure 6: The flow of data between the memory buffer and the hidden layer in a recurrent neural network

rectly through word co-occurrences.

(Roy, 2005) extends Harnad's approach by setting out a framework of semiotic schemas that ground symbolic meaning in a causal-predictive cycle of action and perception. Within Roy's framework meaning is ultimately grounded in schemas where each schema is a belief network that connects action, perception, attention, categorisation, inference, and prediction. These schemas can be understood as the interface between the external world (reached through the action and perception components of the schemas) and the agents internal cognitive processes (attention, categorisation, inference, and prediction).

Spatial language is an interesting case study in grounding language in perception because linguistic descriptions of perceived spatial relations between objects are intrinsically about the world and as such should be grounded within an agent's perception of that world (Dobnik, 2009; Kelleher and Costello, 2009). The most common form of spatial language discussed in the literature is a *locative expression*. A locative expression is composed of a noun phrase modified by a prepositional phrase that specifies the location of the referent of the noun phrase relative to another object. We will use the word *target object* to refer to the object whose position is being described and the term *landmark object* to refer to the object that the target object's location is described relative to[2], the annotations

[2]The literature on locative expressions uses uses a variety of terms to describe the target and the landmark objects, for a review see (Kelleher, 2003; Dobnik, 2009). Other terms found in the literature for the target object include: *trajector*, *local object*, and *figure object*. Other terms used to describe

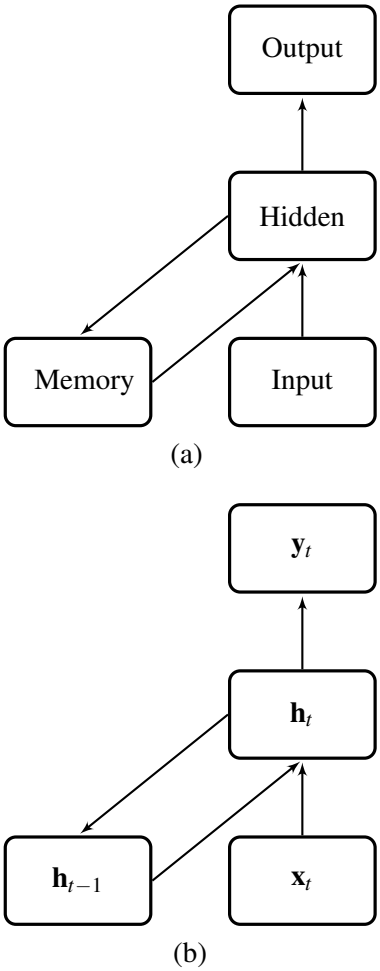
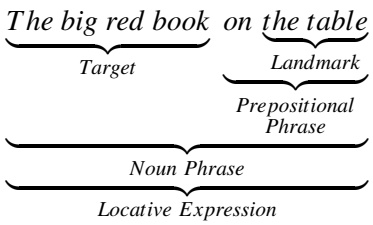


Figure 7: Recurrent Neural Network (RNN)

on the following example locative expression illustrates this terminology:



Previous work on spatial language has revealed a range of factors that impinge on the interpretation of locative expressions. An obvious component in the grounding of a spatial description is the scene geometry and the size and shape of region described by the spatial term within that geometry. The concept of a *spatial template* is used to describe these regions and several experiments have revealed how these templates vary across spatial terms and languages, e.g., (Logan and Sadler, 1996; Kelleher and Costello, 2005; Dobnik and Åstbom, 2017).

It has also been shown that the geometry of a spatial template for a given preposition is affected by a number cognitive and contextual factors, including:

- perceptual attention (Regier and Carlson, 2001; Burigo and Knoeferle, 2015; Kluth and Schultheis, 2014),
- the viewer's perspective on the landmark(s) (Kelleher and van Genabith, 2006),
- object occlusion (Kelleher et al., 2011),
- frame of reference ambiguity and alignment between the landmark object and reference frames (Carlson-Radvansky and Logan, 1997; Burigo and Coventry, 2004; Kelleher and Costello, 2005),
- the location of other *distractor objects* in the scene (Kelleher and Kruijff, 2005b; Costello

the landmark object include: *reference object*, *relatum*, and *ground*.

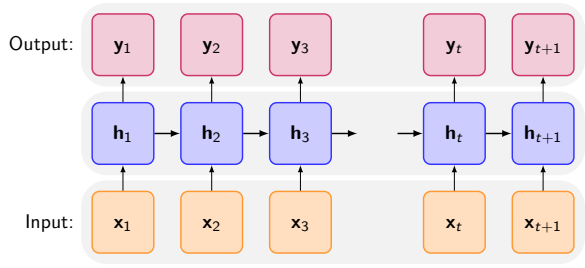


Figure 8: An RNN unrolled in time

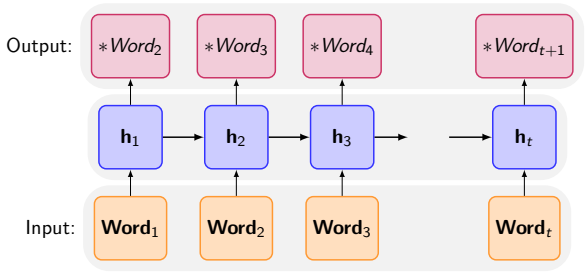


Figure 9: RNN language model unrolled in time

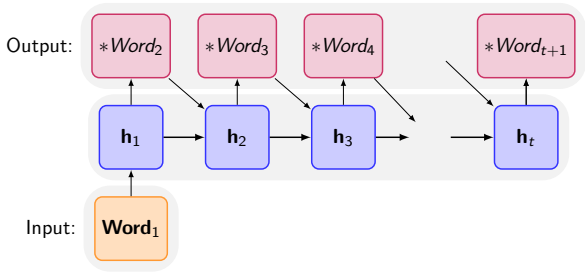


Figure 10: Using an RNN language model to generate a word sequence

and Kelleher, 2006; Kelleher et al., 2006),
- and the richness of the perceptual context (Dobnik and Åstbom, 2017).

The last point is related to the fact that factors affecting semantics of spatial descriptions go beyond scene geometry and include the functional relationships between the target and the landmark (Coventry, 1998; Coventry et al., 2001; Coventry and Garrod, 2004) and force dynamics within the scene (Coventry et al., 2005; Sjöö, 2011). These functional relations can be captured as meanings induced from word distributions (Dobnik and Kelleher, 2013, 2014). Another important factor of (projective) spatial descriptions is their contextual underspecification in terms of the assigned frame of reference which is coordinated through dialogue interaction between conversational participants (Dobnik et al., 2015). It is therefore based on their coordinated intentions in their interaction.

The research in spatial language semantics highlights its multifaceted nature. Spatial language draws upon (i) geometric concepts, (ii) world knowledge (i.e., an understanding of functional relationships and force dynamics), and (iii) perceptual and discourse cues. Thus in order for a computational system to adequately model spatial language semantics it should accommodate all or most of these factors.

4 Spatial Language in DL

The question that this paper addresses is whether deep learning image captioning architectures as currently constituted are capable of grounding spatial language within the images they are captioning. The outputs of these systems are impressive and often include spatial descriptions. Figure 1 (based on an example from (Xu et al., 2015)) provides an indicative example of the performance of these systems. The generated caption in this case is accurate and what is particularly interesting is that it includes a spatial description: *over water*. Indeed, the vast majority of generated captions listed in (Xu et al., 2015) include spatial descriptions, some of which include:[3]

- "A woman is throwing a frisbee *in a park*"
- "A dog is standing *on a hardwood floor*"
- "A group of people sitting *on a boat in the water*".

The fact that these example captions include spatial descriptions and that the captions are often correct descriptions of the input image begs the question of whether image captioning systems are actually learning to ground the semantics of these spatial terms in the images. The nature of neural network systems makes it difficult to directly analyse what a system is learning, however there are a number of reasons why it would be surprising to find that these systems were grounding spatial language. First, recall from the review of grounding in Section 3 that spatial language draws on a variety of information, including:

- scene geometry,
- perceptual cues such as object occlusion,
- world knowledge including functional relationships and force dynamics,
- and coordinated intentions of interacting agents.

Considering only scene geometry, these image captioning systems use CNNs to encode the representation of the input image. Recall from Section 2.1 that CNNs discard locational information through the (down-sampling) pooling mechanism and that such down-sampling may be applied several times within a CNN pipeline. Although it is possible that the encoding generated by a CNN may capture rough relative positions of objects within a scene, it is likely that this encoding is too rough to accommodate the level of sensitivity of spatial descriptions to location that experimental studies of spatial language have found to be relevant (cf. the changes in acceptability ratings in (Logan and Sadler, 1996; Kelleher and Costello, 2005; Dobnik and Åstbom, 2017) as a target object moved position relative to the landmark). The architecture of CNNs also points to the fact that these systems are unlikely to be modelling perceptual cues. CNNs essentially work by identifying what an object is through a hierarchical merging of local visual features that are predictive of the object type. These local visual features are likely to be features that are parts of the object and therefore frequently co-occur with the object label. Consequently, CNNs are unlikely to learn to identify an object type via context and viewpoint dependent cues such as occlusion. Finally, neither a CNN nor an RNN as currently

[3] The emphasis on the spatial descriptions were added here.

used in the image description tasks provide mechanisms to learn force-dynamics or functional relationships between objects (cf. (Coventry et al., 2005; Battaglia et al., 2013)) nor do they take into account agent interaction. Viewed in this light the current image captioning systems appear to be missing most of the key factors necessary to ground spatial language. And, yet they do appear to generate reasonably accurate captions that include spatial descriptions.

There are a number of factors that may be contributing to this apparent ability. First, an inspection of the spatial descriptions used in the generated captions reveals that they tend to include *topological* rather than *projective* spatial prepositions (e.g., *on* and *in* rather than *to the left of* and *above*): *in a forest*, *in a park*, *in a field*, *in the field with trees*, *in the background*, *on a bed*, *on a road*, *on a skateboard*, *at a table*. These spatial descriptions are more underspecified with regard to the location of the target object relative to the landmark object than projective descriptions which also require grounding of direction within a frame of reference. Topological descriptions are semantically adequate already if the target is just proximal to the landmark and hence it is more likely that a caption will be acceptable. Furthermore, it is frequently the case that given a particular label for a landmark it is possible to guess the appropriate preposition irrespective of the image and/or the target object type or location. Essentially the task posed to these systems is to fill the blanks with one of *at*, *on*, *in*:

- *TARGET* _____ *a field*,
- *TARGET* _____ *the background*,
- *TARGET* _____ *a road*,
- *TARGET* _____ *a table*.

Although the system may get some of the blanks wrong it is likely to get many of them right. This is because the system can use distributional knowledge of words which captures some grounding indirectly as discussed in Section 3. Indeed, recent research has shown that co-occurrence of nouns with a preposition within a corpus of spatial descriptions can reveal functional relations between objects referred to by the nouns (Dobnik and Kelleher, 2013, 2014). Word co-occurrence is thus highly predictive of the correct preposition. Consequently, language models trained on image description corpora indirectly model partially grounded functional relations, at least within the scope of the co-occurrence likelihood of prepositions and nouns.

The implication of this is that current image captioning systems do not ground spatial descriptions in the images they take as input. Instead, the apparent ability of these systems to frequently correctly use spatial prepositions to describe spatial relations within the image is the result of the RNN language model learning to predict the most likely preposition to be used given the target and landmark nouns where these nouns are predicted from the image by the CNN.

There is a negative and a positive side to this conclusion. Let's start with the negative side. The distinction between cognitive representations of *what* something is versus *where* something is has a long tradition in spatial language and spatial cognition research (Landau and Jackendoff, 1993). These image captioning systems would appear to be learning representations that allow them to ground the semantics of *what*. But they are not learning representations that enable them to ground the semantics of *where*. Instead, they rely on the RNN language model to make good guesses of the appropriate spatial terms to use based on word distributions. The latter point introduces the positive side. It is surprising how much and how robustly semantic information can be captured by distributional language models. Of course, language models cannot capture the geometric relations between objects, for example they are not able to distinguish successfully the difference in semantics between *the chair is to the left of the table* and *the chair is to the right of the table* as *left* and *right* would occur in exactly the same word contexts. However, as we argued in Section 3 spatial language is not only spatial but also affected by other sources of knowledge that leave an imprint in the word distributions which capture relations between higher-level categorical representations built upon the elementary grounded symbols (Harnad, 1990). It follows that some categorical representations will be closer to and therefore more grounded in elementary symbols, something that has been shown for spatial language (Coventry, 1998; Coventry et al., 2001; Coventry and Garrod, 2004; Dobnik and Kelleher, 2013, 2014). In conclusion, it follows that successful computational models of spatial language require both kinds of knowledge.

5 Conclusions

In this paper we examined the current architecture for generating image captions with deep learning and argued that in its present setup they fail to ground the meaning of spatial descriptions in the image but nonetheless achieve a good performance in generating spatial language which is surprising given the constraints of the architecture that they are working with. The information that they are using to generate spatial descriptions is not spatial but distributional, based on word co-occurrence in a sequence as captured by a language model. While such information is required to successfully predict spatial language, it is not sufficient. We see at least two useful areas of future work. On one hand, it should be possible to extend the deep learning configurations for image description to take into account and specialise to learn geometric representations of objects, just as the current deep learning configurations are specialised to learn visual features that are indicative of objects. The work on modularity of neural networks such as (Andreas et al., 2016; Johnson et al., 2017) may be relevant in this respect. On the other hand, we want to study how much information can be squeezed out of language models to successfully model spatial language and what kind of language models can be built to do so.

Acknowledgements

The research of Kelleher was supported by the ADAPT Research Centre. The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Funds.

The research of Dobnik was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at Department of Philosophy, Linguistics and Theory of Science (FLoV), University of Gothenburg.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *CoRR* abs/1601.01705:1–10.

Peter W. Battaglia, Jessica B. Hamrick, and Joshua B. Tenenbaum. 2013. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences* 110(45):18327–18332.

Michael Brenner, Nick Hawes, John D. Kelleher, and Jeremy L. Wyatt. 2007. Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI, pages 2072–2077.

Michele Burigo and Kenny Coventry. 2004. Reference frame conflict in assigning direction to space. In *International Conference on Spatial Cognition*. Springer, pages 111–123.

Michele Burigo and Pia Knoeferle. 2015. Visual attention during spatial language comprehension. *PloS one* 10(1):e0115758.

L.A. Carlson-Radvansky and G.D. Logan. 1997. The influence of reference frame selection on spatial template construction. *Journal of Memory and Language* 37:411–437.

Fintan Costello and John D. Kelleher. 2006. Spatial prepositions in context: The semantics of *Near* in the presense of distractor objects. In *Proceedings of the 3rd ACL-Sigsem Workshop on Prepositions*. pages 1–8.

Kenny R. Coventry, Angelo Cangelosi, Rohanna Rajapakse, Alison Bacon, Stephen Newstead, Dan Joyce, and Lynn V. Richards. 2005. Spatial prepositions and vague quantifiers: Implementing the functional geometric framework. In Christian Freksa, Markus Knauff, Bernd Krieg-Brückner, Bernhard Nebel, and Thomas Barkowsky, editors, *Spatial Cognition IV. Reasoning, Action, Interaction*, Springer Berlin Heidelberg, volume 3343 of *Lecture Notes in Computer Science*, pages 98–110.

Kenny R. Coventry, Mercè Prat-Sala, and Lynn Richards. 2001. The interplay between geometry and function in the apprehension of Over, Under, Above and Below. *Journal of Memory and Language* 44(3):376–398.

K.R. Coventry. 1998. Spatial prepositions, functional relations, and lexical specification. In P. Olivier and K.P. Gapp, editors, *Representation and Processing of Spatial Expressions*, Lawrence Erlbaum Associates, pages 247–262.

K.R. Coventry and S. Garrod. 2004. *Saying, Seeing and Acting. The Psychological Semantics of Spatial Prepositions*. Essays in Cognitive Psychology Series. Lawrence Erlbaum Associates.

Simon Dobnik. 2009. *Teaching mobile robots to use spatial words*. Ph.D. thesis, University of Oxford: Faculty of Linguistics, Philology and Phonetics and The Queen's College, Oxford, United Kingdom. 289 pages. http://www.dobnik.net/simon/documents/thesis.pdf.

Simon Dobnik and Amelie Åstbom. 2017. (Perceptual) grounding as interaction. In Volha Petukhova and Ye Tian, editors, *Proceedings of Saardial – Semdial 2017: The 21st Workshop on the Semantics and Pragmatics of Dialogue*. Saarbrücken, Germany, pages 17–26.

Simon Dobnik, Christine Howes, and John D. Kelleher. 2015. Changing perspective: Local alignment of reference frames in dialogue. In Christine Howes and Staffan Larsson, editors, *Proceedings of goDIAL – Semdial 2015: The 19th Workshop on the Semantics and Pragmatics of Dialogue*. Gothenburg, Sweden, pages 24–32.

Simon Dobnik and John D. Kelleher. 2013. Towards an automatic identification of functional and geometric spatial prepositions. In *Proceedings of PRE-CogSsci 2013: Production of referring expressions – bridging the gap between cognitive and computational approaches to reference*. Berlin, Germany, pages 1–6.

Simon Dobnik and John D Kelleher. 2014. Exploration of functional semantics of prepositions from corpora of descriptions of visual scenes. In Anja Belz, Marie-Francine Moens, and Alan F. Smeaton, editors, *Proceedings of the 1st Technical Meeting of the European Network on Integrated Vision and Language (V&L Net) a Workshop at the 25th International Conference on Computational Linguistics (COLING)*. Association of Computational Lingustics, Dublin, Ireland, pages 33–37.

Simon Dobnik and John D. Kelleher. 2016. A model for attention-driven judgements in type theory with records. In Julie Hunter, Mandy Simons, and Matthew Stone, editors, *JerSem: The 20th Workshop on the Semantics and Pragmatics of Dialogue*. New Brunswick, NJ USA, volume 20, pages 25–34.

Peter Gorniak and Deb Roy. 2004. Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research* 21:429–470.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D* 42:335–346.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Fei-Fei Li, C. Lawrence Zitnick, and Ross B. Girshick. 2017. Inferring and executing programs for visual reasoning. *CoRR* abs/1705.03633(n):n.

John D. Kelleher. 2003. *A Perceptually Based Computational Framework for the Interpretation of Spatial Language*. Ph.D. thesis, Dublin City University.

John D. Kelleher. 2016. Fundamentals of machine learning for neural machine translation. In *Proceedings of the Translating Europen Forum 2016: Focusing on Translation Technologies (doi:10.21427/D78012)*. European Commission Directorate-General for Translation.

John D. Kelleher and Fintan Costello. 2005. Cognitive representations of projective prepositions. In *Proceedings of the Second ACL-Sigsem Workshop of The Linguistic Dimensions of Prepositions and their Use in Computational Linguistic Formalisms and Applications.*.

John D. Kelleher and Fintan J. Costello. 2009. Applying computational models of spatial prepositions to visually situated dialog. *Computational Linguistics* 35(2):271–306.

John D. Kelleher and Geert-Jan M. Kruijff. 2005a. A context-dependent algorithm for generating locative expressions in physically situated environments. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

John D. Kelleher and Geert-Jan M. Kruijff. 2005b. A context-dependent model of proximity in physically situated environments. In *Proceedings of the 2nd ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications, Colchester, UK*.

John D. Kelleher, Geert-Jan M. Kruijff, and Fintan Costello. 2006. Proximity in context: an empricially grounded computation model of proximity for processing topological spatial expressions. In *Proceedings ACL/Coling 2006*. Sydney, Australia.

John D. Kelleher, Robert J. Ross, Colm Sloan, and Brian Mac Namee. 2011. The effect of occlusion on the semantics of projective spatial terms: A case study in grounding language in perception. *Cognitive Processing* 12(1):95–108.

John D. Kelleher and Josef van Genabith. 2006. A computational model of the referential semantics of projective prepositions. In P. Saint-Dizier, editor, *Syntax and Semantics of Prepositions*, Kluwer Academic Publishers, Dordrecht, The Netherlands, Speech and Language Processing.

Thomas Kluth and Holger Schultheis. 2014. Attentional distribution and spatial language. In *International Conference on Spatial Cognition*. Springer, pages 76–91.

Barbara Landau and Ray Jackendoff. 1993. "What" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences* 16:217–265.

Yann Le Cun. 1989. Generalization and network design strategies. Technical Report CRG-TR-89-4, University of Tronoto Connectionist Research Group.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, New York, NY, USA, ICML '09, pages 609–616.

G.D. Logan and D.D. Sadler. 1996. A computational analysis of the apprehension of spatial relations. In M. Bloom, P.and Peterson, L. Nadell, and M. Garrett, editors, *Language and Space*, MIT Press, pages 493–529.

P. McKevitt, editor. 1995. *Integration of Natural Language and Vision Processing (Vols. I-IV)*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

T Regier and L. Carlson. 2001. Grounding spatial language in perception: An empirical and computational investigation. *Journal of Experimental Psychology: General* 130(2):273–298.

Deb Roy. 2005. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence* 167(1-2):170–205.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61:85 – 117.

Niels Schütte, Brian Mac Namee, and John D. Kelleher. 2017. Robot perception errors and human resolution strategies in situated human–robot dialogue. *Advanced Robotics* 31(5):243–257.

Kristoffer Sjöö. 2011. *Functional understanding of space: Representing spatial knowledge using concepts grounded in an agent's purpose*. Ph.D. thesis, KTH, Computer Vision and Active Perception (CVAP), Centre for Autonomous Systems (CAS), Stockholm, Sweden.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Stefanie Tellex. 2010. *Natural language and spatial reasoning*. Ph.D. thesis, Massachusetts Institute of Technology.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37(1):141–188.

T. Winograd. 1973. A procedural model of language understanding. In R.C. Schank and K.M. Colby, editors, *Computer Models of Thought and Language*, W. H. Freeman and Company, pages 152–186.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*. pages 2048–2057.

Variational Inference for Logical Inference

Guy Emerson and Ann Copestake
Computer Laboratory
University of Cambridge
{gete2,aac10}@cam.ac.uk

Abstract

Functional Distributional Semantics is a framework that aims to learn, from text, semantic representations which can be interpreted in terms of truth. Here we make two contributions to this framework. The first is to show how a type of logical inference can be performed by evaluating conditional probabilities. The second is to make these calculations tractable by means of a variational approximation. This approximation also enables faster convergence during training, allowing us to close the gap with state-of-the-art vector space models when evaluating on semantic similarity. We demonstrate promising performance on two tasks.

1 Introduction and Background

Standard approaches to distributional semantics represent meanings as vectors, whether this is done using the more traditional *count* vectors (Turney and Pantel, 2010), or using *embedding* vectors trained with a neural network (Mikolov et al., 2013). While vector space models have advanced the state of the art in many tasks, they raise questions when it comes to representing larger phrases. Ideally, we would like to learn representations that naturally have logical interpretations.

There have been several attempts to incorporate vectors into logical representations, and while we do not have space for a full literature review here, we will mention two prominent lines of research. Coecke et al. (2010) and Baroni et al. (2014) propose a tensor-based approach, where vectors are combined according to argument structure. However, this leaves open the question of how to perform logical inference, as vector spaces do not provide a natural notion of entailment. Indeed, Grefenstette (2013) proved that quantifiers cannot be expressed using tensor calculus. Garrette et al. (2011) and Beltagy et al. (2016) incorporate a vector space model into a Markov Logic Network, in the form of weighted inference rules (the truth of one predicate implying the truth of another). This approach requires existing vectors, and assumes we can interpret similarity in terms of inference.

In contrast to the above, Emerson and Copestake (2016) (henceforth E&C) introduced the framework of Functional Distributional Semantics, which represents the meaning of a predicate not as a vector, but as a *function*.

To define these functions, we assume a semantic space $\mathcal{X}$, each point representing the features of a possible individual. We refer to points in $\mathcal{X}$ as 'pixies', intuitively 'pixels' of the space, to make clear they are not individuals – two individuals may be represented by the same pixie. Further discussion of model theory will be given in forthcoming work (Emerson and Copestake, 2017) (henceforth E&C-forth). We take $\mathcal{X}$ to be a vector space, each dimension intuitively representing a feature.

A semantic function maps from the space $\mathcal{X}$ to the range $[0, 1]$. This can be interpreted both in the machine-learning sense of a classifier, and in the logical sense of a truth-conditional function.[1] In the machine learning view, a semantic function is a probabilistic classifier for a binary classification task – each input $x \in \mathcal{X}$ is either an instance of the predicate's class, or it is not. In the logical view, a semantic function specifies what features a pixie needs to have in order for the predicate to be true of it – that is, the predicate's truth conditions.

[1] We take a probabilistic approach, where a predicate has a probability of truth for any pixie. We believe this is a strength of the model, as it can model fuzzy boundaries of concepts. However, we could also use semantic functions in a more traditional logic, by assigning truth when the function's value is above 0.5, and falsehood otherwise. This is equivalent to turning the probabilistic classifier into a 'hard' classifier.

This is related to probabilistic type judgements in the framework of Probabilistic Type Theory with Records (TTR) (Cooper, 2005; Cooper et al., 2015). Working within TTR, Larsson (2013) argues in favour of representing perceptual concepts as classifiers of perceptual input. While TTR represents situations in terms of situation types, a semantic function model defines a semantic space without reference to any types or predicates.

Schlangen et al. (2016) take a similar view, representing meanings as image classifiers. Zarrieß and Schlangen (2017) use a distributional model to help train such classifiers, but do not directly learn logical representations from distributional data.

Our approach to logical inference is related to the work of Bergmair (2010) and Clarke and Keller (2015), who use fuzzy truth values and probabilistic truth values, respectively. However, neither incorporate distributional data.

In contrast to the above, a semantic function model can be trained on a parsed corpus. By defining a generative model, we can apply unsupervised learning: optimising the model parameters to maximise the probability of generating the corpus data.

Our model can be trained on a corpus annotated with Dependency Minimal Recursion Semantics (DMRS) (Copestake et al., 2005; Copestake, 2009). This represents the meaning of a sentence as a semantic dependency graph that has a logical interpretation. An example DMRS graph is shown in Fig. 1 (ignoring quantifiers, and ignoring properties such number and tense). Note that, as these are semantic dependencies, not syntactic dependencies, active and passive voice sentences can be represented with the same graph. This dependency graph could be generated by the probabilistic graphical model shown in Fig. 2. The generated predicates are at the bottom: q might correspond to a verb, p its subject, and r its object. The dependency links (ARG1 and ARG2) are at the top.

Rather than generating the predicates directly, we assume that each predicate is true of a latent, unobserved pixie. For example, the DMRS graph for *the dog chased the cat* has three pixies, corresponding to the dog, the chasing event, and the cat. We can define a generative model for such sets of pixies (the top row of Fig. 2), assuming each link corresponds to a probabilistic dependence; intuitively, different kinds of events occur with different kinds of arguments. In machine learning terms, this forms an undirected graphical model.

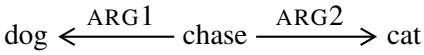


Figure 1: A simplified DMRS graph, illustrating the type of data observed during training. This graph would correspond to sentences like *The dog chased the cat*, or *Cats are chased by dogs*.

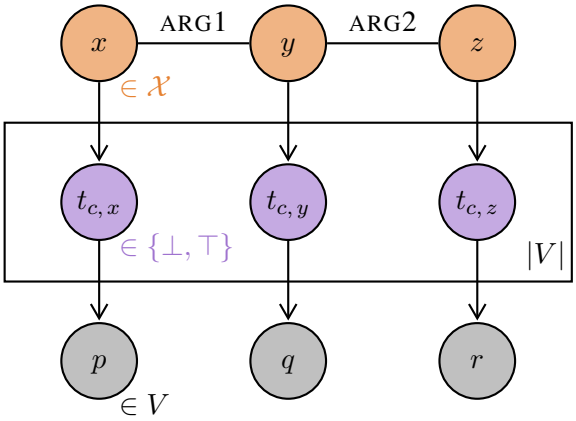


Figure 2: A probabilistic graphical model for Functional Distributional Semantics (E&C Fig. 3). Each node denotes a random variable, but only the bottom row is observed. The plate (middle row) denotes repeating variables across the vocabulary. **Top row:** latent pixies x, y, and z, lying in a semantic space $\mathcal{X}$. Their joint distribution is determined by the DMRS links ARG1 and ARG2. **Middle row:** each predicate c in the vocabulary V is probabilistically true or false for each pixie. **Bottom row:** for each pixie, we observe exactly one predicate, probabilistically chosen out of all predicates that are true of the pixie.

We generate the predicates in three stages (from top to bottom in Fig. 2). First, we generate a set of pixies, with DMRS links specifying probabilistic dependence. Second, we use the semantic functions for all predicates to generate a truth value for each predicate applied to each pixie. Third, for each pixie, we generate a single predicate out of all true predicates. The separation of pixies and truth values gives us a connection with logical inference, as we will see in §2.1.

For a DMRS graph with a different structure, we can define a similar graphical model. For example, for an intransitive sentence, with just a verb and its subject, we can remove the right-hand column of Fig. 2. The model parameters are shared across all such graphs, so we can train our model on a heterogenous set of DMRS graphs.

We follow E&C, and implement this model as shown in Fig. 3. The semantic space $\mathcal{X}$ consists of

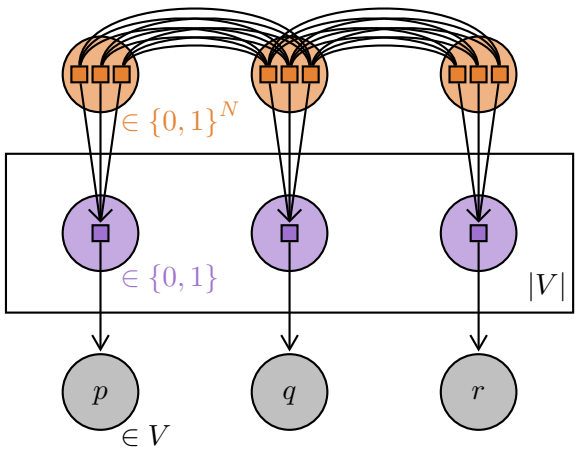


Figure 3: Implementation of the model in Fig. 2. **Top row:** pixies are binary-valued vectors, forming a CaRBM. For each link, connections between the dimensions of the two pixies determine how likely they are to be active at the same time. **Middle row:** each semantic function is a one-layer feedforward network, with a single output interpreted as the probability of truth. **Bottom row:** for each pixie, we generate exactly one predicate, as in Fig. 2.

sparse binary-valued vectors, where a small fixed number of dimensions are 1, and the rest 0. Intuitively, each dimension is a 'feature' of a pixie, and only a small number are present. The joint distribution over pixies is given by a Cardinality Restricted Boltzmann Machine (CaRBM) (Swersky et al., 2012). The semantic functions are one-layer feedforward networks, with a sigmoid activation so the output is in the range $[0, 1]$. The probability of generating a predicate (bottom row) is weighted by the observed frequency of the predicate.

2 Theoretical Contributions

2.1 Logical Inference

The model in Fig. 2 contains, in the middle row, a node for the truth of each predicate for each pixie. Using these nodes, we can convert certain logical propositions into statements about probabilities.

For example, we might be interested in whether one predicate implies another. For simplicity, consider a single pixie x, as shown in Fig. 4. Then, the logical proposition $\forall x \in \mathcal{X},\ a(x) \Rightarrow b(x)$ is equivalent[2] to the statement $P(t_{b,x}|t_{a,x}) = 1$. Intuitively, conditioning on $t_{a,x}$ means restricting to those pixies x for which the predicate a is true. If the probability of $t_{b,x}$ being true is 1, then the predicate b is true for all of those x. Similarly, $\exists x \in \mathcal{X},\ a(x) \wedge b(x)$ is equivalent to $P(t_{b,x}|t_{a,x}) > 0$. Furthermore, classical rules of inference hold under this equivalence. For example, from $P(t_{b,x}|t_{a,x}) = 1$ and $P(t_{c,x}|t_{b,x}) = 1$, we can deduce that $P(t_{c,x}|t_{a,x}) = 1$. This is precisely the classical Barbara syllogism. A proof is given in Appendix A.

In practice, when training on distributional data, the conditional probability $P(t_{b,x}|t_{a,x})$ will never be exactly 0 or 1, because the model only implements soft constraints. Nonetheless, this quantity can be very informative: if it is 0.999, then we know that if $a(x)$ is true, it is *almost always* the case that $b(x)$ is also true. So, it represents the degree to which a implies b, in an intuitive sense.

Separate from this notion of inference, we can also consider the similarity of two latent pixies – if a is true of x, and b is true of y, how many features do x and y share? If a and b are antonyms, the truth of one will not imply the truth of the other, but the pixies may share many features.

As Copestake and Herbelot (2012) note, distinguishing synonyms and antonyms requires checking whether expressions are mutually exclusive. We do not have access to such information in our training data, and such cases are inconsistently annotated in our test data (see §3.1). Nonetheless, the model allows us to make such a distinction, which is an advantage over vector space models. Exploiting this distinction (perhaps by using coreference information) would be a task for future work.

To calculate $P(t_{b,x}|t_{a,x})$, we must marginalise out x, because the model actually defines the joint probability $P(x,\ t_{b,x},\ t_{a,x})$. This is analogous to removing bound variables when calculating the truth of quantified expressions in classical logic. Quantifiers will be discussed further by E&C-forth. However, marginalising out x requires summing over the semantic space $\mathcal{X}$, which is intractable when $\mathcal{X}$ has a large number of dimensions. In §2.2, we introduce a variational approximation to make this calculation tractable.

In the general case, there are multiple pixie variables. This opens up the possibility of inferring what is true of one pixie, given what is true of another. For example, we might be interested in what is true of a verb's arguments, which we could ex-

[2] More precisely, the equivalence requires the logic to have 'existential import': *Every A* implies that some *A* exists. This follows from the definition of conditional probability $P(B|A) = P(A \wedge B)/P(A)$, only defined if $P(A) \neq 0$

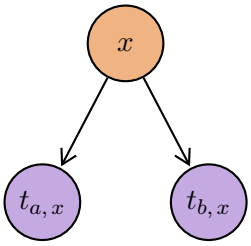


Figure 4: Logical inference for a single pixie x, and two predicates a and b. We have a joint distribution between x and the truth values $t_{a,x}$ and $t_{b,x}$, which lets us consider logical inferences in terms of conditional probabilities, such as $P(t_{b,x}|t_{a,x})$, the probability of b being true, given that a is true.

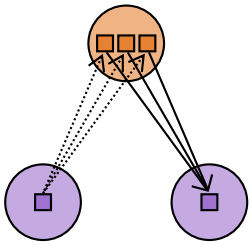


Figure 6: Variational inference for Fig. 4. Exactly calculating the distribution of x given $t_{a,x}$ is intractable, but we can use a mean-field approximation. The dotted lines indicate approximate inference, and the solid lines indicate inference from the mean-field pixie vector.

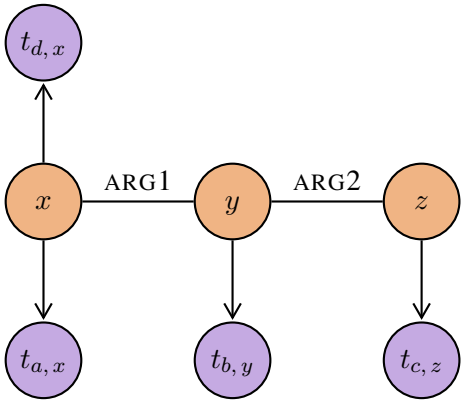


Figure 5: Logical inference for three pixies and four predicates of interest: we know whether a, b, c are true of x, y, z, respectively, and we would like to infer whether d is true of x. The distribution for $t_{d,x}$ depends on all the other truth values, because it is indirectly connected to them via the latent pixies.

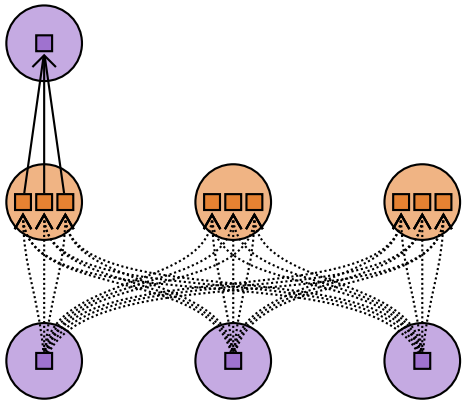


Figure 7: Variational inference for Fig. 5. The ARG1 and ARG2 links are not explicitly represented, but the mean-field probabilities are optimised to approximate the joint distribution induced by the links. Each dimension has an independent probability, but they are jointly optimised, so they depend on all truth values for all pixies.

plore with the three-pixie graph in Fig. 5. We can ask questions such as: if the predicate *paint* is true of an event, what predicates are true of its arguments? A good model might answer that for the ARG1 pixie, *artist* and *person* are likely true, while *democracy* and *aubergine* are likely false.

Just as with the one-pixie case, there is an equivalence between logical propositions and statements about probabilities. For example, $\exists x, y \in \mathcal{X},\ a(y) \wedge b(x) \wedge \text{ARG1}(y, x)$ is equivalent to $P(t_{b,x}|t_{a,y}) > 0$. Note that ARG1 does not correspond to a random variable – it is instead represented directly by the structure of the graphical model (the edges in the top row of Fig. 2 and the middle row of Fig. 5). As before, this conditional probability is never going to be exactly 0 or 1, but it is nonetheless a useful quantity when performing approximate inference, as we will see in §3.2.

2.2 Variational Inference

As explained in the previous section, we can express certain logical propositions as conditional probabilities, but calculating these probabilities exactly is intractable, as it involves summing over the semantic space, which grows exponentially with the number of dimensions. Furthermore, we need to calculate similar conditional probabilities when training the model in the first place.

Instead of summing over the entire space, E&C proposed summing over a small number of carefully chosen pixies, using a Markov Chain Monte Carlo method. However, this algorithm is slow for two reasons. Firstly, many iterations of the Markov chain are required before the samples are useful. Secondly, even if we are not summing over the entire space, many samples are still needed, because the discrete values lead to high variance.

In this section, we introduce a variational inference algorithm, where we directly approximate the distribution over pixies that we need to calculate, and then optimise this approximation. This makes the calculations in the previous section tractable, and also makes training more efficient.

The distribution we need to approximate is $P(x|t_{c,x})$, the probability that a latent pixie x has particular features, given the truth of some predicate c. We use a mean-field approximation: we assume that each dimension has an independent probability q_i of being active, as shown in (1). The approximate probability $Q(x)$ is simply the product of the probabilities of each dimension. Furthermore, we assume that each of these probabilities depends on the average activation of all other dimensions (i.e. the *mean field* activation).

$$P(x|t_{c,x}) \approx Q(x) = \prod_{i|x_i=1} q_i \prod_{i|x_i=0} (1 - q_i) \quad (1)$$

For Q to be a good approximation, it needs to be close to P. We can measure this using the Kullback-Leibler divergence from Q to P.[3] Minimising this quantity is also done in the Expectation Propagation algorithm (Minka, 2001). However, a semantic function model is not in the exponential family, making it difficult to apply Expectation Propagation directly.

Given this mean-field approximation $Q(x)$, we have a a mean-field vector q_i. This vector is not in $\mathcal{X}$, because each dimension is now a value in the range $[0, 1]$, rather than taking one of the values 0 or 1. It represents a 'typical' pixie for these truth values. Furthermore, we have implemented semantic functions as one-layer neural networks, and each weight in the network can be multiplied by a value in the range $[0, 1]$ just as easily as it can be multiplied by 0 or 1. Since a mean-field vector defines a distribution over pixies, applying a semantic function to a mean-field vector lets us approximately calculate the probability that a predicate is true of a pixie drawn from this distribution.

Differentiating the KL-divergence with respect to q_i, and using the above idea that we can apply semantic functions to mean-field vectors, we can derive the update rule given in (2), with a full derivation given in Appendix B. This updates the value of q_i, while holding the rest fixed. Here, $x^{(+i)}$ is the mean-field vector where unit i is fixed to be on, $x^{(-i)}$ is the mean-field vector where unit i is fixed to be off, t_c is the semantic function for the predicate c, D is the number of dimensions of $\mathcal{X}$, and C is the number of active units. Optimising Q can then be done by repeatedly applying this update rule across all dimensions.

$$q_i = \left(1 + \frac{D - C}{C} \frac{t_c\left(x^{(-i)}\right)}{t_c\left(x^{(+i)}\right)}\right)^{-1} \quad (2)$$

This update rule looks at how likely the predicate c is to be true when the dimension x_i is active, and when it is not. If c is much more likely to be true when x_i is active, then q_i will be close to 1. If c is much more likely to be true when x_i is inactive, then q_i will be close to 0. If there is no difference at all, then q_i will be C/D, the expected probability if all dimensions are equally likely.

We can apply this to logical inference, to calculate $P(t_{b,x}|t_{a,x})$, as shown in Fig. 6. We first find the mean-field vector for x, conditioning on the truth of a. This approximates $P(x|t_{a,x})$. Then, we evaluate the semantic function for b on this mean-field vector. This approximates $P(t_{b,x}|t_{a,x})$.

For multiple pixies, the process is similar, as shown in Fig. 7. We have one mean-field vector for each pixie, and we optimise these together. The only difference to the update rule is that, as well as considering how activating one dimension changes the probability of a predicate being true, we also have to consider how likely this dimension is to be active, given the other pixies in the graph. This leads to an extra term in the update rule, as exemplified in (3), where there is a link from x to y. The link has weights W_{ij} which control how likely it is that x_i and y_j are both active.

$$q_i = \left(1 + \frac{D - C}{C} \frac{t_c\left(x^{(-i)}\right)}{t_c\left(x^{(+i)}\right)} e^{-\Sigma_j W_{ij} y_j}\right)^{-1} \quad (3)$$

[3] Variational Bayes minimises the KL-divergence in the opposite direction – that is, the KL-divergence from P to Q. However, for the above approximation, this is infinite: if the number of active units is not equal to the fixed cardinality, then $P(x|t_{c,x}) = 0$ but $Q(x) \neq 0$, giving infinite $Q(x) \log P(x|t_{c,x})$. Furthermore, while Variational Bayes prefers 'high precision' approximations (areas of high Q are accurate), we will prefer 'high recall' approximations (areas of high P are accurate). This is appropriate for two reasons. Firstly, in areas where the number of active units is wrong, Q is bound to be too high, but if we want to sample from Q, we can avoid these areas by using belief propagation, as explained by Swersky et al. (2012). Secondly, in areas where the number of active units is correct, Q will be much higher than P only if there is a dependence between dimensions that Q cannot capture, such as if P is a multi-modal distribution. Because of the definition of an RBM, such a dependence is impossible within one pixie, and combined with the simple form of our semantic functions, such a dependence will be rare between pixies.

Model	SimLex Nouns	SimLex Verbs	WordSim Sim.	WordSim Rel.
Word2Vec	.40	**.23**	**.69**	.46
SVO Word2Vec	.44	.18	.61	.24
Semantic Functions	**.46**	.19	.60	**.14**

Table 1: Spearman rank correlation with average annotator judgements. Note that we would like to have a *low* score on the final column (which measures relatedness, rather than similarity).

Model	Development	Test
Word2Vec, Addition	.50	.47
Semantic Functions	.20	.16
Word2Vec and Sem-Func Ensemble	**.53**	**.49**

Table 2: Mean average precision on the RELPRON development and test sets. Note that this Word2Vec model was trained on a more recent (and hence larger) version of Wikipedia, to match Rimell et al.

3 Experimental Results[4]

Finding a good evaluation task is difficult. Lexical similarity tasks do not require logical inference, while tasks like textual entailment require a level of coverage beyond the scope of this paper. We consider two tasks: lexical similarity, as a simple benchmark, and the RELPRON dataset, which lets us explore a controlled kind of inference.

We trained our model on subject-verb-object (SVO) triples extracted from WikiWoods[5], a parsed version of the July 2008 dump of the English Wikipedia, distributed by DELPH-IN. This resource was produced by Flickinger et al. (2010), using the English Resource Grammar (Flickinger, 2000, 2011), and the PET parser (Callmeier, 2001; Toutanova et al., 2005), with parse ranking trained on the manually treebanked subcorpus WeScience (Ytrestøl et al., 2009).

Our source code is available online.[6] The WikiWoods corpus was pre-processed using the Python packages pydelphin[7] (developed by Michael Goodman), and pydmrs[8] (Copestake et al., 2016).

To speed up training, we initialised our model using random positive-only projections, a simple method for producing reduced-dimension count vectors (QasemiZadeh and Kallmeyer, 2016). Rather than counting each context separately, every context is randomly mapped to a dimension, so each dimension corresponds to the total count of several contexts. These counts can then be transformed into PPMI scores. As with normal PPMI-based count vectors, there are several hyperparameters that can be tuned (Levy et al., 2015) – however, as we are using these vectors as parameters for semantic functions, it should be noted that the optimal hyperparameter settings are not the same.

We compare our model to two vector space models, also trained on Wikipedia. Both use Mikolov et al. (2013)'s skipgram algorithm with negative sampling. "Word2Vec" was trained on raw text, while "SVO Word2Vec" was trained on the same SVO triples used to train our model. We tested these models using cosine similarity.

3.1 Lexical Semantic Similarity

To measure the similarity of two predicates a and b, we use the conditional probability described in §2.1, and illustrated in Figs. 4 and 6. Since this is an asymmetric measure, we multiply the conditional probabilities in both directions, i.e. we calculate $P(t_{a,x}|t_{b,x})P(t_{b,x}|t_{a,x})$.

We evaluated on two datasets which aim to capture similarity, rather than relatedness: SimLex-999 (Hill et al., 2015), and WordSim-353 (Finkelstein et al., 2001), which Agirre et al. (2009) split into similarity and relatedness subsets. Results are shown in Table 1.[9] For each dataset, hyperparameters were tuned on the remaining datasets. As WordSim-353 is a noun-based dataset, it is possible that performance on SimLex-999 verbs could be improved by optimising hyperparameters on a more appropriate development set.

Note that we would like a *low* correlation on the

[4] A fuller set of results, with further discussion, will be given by E&C-forth.

[5] http://moin.delph-in.net/WikiWoods

[6] http://github.com/guyemerson/sem-func

[7] http://github.com/delph-in/pydelphin

[8] http://github.com/delph-in/pydmrs

[9] Performance of Word2Vec on SimLex-999 is higher than reported by Hill et al. (2015). Despite correspondence with the authors, it is not clear why their figures are so low.

relatedness subset of WordSim-353. In the real world, related predicates are unlikely to be true of the same pixies (and the pixies they are true of are unlikely to even share features). For predicates which are true of similar but disjoint sets of pixies, annotations in these datasets are inconsistent. For example, SimLex-999 gives a low score to (*aunt*, *uncle*), but a high score to (*cat*, *dog*). The semantic function model achieves the lowest correlation on the relatedness subset.

Compared to E&C, the gap between the semantic function model and the vector space models has essentially been closed. Which model performs best is inconsistent across the evaluation datasets. This shows that the previously reported lower performance was not due to a problem with the model itself, but rather with an inefficient training algorithm and with poor choice of hyperparameters.

3.2 RELPRON

The RELPRON dataset was produced by Rimell et al. (2016). It consists of 'terms' (all nouns), each paired with up to ten 'properties'. For example, a *telescope* is a *device that astronomers use*, and a *saw* is a *device that cuts wood*. All properties are of this form: a hypernym of the term, modified by a relative clause with a transitive verb. For each term, the task is to identify the properties which apply to this term. Since every property follows one of only two patterns, this dataset lets us focus on semantics, rather than parsing.

A model that captures relatedness can achieve good performance on this dataset – Rimell et al. found that the *other* argument of the verb was the best predictor of the term (e.g. *astronomer* predicts *telescope*). Logically speaking, these predicates do not imply each other. However, Rimell et al. included confounders for a model relying on relatedness – e.g. a *document that has a balance* is an *account*, not the quality of *balance*. In all of their models, this was the top-ranked property for *balance*. By combining a vector model with our model, we hoped to improve performance.

We tested our model using the method described in §2 and illustrated in Figs. 5 and 7: for each term and property, we find the probability of the term being true, conditioned on all predicates in the property. Results are given in Table 2. As noted in §3.1, our model does not capture relatedness, and it performs below vector addition. However, the ensemble outperforms the vector space model alone. This improvement is not simply due to increasing the capacity of the model – increasing the dimensionality of the vector space did not yield this improvement.

Furthermore, inspecting the differences in predictions between the vector space model and the ensemble, it appears that there is particular improvement on the confounders included in the dataset, which require some kind of logical inference. In our ensemble model, for the term *balance*, the top-ranked property is no longer the confounder *document that has a balance*, but instead the correct property *quality that an ear maintains*.

4 Conclusion

We can define probabilistic logical inference in Functional Distributional Semantics, and efficiently calculate it using variational inference. We can use this to improve performance on the RELPRON dataset, suggesting our model can learn structure not captured by vector space models.

Acknowledgements

We would like to thank Emily Bender, for helpful discussion and feedback on an earlier draft.

This work was supported by a Schiff Foundation Studentship.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics*. pages 19–27.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology* 9.

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

Richard Bergmair. 2010. *Monte Carlo Semantics: Robust inference and logical pattern processing with Natural Language text*. Ph.D. thesis, University of Cambridge.

Ulrich Callmeier. 2001. *Efficient parsing with large-scale unification grammars*. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Daoud Clarke and Bill Keller. 2015. Efficiency in ambiguity: Two models of probabilistic semantics for natural language. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS)*. pages 129–139.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis* 36:345–384.

Robin Cooper. 2005. Austinian truth, attitudes and type theory. *Research on Language and Computation* 3(2-3):333–362.

Robin Cooper, Simon Dobnik, Staffan Larsson, and Shalom Lappin. 2015. Probabilistic type theory and natural language semantics. *LiLT (Linguistic Issues in Language Technology)* 10.

Ann Copestake. 2009. 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 Association for Computational Linguistics*. pages 1–9.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyńska. 2016. Resources for building applications with Dependency Minimal Recursion Semantics. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA).

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal Recursion Semantics: An introduction. *Research on Language and Computation* 3(2-3):281–332.

Ann Copestake and Aurélie Herbelot. 2012. Lexicalised compositionality. Unpublished draft.

Guy Emerson and Ann Copestake. 2016. Functional Distributional Semantics. In *Proceedings of the 1st Workshop on Representation Learning for NLP (RepL4NLP)*. Association for Computational Linguistics, pages 40–52.

Guy Emerson and Ann Copestake. 2017. Semantic composition via probabilistic model theory. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on the World Wide Web*. Association for Computing Machinery, pages 406–414.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6(1):15–28.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. In Emily M Bender and Jennifer E Arnold, editors, *Language from a cognitive perspective: Grammar, usage, and processing*, CSLI Publications, pages 31–50.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. 2010. WikiWoods: Syntacto-semantic annotation for English Wikipedia. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA).

Dan Garrette, Katrin Erk, and Raymond Mooney. 2011. Integrating logical representations with probabilistic information using Markov logic. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS)*. Association for Computational Linguistics, pages 105–114.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics (*SEM)*. pages 1–10.

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

Staffan Larsson. 2013. Formal semantics for perceptual classification. *Journal of Logic and Computation* 25(2):335–369.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics (TACL)* 3:211–225.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*.

Thomas P Minka. 2001. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*. pages 362–369.

Behrang QasemiZadeh and Laura Kallmeyer. 2016. Random positive-only projections: PPMI-enabled incremental semantic space construction. In *Proceedings of the 5th Joint Conference on Lexical and Computational Semantics (*SEM)*. pages 189–198.

Laura Rimell, Jean Maillard, Tamara Polajnar, and Stephen Clark. 2016. RELPRON: A relative clause evaluation dataset for compositional distributional semantics. *Computational Linguistics* 42(4):661–701.

David Schlangen, Sina Zarrieß, and Casey Kennington. 2016. Resolving references to objects in photographs using the words-as-classifiers model. In *The 54th Annual Meeting of the Association for Computational Linguistics*. pages 1213–1223.

Kevin Swersky, Ilya Sutskever, Daniel Tarlow, Richard S Zemel, Ruslan R Salakhutdinov, and Ryan P Adams. 2012. Cardinality Restricted Boltzmann Machines. In *Advances in Neural Information Processing Systems 25 (NIPS)*. pages 3293–3301.

Kristina Toutanova, Christopher D Manning, and Stephan Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation* 3(1):83–105.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37:141–188.

Gisle Ytrestøl, , Stephan Oepen, and Dan Flickinger. 2009. Extracting and annotating Wikipedia subdomains. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*.

Sina Zarrieß and David Schlangen. 2017. Is this a child, a girl or a car? Exploring the contribution of distributional similarity to learning referential word meanings. In *Proceedings of the 15th Annual Conference of the European Chapter of the Association for Computational Linguistics*. pages 86–91.

A Logical equivalence

A.1 Proof of general case

Syllogisms are classically expressed in set-theoretic terms. A quantified proposition of the form *Q A are B*, where *Q* is some quantifier, gives constraints on the sizes of the sets $A \cap B$ and $A \setminus B$, and says nothing about the size of B.

For the quantifier $\exists$, we have:

$$|A \cap B| > 0$$

For the quantifier $\forall$, we have the following, where the second constraint assumes existential import:

$$|A \setminus B| = 0$$
$$|A \cap B| > 0$$

From these definitions, we can use standard set theory to prove all and only the valid syllogisms. To show equivalence with our probabilistic framework, we first note that sizes of sets form a measure (the 'counting measure'), and probabilities also form a measure. The above conditions are all constraints on sizes of sets being zero or nonzero, so it suffices to show that the sizes and probabilities are equivalent in the measure-theoretic sense: they agree on which sets have measure zero.

First, we note that $P(B|A) = \frac{P(A \cap B)}{P(A)}$ is defined only when $P(A) > 0$, which will give us existential import.

For $\exists$, we have:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} > 0$$
$$P(A \cap B) > 0$$

We can say nothing further about the probability $P(A \setminus B) = P(A) - P(A \cap B)$, which may be zero or nonzero, just as in the classical case.

For $\forall$, we have:

$$\frac{P(A \cap B)}{P(A)} = 1$$
$$P(A \cap B) = P(A)$$
$$P(A \cap B) = P(A \cap B) + P(A \setminus B)$$
$$P(A \setminus B) = 0$$

And we also have:

$$P(A \setminus B) + P(A \cap B) = P(A) > 0$$
$$P(A \cap B) > 0$$

This demonstrates the equivalence.

A.2 Example

We can prove the Barbara syllogism as follows:

$$P(B|A) = 1 \implies P(A \setminus B) = 0,\ P(A) > 0$$
$$P(C|B) = 1 \implies P(B \setminus C) = 0$$

$$\begin{aligned} P(A \setminus C) &= P(A \cap B \setminus C) + P(A \setminus B \setminus C) \\ &\leq P(B \setminus C) + P(A \setminus B) \\ &= 0 \end{aligned}$$

$$\begin{aligned} P(A \cap C) &= P(A) - P(A \setminus C) \\ &= P(A) > 0 \end{aligned}$$

$$\implies P(C|A) = \frac{P(A \cap C)}{P(A)} = 1$$

B Derivation of update rule

We are trying to optimise Q to minimise the KL-divergence from $Q(x)$ to $P(x|t_{c,x})$:

$$D_{\mathrm{KL}}(P||Q) = \sum_x P(x|t_{c,x}) \log \frac{P(x|t_{c,x})}{Q(x)}$$
$$= \sum_x P(x|t_{c,x}) \big(\log P(x|t_{c,x}) - \log Q(x) \big)$$

Note that the first term is independent of Q. To iteratively optimise one parameter q_i at a time, we take the derivative:

$$\frac{\partial}{\partial q_i} D_{\mathrm{KL}}(P||Q) = -\frac{\partial}{\partial q_i} \sum_x P(x|t_{c,x}) \log Q(x)$$
$$= \sum_{x|x_i=1} P(x|t_{c,x}) \frac{1}{q_i} - \sum_{x|x_i=0} P(x|t_{c,x}) \frac{1}{1-q_i}$$

Now we can rewrite $P(x|t_{c,x})$ as the following. If there is just one pixie, then we can assume a uniform prior over x. For D dimensions, of which C are active, there are $\binom{D}{C}$ different vectors.

$$P(x|t_{c,x}) = \frac{P(x)P(t_{c,x}|x)}{P(t_{c,x})}$$
$$= \frac{t_c(x)}{\binom{D}{C} P(t_{c,x})}$$

Note that $\binom{D}{C} P(t_{c,x})$ is constant in x. Setting the derivative to 0, we have:

$$\sum_{x|x_i=1} t_c(x) \frac{1}{q_i} = \sum_{x|x_i=0} t_c(x) \frac{1}{1-q_i}$$

Summing over all x is intractable, but we can approximate this sum using mean-field vectors for x. For most values of x, $t_c(x)$ will be close to 0, and the regions of interest will be near the mean-field vectors. Let $x^{(+i)}$ denote the mean-field vector when $x_i = 1$ and the total activation of the remaining dimensions is $C-1$, and let $x^{(-i)}$ denote the mean-field vector when $x_i = 0$ and the total activation of the remaining dimensions is C. Both of these vectors can be approximated using the values of q_j for $j \neq i$, scaled so that their sum is correct. Then we have:

$$\binom{D-1}{C-1} t_c(x^{(+i)}) \frac{1}{q_i} \approx \binom{D-1}{C} t_c(x^{(-i)}) \frac{1}{1-q_i}$$
$$t_c(x^{(+i)}) \frac{1}{q_i} \approx \frac{D-C}{C} t_c(x^{(-i)}) \frac{1}{1-q_i}$$

Re-arranging for q_i yields the following, which is the optimal value for q_i, given the other dimensions q_j, and given the above approximations:

$$q_i \approx \left(1 + \frac{D-C}{C} \frac{t_c\left(x^{(-i)}\right)}{t_c\left(x^{(+i)}\right)} \right)^{-1}$$

In the above derivation, we assumed a uniform prior over x, which meant that $P(x|t_{c,x}) \propto t_c(x)$. If there are links between pixies, then this no longer holds, and we instead have $P(x)$ being determined by the RBM weights, which gives the following, where we sum over all links $x \xrightarrow{l} y$, from the pixie x to another pixie y with label l. Each link type l has weights $W^{(l)}_{jk}$ (and for incoming links, we simply take the transpose of this matrix). For clarity, we do not write bias terms.

$$P(x|t_{c,x}) \propto t_c(x) \exp \sum_{x \xrightarrow{l} y} \sum_{j,k} W^{(l)}_{jk} x_j y_k$$

So to amend the update rule, we replace $t_c(x)$ with the above expression, which gives:

$$\left(1 + \frac{D-C}{C} \frac{t_c\left(x^{(-i)}\right) \exp \sum W^{(l)}_{jk} x^{(-i)}_j y_k}{t_c\left(x^{(+i)}\right) \exp \sum W^{(l)}_{jk} x^{(+i)}_j y_k} \right)^{-1}$$

Now note that this ratio of exponentials can be rewritten as:

$$\exp \sum W^{(l)}_{jk} \left(x^{(-i)}_j - x^{(+i)}_j \right) y_k$$

For dimensions $j \neq i$, the difference between the two mean-field vectors will be small, so if $\sum_k W^{(l)}_{jk} y_k$ is on average close to 0, the above expression will be dominated by the value at $j = i$. So, we can approximate it as:

$$\exp - \sum_{x \xrightarrow{l} y} \sum_k W^{(l)}_{ik} y_k$$

This gives the following update rule, which reduces to (3) in the case of a single link:

$$\left(1 + \frac{D-C}{C} \frac{t_c\left(x^{(-i)}\right)}{t_c\left(x^{(+i)}\right)} \exp - \sum_{x \xrightarrow{l} y} \sum_k W^{(l)}_{ik} y_k \right)^{-1}$$

Explainable Machine Translation with Interlingual Trees as Certificates

Aarne Ranta
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
aarne@chalmers.se

Abstract

Explainable Machine Translation (XMT) is an instance of Explainable Artificial Intelligence (XAI). An XAI program does not only return an output, but also an explanation of how the output was obtained. This helps the user to assess the reliability of the result, even if the AI program itself is a black box. As a promising candidate for explanations in MT, we consider interlingual meaning representations—abstract syntax trees in the sense of Grammatical Framework (GF). An abstract syntax tree encodes the translatable content in a way that enables accurate target language generation; the main problem is to find the right tree in parsing. This paper investigates a hybrid architecture where the tree is obtained by a black box robust parser, such as a neural dependency parser. As long as the parser returns a tree from which the target language is generated in a trusted way, the tree serves as an explanation that enables the user to assess the reliability of the whole chain of translation.

1 Introduction

A Machine Translation (MT) system converts text or speech from one language to another. In typical situations, the user of MT knows only one of the languages; otherwise, no translation would be needed. This raises a question: how can the user be sure that the translation is correct, if she has no way to check it? This problem is real because the quality of MT is so low: the state of the art seldom reaches higher than 50% accuracy in the standard BLEU scale (Wu et al., 2016). Typical errors in MT include:

- incorrect grammar in the output (e.g. agreement and word order errors)
- words omitted from the output (e.g. the negation word missing)
- words interpreted in wrong senses (e.g. *bat* rendered as in "baseball bat" instead of "flying mammal")

In recent years, NMT (Neural Machine Translation) has improved the average BLEU scores, in the best cases from the 30's to the 40's (Wu et al., 2016). Improvements have been observed, in particular, in grammaticality, resulting in better fluency of the output. At the same time, NMT may produce output that has little to do with the original message, since it may omit words more freely than previous methods, invent new words, and make "sense" of senseless input (e.g. misspellings) in arbitrary ways. Reliability is thus an even more serious issue with NMT than with older MT techniques.

The problem is similar to many other AI techniques using neural networks: for instance, the problem of creating sense from nonsense is well known in image recognition (Nguyen et al., 2015). The common factor is the black-box character of neural techniques: as there are no explicit rules, it is hard to know how the system "reasons" when producing its output. It is also hard to "fix bugs" in a holistic system: feeding in more data, tuning the parameters, or improving the algorithms may improve the average scores, but it gives no guarantees to correct any particular errors.

Now that AI techniques are used for more and more mission-critical tasks, the demand of reliability increases. XAI (eXplainable Artificial Intelligence) is an initiative in this direction (Gunning, 2017). If ordinary AI stays content with outputs from black boxes, XAI wants to have explanations together with the outputs (see Figure 1.) :

- Why is the result this and not something else?

- How reliable is this result?

The explanation should be some kind of evidence that the user can inspect and understand, and it should make explicit the reasons why the system has made its decisions.

XAI has been promoted as a new idea in the field of AI, even as "the next big disruptive technology" (Prevett, 2017). However, the general idea that programs should produce explanations is older. It is known as **certifying algorithms**: algorithms that don't just yield a result, but also a **certificate** that can be inspected by the user and that justifies the result (McConnell et al., 2011). In the best case, the certificate can even be mechanically checked by an independent program, which decides whether the certificate really justifies the output; see Figure 2.

An an example, consider an expert system that has a large knowledge base to which the user can pose questions. The system may be too large and complex to have a logically sound and complete proof search. Therefore it may use heuristics and probabilistic reasoning to deliver answers. If the system just delivers the answers, it cannot be trusted in any situation. But if every answer is accompanied by an explanation, its reliability can be assessed case by case. In the best case, the explanation may be a logically valid proof, and a checking program can automatically confirm the correctness of the answer. In some other cases, the program may just display a sequence of reasoning steps, and tell the user which steps are uncertain.

In this paper, we propose XMT (eXplainable Machine Translation) as kind of MT, just like XAI more generally is a kind of AI. We approach the problem as an instance of certifying algorithms: what kind of certificates could there be for MT? How could they be verified mechanically, and how could they serve as explanations to a user who knows only one of the languages involved?

2 Explanations vs. certificates vs. proofs

Certifying algorithms should be distinguished from **verified algorithms**. The idea of verification, or proof of correctness, is much older, dating back the 1960's (Burstall, 1969; Hoare, 1969). A verified algorithm has a total proof of correctness—a proof that guarantees every input-output instance. Certifying algorithms deliver proofs of individual instances only. They are useful in situations where total correctness is hard to prove, but proofs of instances are possible. This is often the case with complicated programs, as well as programs whose correctness depends on external components such as compilers and hardware. Certificates can be delivered by black-box programs, such as neural networks or proprietary binary-only programs. They are particularly useful in programs that are known not to be totally correct but still useful. MT programs typically have many of these characteristics.

In addition to the distinction between total and instance-based correctness, there is the dimension of formal vs. informal. In the original set-up of (McConnell et al., 2011), certificates are formal objects that can be mechanically checked by programs. In XAI, explanations need not be formal, but they must be understandable by humans. In some cases, one and the same explanation/certificate can serve both purposes: for instance, a formal proof can be both checkable by a machine and understandable by a trained logician. If the evidence is not conclusive but probabilistic, human inspection is crucial when assessing the seriousness of the weak points.

The two dimensions of evidence are summarized in Table 1. In the top-left corner, we have proofs in the sense of program verification. Below that, we have formal certificates for instances. These types of evidence are applicable in situations where the input-output relation can be formally defined. Since this is not always the case in AI, explanations in XAI are informal evidence that apply to instances of a program.

But there is also a sense in which entire programs can have informal evidence. This is often referred to as "correct by construction", or "correct by design", or, in special cases, "correct by definition". An example of this is the grammar of a programming language, from which the parser of the language is derived. In the case of a programming language, the grammar *defines* the language. There is no question of it being incorrect: it is correct by definition. The parser, on the other hand, can be correct by proof—typically via a proof of an algorithm that derives the parser from the grammar.

Any software system necessarily contains parts that are not provable, but correct only in the informal sense, if at all. The reason is the same as in all mathematics: since proofs cannot be circular, some things must be assumed as axioms. An

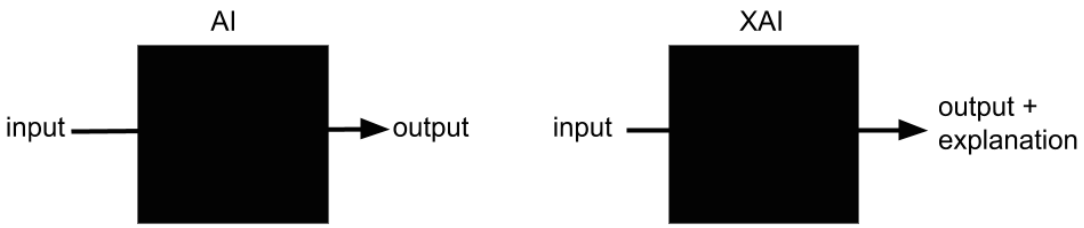


Figure 1: Ordinary black box AI vs. explainable AI.

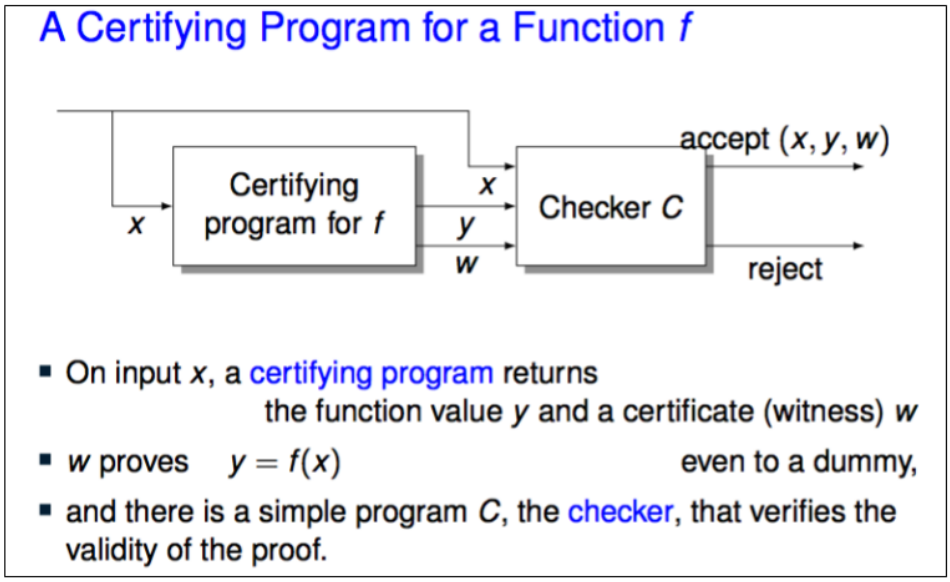


Figure 2: Certifying algorithms, according to (McConnell et al., 2011).

evidence	**formal**	**informal**
for program	proof	correctness by construction
for instance	certificate	explanation

Table 1: The two dimensions of evidence for a program.

axiom system should be convincing in some informal way. It should also be as small as possible—both in the sense of being simple and easy to understand, and in the sense of not assuming things that could be proved.

3 Explanations in machine translation

The traditional MT pipeline follows a sequence of steps shown in Figure 3. A trend in NMT has been to abandon this pipeline in favour of an end-to-end procedure (Lee et al., 2016). Thus some of the best-performing NMT systems convert character strings of the source language to character strings of the target language. They don't assume even the most rudimentary linguistic abstractions, such as words. From a theoretical AI point of view, this makes some sense: end-to-end systems may be better models of the intuitive behaviour of humans, under the assumption that humans' knowledge of language is implicit and not based on linguistic concepts. From the engineering point of view, however, end-to-end translation is problematic:

- it can make mistakes that could be easily avoided by well-established linguistic knowledge;
- its functioning is hard to understand, predict, and improve;
- it leaves no trace (intelligible to humans) of how the translations are obtained.

By not using established linguistic knowledge, an end-to-end black box system is in conflict with the scientific ideal of assuming as little as possible and proving as much as possible. In its simplest form, this ideal says: **don't guess if you know** (Tapanainen and Voutilainen, 1994). It is a virtue of the traditional pipeline that it builds on known things and minimizes guessing. From this pipeline, it is moreover possible to extract explanations of decisions made at different stages. These explanations can help engineers to improve the program and users to assess the realiability of each translation.

The pipeline in Figure 3 is based on the Vauquois triangle, which was originally used as a classification of MT systems (Vauquois, 1968). Thus "lexical transfer" makes only a lexical analysis of the source, and generates the target by looking up dictionary equivalents. We have marked this method as "highly uncertain", because it is not at all sure that it ends up with correct senses of words, not even correct parts of speech, and the result is likely to be grammatically a mess. "Syntactic transfer" climbs to the level of syntax trees before generating the output, and achieves better grammaticality. But this method is still "uncertain", because it may select a wrong interpretation of the input and also miss idiomatic translations. Full correctness is only guaranteed on the highest level, semantic interlingua, which represents the intended meaning precisely. The semantic interlingua moreover makes it easier to generate idiomatic output, because it doesn't need to replicate the same syntactic structures as the source language.[1]

While the certainty of transfer rises when we go up to higher levels, the analysis leading to those levels gets increasingly uncertain. Thus lexical analysis, in the sense of morphology, is for many languages a solved problem, modulo misprints and unknown words. Syntactic parsing, which in our diagram includes part-of-speech disambiguation, is an uncertain process, because syntax is not so well understood as morphology, and also because it involves disambiguation based on superficial information. The final step, semantic interpretation is even more uncertain, since semantics is even less well understood than syntax, and since disambiguation may require any amount of non-linguistic context and world knowledge (Bar-Hillel, 1964; Kay, 1997). Moreover, the interlingua itself has an uncertain status, because there is no known interlingua that expresses with formal precision everything that a natural language can express.[2]

The interlingua can be made complete in the special case of controlled natural languages (CNL), which address limited and well-

[1] The lowest level, character transfer, is the current state of the art in NMT. From the explanation point of view, it could be marked as completely uncertain, as it doesn't deliver any explanation understandable to humans.

[2] It was Descartes who proposed in 1629 that translation be done via a mathematically precise interlingua such that "in a single day one can learn to name every one of the infinite series of numbers, and thus to write infinitely many different words in an unknown language. The same could be done for all the other words necessary to express all the other things which fall within the purview of the human mind." (Letter to Mersenne, (Descartes, 1984)) His experience from creating analytic geometry may have convinced him that such precision is possible beyond numbers. But his final conclusion was not optimistic: "do not hope ever to see such a language in use. For that, the order of nature would have to change so that the world turned into a terrestrial paradise; and that is too much to suggest outside of fairyland."

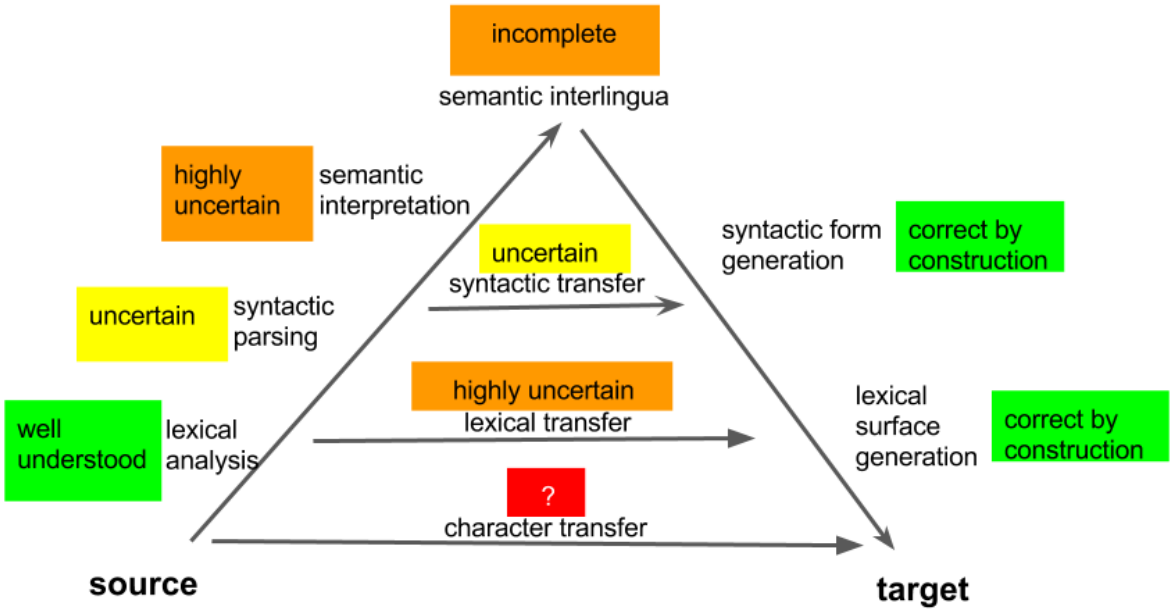


Figure 3: The reliability of the main steps in a translation pipeline, adapted from the Vauquois triangle.

understood domains (Kuhn, 2014; Ranta and Angelov, 2010). For a CNL, it is even possible to make the analysis of source language correct by proof, if the parser is obtained from the grammar by a provably correct algorithm.

However, the problem of ambiguity can arise even in CNLs, especially in multilingual CNL systems: a concept believed to be unambiguous when the language is designed can turn out ambiguous when the next language is added. To give a very simple example, the pronoun *you* in English has many counterparts in other languages (singular, plural, polite, masculine, feminine). A CNL that aims to be both controlled and semantic-preserving must introduce different interlingual concepts for these. At the same time, if it also aims to be natural, it cannot require English to use different expressions for these concepts. In the resulting CNL system, some form of disambiguation is therefore needed when translating from English to other languages. The only guarantee can ultimately be user interaction; many examples of this are shown in (Kay, 1997).

Since the pipeline in Figure 3 contains uncertain parts, it does not solve the problem of translation. But it does suggest what parts to focus on, and it defines checkpoints that can be used in explaining the output. The first thing to notice is that all uncertainty lies on the left side, in analysing the source language. When all information is in place in the semantic interlingua, generating the target language can be performed by mechanical rules of **linearization** in a generation-oriented grammar formalism. The linearization grammar can be made correct by construction. This is of course a substantial task, and it can therefore be hard to get rid of all bugs, but there is no inherent uncertainty like in the analysis part.

The usual way cope with the uncertainty of deep analysis is to give up and proceed with direct transfer instead. The first MT systems typically did this with morphological analysis followed by lexical transfer and synthesis (Hutchins, 2000). Apertium (Forcada et al., 2011) is a contemporary approach in this tradition, showing that the approach works reasonably well for closely related languages such as Spanish and Catalan. More recently, statistical machine translation (SMT) usually starts with words identified by tokenization rather than morphological analysis (Brown et al., 1990). But there is also a variant, Factored SMT, which uses morphologically tagged lemmas (Koehn and Hoang, 2007). The transfer component is in both cases based on **word alignments**, mappings between source and target language words, which are inferred statistically from parallel texts. Other variants of SMT use alignments between **phrases** (short sequences of words, identified by statistical means rather than by grammars) (Och and Ney, 2004), or even between syntax trees (Chiang,

2007).

In all of the above-mentioned systems, translation goes through intermediate steps that could serve as explanations. In phrase-based SMT, for instance, one can look at the way in which the translation is built from aligned phrases. If an alignment looks wrong, it gives a hint that there might be an error in the translation, while it is more difficult to get explanations that convincingly show that the translation is correct. Until recently, a development was seen in SMT towards higher levels of abstraction and use of linguistic knowledge (**hybrid systems**) (Koehn, 2010). This development was stopped abruptly by NMT, which has been moving to the opposite direction (Lee et al., 2016). However, the possibility to inspect the process is an advantage of SMT over NMT that remains when NMT achieves better average scores.

4 Interlingua as explanation

An interlingual meaning representation can serve as a certificate in the sense of (McConnell et al., 2011). Its linearization back to the source language then works as a checker, which can validate the translation even if the interlingua is obtained by an uncertain method such as machine learning. Figure 4 shows the architecture of an XMT system based on these components.

However, this kind of certificate is only a partial guarantee of translation correctness. It can show that the interlingua expression is *one* possible interpretation of the input, but it does not guarantee that it is the correct one. How can we know that it is? How can it serve as an explanation to a user who only knows one of the languages?

The best situation is the one where the user knows the source language and understands the interlingua. She can then compare the interlingua expression with the input and judge whether it is a correct interpretation; in the case of ambiguities, she can select the one that is correct. The user can thus rely on the translation even if she doesn't know the target language.

Translating from a known to an unknown language is the position of **producers** of information. A producer starts with a text originally written in her own language and translates it to other languages to reach new audience. Examples of producers are companies translating their web pages or user manuals and authorities translating legal or administrative information. As producers, they are responsible for the translations that they publish, and the quality requirements are therefore high. For this reason, the state of the art in the producer scenario is manual translation (with some help from computers), because automatic MT is not good enough. However, interlingual translation is a possible technique in many producer cases, because the content to be translated can be limited so that a CNL can be created for it. Once a CNL is in place, it can be linearized to many target languages, which is another typical need of producer translation.

Much of current MT research and most of media publicity have focused on MT for **consumers** of information. A consumer starts with a text written in a foreign language, typically on the internet, and tries to find out what the text is about. No-one is responsible for the translation, and consumers are expected to be aware of its unreliability. Thus if a negation word is dropped and the message is turned to its opposite, neither the producer of the original information nor the provider of the translation system can be blamed. In the producer scenario it could lead to a disaster for both the producer and the translator.

In this paper, our main focus is on the producer scenario, which is the best fit for both the goals of XMT and the methods we are going to propose.

Figure 5 gives an example of how an interlingual expression can explain translation. It shows a screen dump from the mobile GF Offline Translator (Angelov et al., 2014), whose principles will be explained in the Section 5. The screen dump shows three ways to translate a sentence from English to Swedish, resulting from three senses of the verb *work*. They are shown as raw interlingual expressions, where one can discern the constants `work_1_V` `work_2_V`, and `work_on_V2`. One can also see that the first two sentences treat *on the floor* as an adverbial modifier, whereas in the third sentence, *the floor* is the complement of the two-place verb *work on*.

Raw interlingual expressions might not say much to the end user, but they can be made more readable with some engineering, so that the user can select which translation she considers the closest match to the original. Figure 6 shows an example, where the interlingua expressions are converted to graphical syntax trees, and the word sense constants are mapped to their Word-

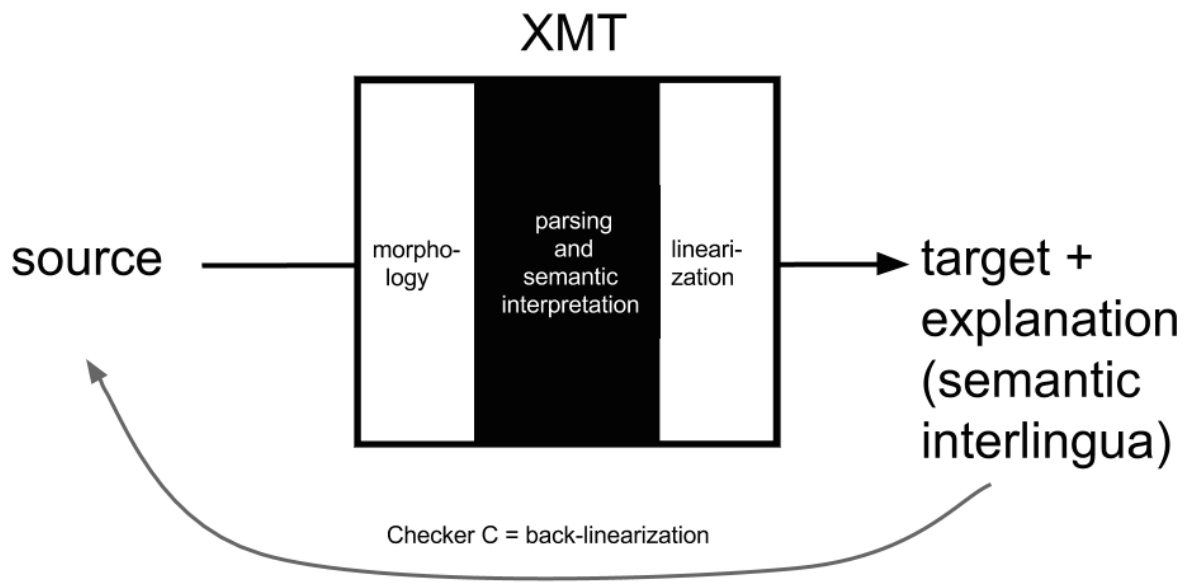


Figure 4: A hybrid XMT architecture using interlingua, grammatical knowledge, and black box parsing.

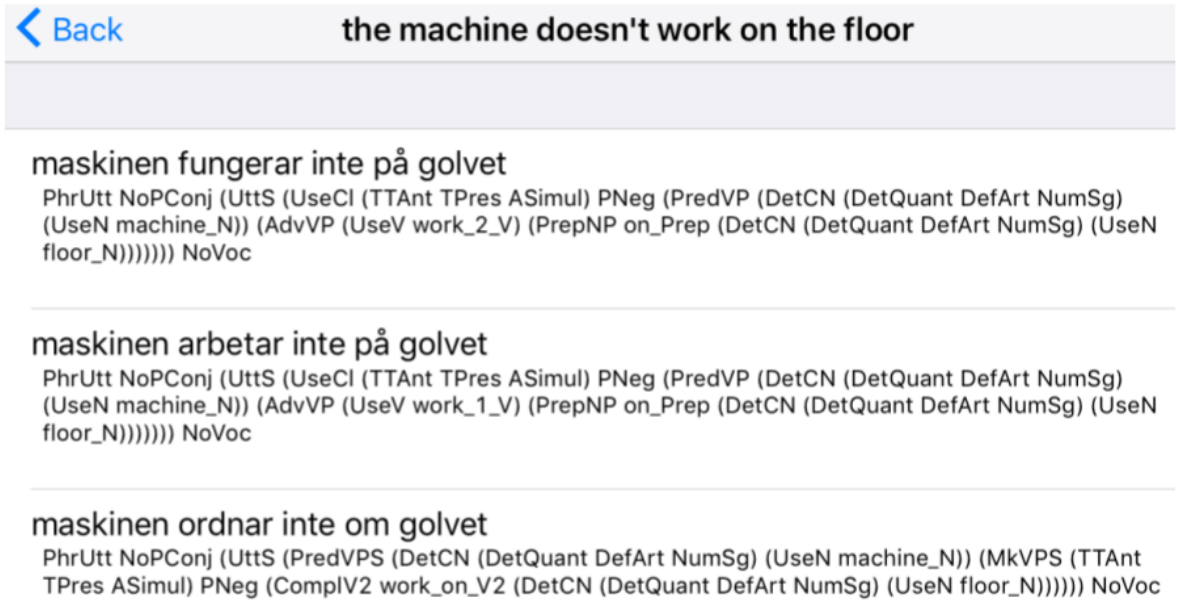


Figure 5: Translation alternatives from English to Swedish and their interlingual expressions: a screen dump from GF Offline Translator.

Net sense explanations.[3]

Yet another form of explanation, accessible to a user who knows at least some other language than the source language, is to use that language as reference. For example, if the translation is from English to Swedish and the user knows English and German, she can inspect the German translation to assess the correctness of the Swedish translation. Since the same tree is the source of both Swedish and German translations, she can trust that the Swedish translation expresses the same meaning as the German one.[4]

5 Translation in GF

Grammatical Framework, GF (Ranta, 2004, 2011) is a grammar formalism designed for interlingual translation and other multilingual applications. The interlingua is called **abstract syntax**, following the tradition of compiler construction, where abstract syntax is an intermediate representation between the source and target languages. The interlingua is not fixed: instead, GF provides a notation for defining new abstract syntaxes. Technically, an abstract syntax is a free algebra, whose elements are nested function applications, similar to LISP terms. Examples are shown in Figure 5 above, using the formal notation of GF, and in Figure 6, as graphical visualizations.

An abstract syntax is defined by a set of categories (such as `NP`, `VP`) and a set of constructor functions, such as

```
fun PredVP : NP -> VP -> Cl
```

which takes as its arguments a noun phrase (`NP`) and a verb phrase (`VP`) and returns a clause (`Cl`).

In addition to the abstract syntax, a GF grammar has a set of **concrete syntaxes**, which specify how abstract syntax trees are **linearized** to strings in some actual languages (usable as both source and target). Technically, a concrete syntax is a homomorphism (compositional mapping) from the abstract syntax algebra to a system of tuples of strings (similar to feature structures in some other formalisms). The tuples represent linguistic variation such as inflection tables, free word order, and discontinuous constituents. At its simplest, linearization returns just strings. For instance, predication could just concatenate the `NP` with the `VP`:

```
lin PredVP np vp = np ++ vp
```

However, natural languages usually also need to express subject-verb agreement. Moreover, they need to keep the main constituents of a clause separate until a higher-level construction decides their order: whether for instance the subject precedes the verb (as in Germanic main clauses) or follows it (as in inverted questions). This variation is expressed by using tuples of strings, which in GF source code are expressed by user-friendly notation for records and projections:

```
lin PredVP np vp = {
  subj = np.s ! Nom ;
  verb = vp.verb ! np.agr ;
  obj  = vp.obj
  }
```

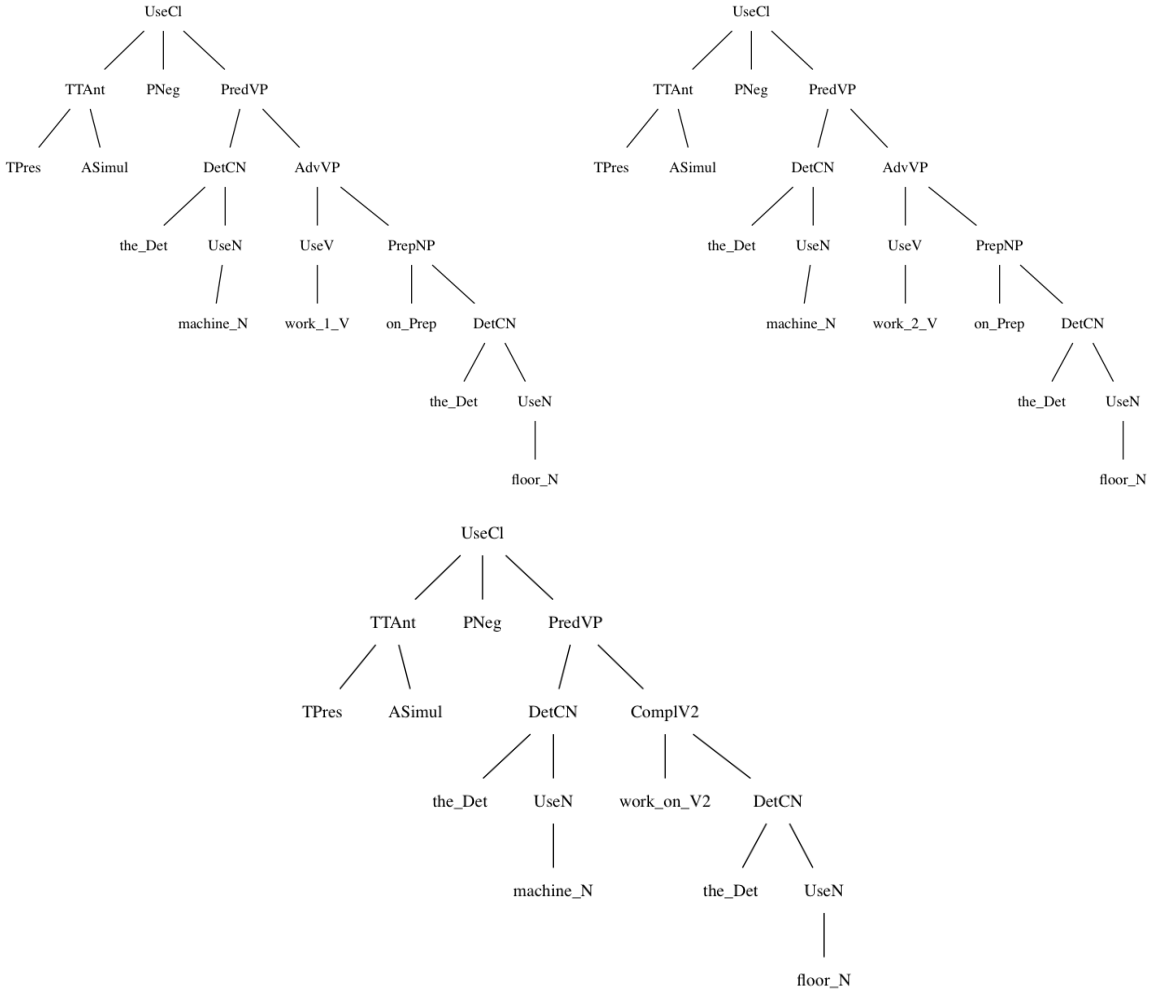
The use of tuples makes GF concrete syntax equivalent to PMCFG (Parallel Multiple Context-Free Grammars, (Seki et al., 1991; Ljunglöf, 2004)). PMCFG enjoys very efficient linearization (of usually linear time complexity), while the parsing problem is also solvable polynomially. Thus PMCFG falls in the category of mildly context-sensitive grammars, which are widely considered adequate for natural languages (Kallmeyer, 2010).[5] In GF, this has been confirmed in practice by implementing a large number of different languages (33 at the time of writing) with a common abstract syntax in the GF Resource Grammar Library, GF-RGL (Ranta, 2009b).

The GF-RGL is intended to cover the most important syntactic structures of natural languages, such as predication (`PredVP` shown above), complementation, coordination, noun phrase construction, relative clauses, questions, and imperatives. It can be compared with the Core Language Engine (Rayner et al., 2000), LinGO Matrix (Bender and Flickinger, 2005), and ParGram (Butt et al., 2002). But due to the abstract/concrete syntax distinction, it can use shared representations for

[3] `http://wordnetweb.princeton.edu/`.These explanations are at the time of writing available in a development version of GF Offline Translator.

[4] This method is not 100% reliable if the reference language has the same ambiguities as the source language, but different from the target language. For instance, translating from English to Chinese with German as reference would miss cases where the parser makes wrong PP attachments: the ambiguities are the same in English and German, but in Chinese, different attachments result in different word orders.

[5] According to some definitions, mildly context-sensitive includes MCFG but not PMCFG, where the "P" means that reduplication of constituents is permitted. In our experience, reduplication is crucial when describing semantic constructions such as intensification. The parsing complexity of PMCFG is still polynomial and weaker than full context-sensitivity.



work 1: (v) *function, work, operate, go, run (perform as expected when applied) "The washing machine won't go unless it's plugged in"; "Does this old car still run well?"; "This old radio doesn't work anymore"*
work 2: (v) *work (exert oneself by doing mental or physical work for a purpose or out of necessity) "I will work hard to improve my grades"; "she worked hard for better living conditions for the poor"*
work on: (v) *work, work on, process (shape, form, or improve a material) "work stone into tools"; "process iron"; "work the metal"*

Figure 6: A visualization of translation alternatives with WordNet sense explanations.

different languages more aggressively. The structures used in Universal Dependencies, UD (McDonald et al., 2013) have shown to be very similar to those in GF-RGL—sufficiently similar to enable effective mappings between GF and UD trees (Kolachina and Ranta, 2016). As we will see in Section 7, the GF-UD connection provides a new way to build hybrid MT systems. But let us first take a look at how GF alone is used in translation.

The applications of GF have typically addressed the producer scenario. The GF grammar then defines a CNL with a few hundred or thousand abstract syntax functions. These functions are built with semantics in mind. They may for instance express logical predicates, operating on semantic categories (Saludes and Xambo, 2011). Thus one might in a CNL for mathematics define

```
fun Equal : Int -> Int -> Prop
```

to express the equality of integers as a two-place propositional function. Such an abstract syntax allows for a more accurate semantic interpretation and reasoning than the raw RGL syntax. It also allows abstraction over linguistically different ways to express the predicate. Thus `Equal` can be expressed in many different ways in a mathematical text:

- adjective-complement predication: *x is equal to y*
- collective predication: *x and y are equal*
- nominalization: *the equality of x and y*
- formula: $x = y$

The rationale for translation is that different languages may favour different ways of expressing the same predicates. Selecting these ways is left to the concrete syntax. The semantic interlingua can therefore support idiomatic translations, which is a useful property even in cases where no reasoning is targeted. The MOLTO Phrasebook (Ranta et al., 2012) is an example of this, collecting a set of everyday phrases with widely varying syntactic realizations.

A CNL-based abstract syntax does not use the RGL as interlingua, but the RGL plays an important role in making the technology productive. It serves as a library to the CNL implementor, so that she doesn't need to care about low-level linguistic details (morphology, agreement, word order), but gets these details from the RGL (Ranta, 2009a). The linearization rules for the CNL are written as mappings of semantic CNL trees to syntactic RGL trees. For instance, the rule corresponding to *x and y are equal* maps the logical predication `(Equal x y)` to a syntactic predication where the noun phrase conjunction of x and y becomes the subject of the verb phrase formed from the adjective *equal*. It could be written as follows:

```
lin Equal x y =
  PredVP
    (ConjNP and_Conj x y)
    (CompA "equal")
```

6 Scaling up GF translation

GF was originally not expected to scale up to the consumer scenario, where semantic interlinguas can't be constructed in any foreseeable future. However, in recent years, improved parsing technology (Angelov and Ljunglöf, 2014) and the availability of open-source multilingual lexica (Virk et al., 2014) have made it possible to use GF in open-domain translation as well. The performance is still behind the state of the art in SMT and NMT in terms of BLEU scores for major languages (Kolachina and Ranta, 2015). But the GF method does have some advantages:

- the interlingual architecture needs only $n+1$ components for n languages to cover all language pairs, instead of $n(n+1)$ as in transfer-based systems;
- grammar-based systems don't need large training data, which helps the treatment of under-resourced languages;
- the interlingual and rule-based architecture is compact enough to be run off-line on mobile devices;
- the translation process can be traced, controlled, debugged, and adapted due to its modular and transparent structure.

This paper is mainly an elaboration on the last-mentioned property. The transparency of the GF translator has been mainly exploited by MT developers. But the need for XMT puts the user in focus: how to use the interlingual representation as a certificate of translation, and as an explanation that the user can inspect?

A simple kind of feedback, which has been in the GF Offline Translator from the beginning, is a classification of translations by using colours. The colours correspond to the level of analysis that the translation is based on:

- **Green** translations use the semantic interlingua.

- **Yellow** translations use the syntactic analysis trees of RGL, when the input cannot be parsed to the semantic interlingua.
- **Red** translations use chunks, which are phrases of different sizes varying from single words to complete sentences, and used when the input cannot be completely parsed syntactically.

The colours reflect the different levels in the Vauquois triangle (Figure 5): semantic interlingua, syntactic transfer, lexical transfer. But there is one difference: all levels use an interlingua rather than transfer. This makes it possible to inspect them more thoroughly than if the analysis was merely in terms of the source language. For example, the interlingua on the lowest level, the chunk interlingua, builds shallow trees from abstract word senses. Displaying these senses tells more about the translation than just showing morphological analyses of the source language, as in traditional lexical transfer.

Furthermore, using the interlingua minimizes the amount of guessing in the translation procedure. In Figure 3, the entire transfer (lexical or syntactic) is uncertain. But assuming that the synthesis of target language from the interlingua is certain, only "half of the translation" remains uncertain: the analysis part. This gives us a new picture of the architecture, shown in Figure 7.

7 Combining grammars and machine learning

Parsing in GF is, theoretically, always available as the inverse of linearization (Ranta, 2004; Ljunglöf, 2004). However, it is not always practically applicable in translation. The input can contain typos and grammatical errors, unusual word orders, or just rare constructions that the grammarian hasn't thought about. In the three-layer GF translator (Figure 7) this may result in the use of the lowest, "red" level more often than desired. This has led to a need to find alternatives to automatically derived grammar-based parsers.

An alternative parser can even be a black box, such as a neural network, as long as it delivers an interlingual expression as certificate. The automatic checking of the certificate by backlinearization will not work quite as usual, if the input does not exactly match the grammar, but it can be decided to be "close enough". What is more important, the abstract syntax tree itself can serve as an explanation to the human.

In addition to robustness, alternative parsers may provide better disambiguation than native GF parsing. The current model used in GF is based on probabilities extracted from the Penn treebank (Marcus et al., 1993; Angelov, 2011). These probabilities are defined on abstract syntax trees and can therefore work for many languages, but they cannot be assumed to be as good as probabilities from language-specific treebanks. What is more, the probability model is context-free, in the sense that it is defined for individual abstract syntax functions, not to their combinations.

A natural way to build an alternative parser is to exploit the similarity of GF and UD (Kolachina and Ranta, 2016; Ranta and Kolachina, 2017). Figure 8 shows a GF tree and the corresponding UD trees in English and French. The UD trees are derived mechanically from the GF tree by means of **dependency configurations**. A dependency configuration is an assignment of labels to each of the arguments of every abstract syntax function. For instance, the predication function has the configuration

```
PredVP : NP -> VP -> Cl
-- nsubj head
```

saying that the `NP` argument contains the `nsubj` and the `VP` argument contains the `head`. In Figure 8, the non-head labels resulting from the configurations are marked below each construction. All non-marked branches correspond to `head` labels. The UD tree is constructed in the following way:

1. Starting from a leaf *U*, go up until a non-head label *L* is reached.
2. Starting from the sister `head` branch of this label, follow the `head` branches until a leaf *V* is reached.
3. Create a dependency arc from *V* to *U* labelled with *L*.

For example, `cat_N` becomes in this way the `nsubj` dependent of `see_V2` in Figure 8. The same method finds the heads and labels of all other leaves as well. To generate the final UD trees, the concrete syntax lexicon is used with word forms determined by agreement relations in the original concrete syntax.

Converting GF trees to UD trees is mechanical and deterministic. It is for the most part uniformly definable in terms of the abstract syntax, even though individual languages also need some

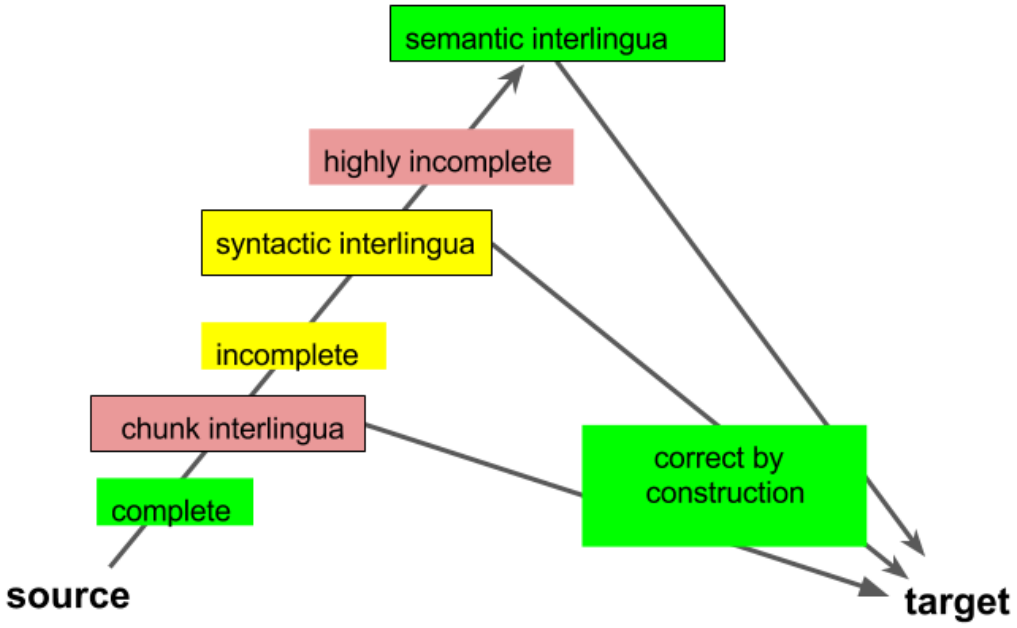


Figure 7: The reliability of different levels of interlingual translation, adapted from the Vauguois triangle. The interlinguas (chunk, syntactic, semantic) are increasingly reliable as sources of target language generation. The analysis methods leading from the source language to the interlingual trees are increasingly incomplete. The generation of target language is always reliable from a given interlingual tree.

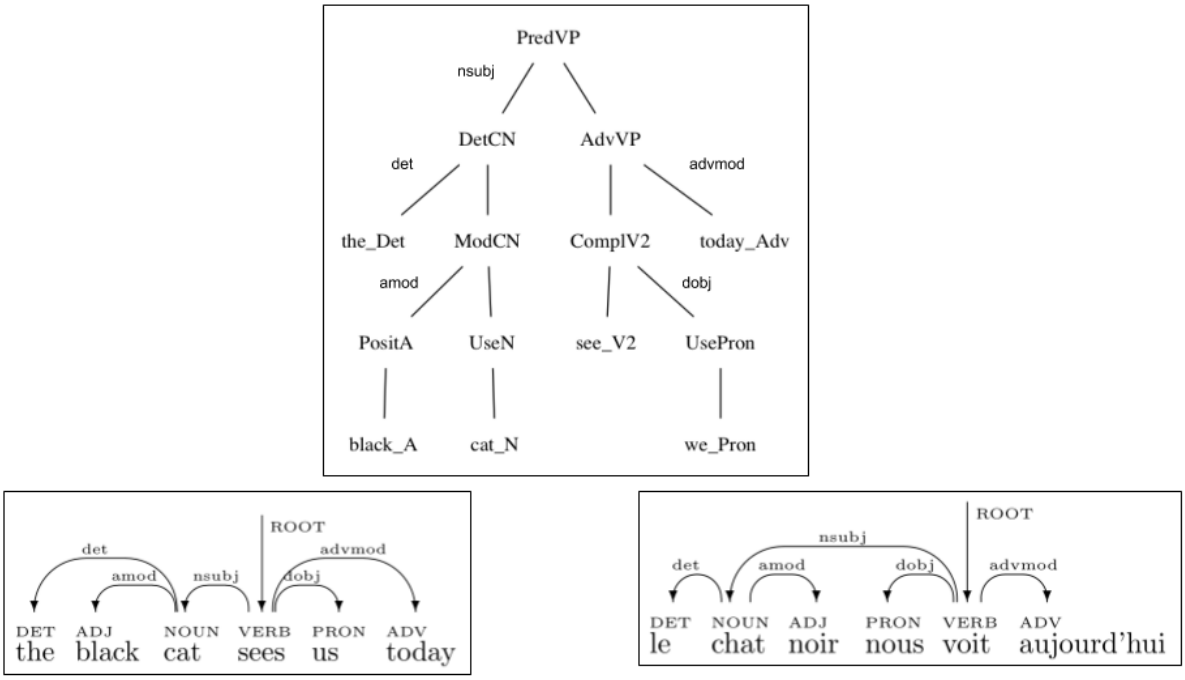


Figure 8: The relation between GF abstract syntax trees and UD dependency trees in English and French.

```
STRING: Fast and friendly service , they know my order when I walk in the door !

root NOUN service_N : N [] {} (4) 4
    amod ADJ fast_A : A [] {} (1) 1
        cc CONJ "and" : Conjand_ [and_Conj : Conj] {} (2) 2
        conj ADJ friendly_A : A [] {} (3) 3
    punct PUNCT "," : Comma_ [] {} (5) 5
    parataxis VERB know_VQ : VQ [know_VS : VS, know_V2 : V2, know_V : V] {} (7) 7
        nsubj PRON they_Pron : Pron [theyFem_Pron : Pron] {} (6) 6
        dobj NOUN order_N : N [] {} (9) 9
            nmod:poss PRON i_Pron : Pron [] {} (8) 8
        advcl VERB walk_V2 : V2 [walk_V : V] {} (12) 12
            mark ADV when_Subj : Subj [when_IAdv : IAdv] {} (10) 10
            nsubj PRON i_Pron : Pron [iFem_Pron : Pron] {} (11) 11
            nmod NOUN door_N : N [] {} (15) 15
                case ADP in_Prep : Prep [] {} (13) 13
                det DET DefArt : Quant [] {} (14) 14
    punct PUNCT StringPN "!" : PN [StringPunct "!" : Punct] {} (16) 16

Eng: fast and friendly service "!" [ they know my order when I walk in the door ]
Fin: nopea ja ystävällinen palvelu "!" [ he tuntevat minun järjestykseni kun minä kävelen ovessa ]
Swe: snabb och vänlig tjänst "!" [ de känner min ordning när jag går i dörren ]
```

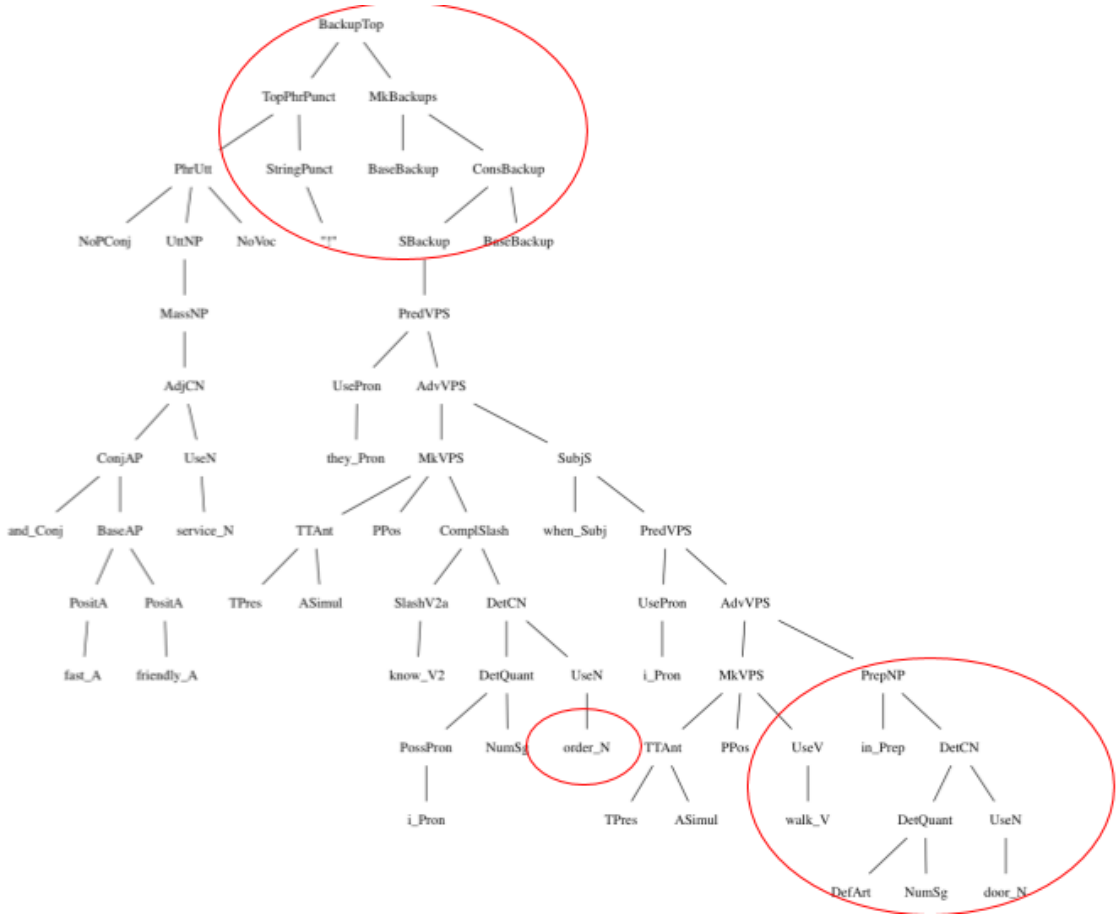


Figure 9: A partially interpreted UD tree, the corresponding GF tree, and linearizations to English, Finnish, and Swedish, with problematic subtrees marked in red.

concrete syntax configurations to cover syncategorematic words such as copulas and tense auxiliaries (Kolachina and Ranta, 2016).

Converting UD trees to GF is a reversal of the GF to UD conversion. It is a non-deterministic search problem, which needs both disambiguation and back-up for structures not covered by the grammar (Ranta and Kolachina, 2017). But the usage of UD as front-end to GF parsing has several potential advantages over raw GF parsing:

- the space of disambiguation is narrowed down by the UD tree;
- the probability model of UD model is (currently) better than the GF model;
- the parser can recover from grammar errors such as agreement errors;
- the parts not covered by the grammar are attached to subtrees of the UD trees instead of being completely unanalysed chunks.

Figure 9 shows an example of UD-based translation from English to Swedish and Finnish, as well as back-linearization to English. The subtree with label `parataxis` is not covered by the GF grammar used; hence, it is attached to the GF tree as an element of the `Backup` category. The UD tree itself can be shown as an explanation, with red colour marking the problematic part. But the GF tree gives even more information; for instance, it shows that the word *order* is interpreted in the sense of "linear order" rather than "order in a restaurant", which is a wrong interpretation.

8 Conclusion

We have outlined some problems and solutions for XMT, Explainable Machine Translation, as a special case of XAI, Explainable Artificial Intelligence. The main problem addressed is that MT in general is unreliable, and its users have a hard time telling when to trust it. The XMT solution is to deliver explanations together with the translation output. Such an explanation can be generated from an interlingual tree, which formalizes the meaning of the input and from which the output can be generated by trusted algorithms. The interlingual tree itself is a formal object that can be checked automatically by translating it back to the input. In this sense, XMT is an instance of certifying algorithms.

However, the automatic checking by back-translation can only guarantee that the translation is *one* possible intepretation of the input. It doesn't prove that it is the correct one, if the input is ambiguous with respect to the interlingua. To assess the correctness of the interpretation, the only way that ultimately works is to let a human decide. For this purpose, XMT generates human-readable explanations from the interlingua, such as graphical visualizations of trees and glosses for the word senses. But there are more things to do on this front, in particular to provide user-friendly ways to compare large numbers of ambiguities, which is a common problem in parser-based translation.

We have presented GF (Grammatical Framework) as a formalism that supports XMT, as it performs translation via an interlingual abstract syntax tree. GF has proved to work in highly multilingual settings with over 30 parallel languages, whenever the domain of translation can be restricted to a CNL (Controlled Natural Language). Scaling up GF translation to open domains is possible by using syntactic and chunk-based interlinguas as backups for the CNL. However, the performance of such systems is not yet on the level of statistical and neural systems for many languages. The new proposal in this paper is a hybrid translation system where UD parsing (Universal Dependencies) is used as a robust front end to GF-based linearization. UD parsers are neural networks or more traditional classifiers trained by machine learning methods. Even though the resulting parser is a black box, it can be safely integrated in a GF-based system, because it generates interlingual expressions that can be partially checked by a machine and fully inspected by humans.

Acknowledgements

I am grateful to John Camilleri, Prasanth Kolachina, and Inari Listenmaa for comments on the paper. The research has been funded by grant nr. 2012-5746 (Reliable Multilingual Digital Communication) from the Swedish Research Council.

References

K. Angelov. 2011. *The Mechanics of the Grammatical Framework*. Ph.D. thesis, Chalmers University of Technology.

K. Angelov, B. Bringert, and A. Ranta. 2014. Speech-enabled hybrid multilingual translation for mobile devices. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the*

Association for Computational Linguistics. Gothenburg, Sweden, pages 41–44.

K. Angelov and P. Ljunglöf. 2014. Fast statistical parsing with parallel multiple context-free grammars. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 368–376.

Y. Bar-Hillel. 1964. *Language and Information*. Addison-Wesley, Reading, MA.

E. M. Bender and D. Flickinger. 2005. Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing IJCNLP-05 (Posters/Demos)*. Jeju Island, Korea.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics* 16(2):76–85.

R. M. Burstall. 1969. Proving properties of programs by structural induction. *The Computer Journal* 12(1):41–48.

M. Butt, H. Dyvik, T. Holloway King, H. Masuichi, and C. Rohrer. 2002. The Parallel Grammar Project. In *COLING 2002, Workshop on Grammar Engineering and Evaluation*. pages 1–7.

D. Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.* 33(2):201–228.

R. Descartes. 1984. *The Philosophical Writings of Descartes: Volume 3, The Correspondence*. Cambridge University Press, Cambridge. Translated by J. Cottingham, R. Stoothoff, D. Murdoch, and A. Kenny, .

M. Forcada, M. Ginesti-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J. Perez-Ortiz, F. Sanchez-Martinez, G. Ramirez-Sanchez, and F. Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation* 24:1–18.

D. Gunning. 2017. Explainable Artificial Intelligence (XAI). https://www.darpa.mil/program/explainable-artificial-intelligence.

C. A. R. Hoare. 1969. An axiomatic basis for computer programming. *Commun. ACM* 12(10):576–580.

W. J. Hutchins. 2000. *Early years in machine translation: Memoirs and biographies of pioneers*. John Benjamins, Amsterdam.

L. Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer Publishing Company, Incorporated.

M. Kay. 1997. The Proper Place of Men and Machines in Language Translation. *Machine Translation* 12(1–2):3–23.

P. Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press.

P. Koehn and H. Hoang. 2007. Factored translation models. In *EMNLP-CoNLL*. pages 868–876.

P. Kolachina and A. Ranta. 2015. GF Wide-coverage English-Finnish MT system for WMT 2015.

P. Kolachina and A. Ranta. 2016. From Abstract Syntax to Universal Dependencies. *Linguistic Issues in Language Technology* 13:1–57.

T. Kuhn. 2014. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics* 40(1):121–170.

J. Lee, K. Cho, and T. Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *CoRR* abs/1610.03017.

P. Ljunglöf. 2004. *The Expressivity and Complexity of Grammatical Framework*. Ph.D. thesis, Dept. of Computing Science, Chalmers University of Technology and Gothenburg University.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

R.M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. 2011. Certifying algorithms. *Computer Science Review* 5(2):119 – 161.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 92–97. http://www.aclweb.org/anthology/P13-2017.

A. M. Nguyen, J. Yosinski, and J. Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*. IEEE Computer Society, pages 427–436.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics* 30(4):417–449.

R. Prevett. 2017. The Next Big Disruptive Trend in Business...Explainable AI. https://disruptionhub.com/next-big-disruptive-trend-business-explainable-ai/.

A. Ranta. 2004. Grammatical Framework: A Type-Theoretical Grammar Formalism. *The Journal of Functional Programming* 14(2):145–189.

A. Ranta. 2009a. Grammars as Software Libraries. In Y. Bertot, G. Huet, J-J. Lévy, and G. Plotkin, editors, *From Semantics to Computer Science. Essays in Honour of Gilles Kahn*. Cambridge University Press, pages 281–308.

A. Ranta. 2009b. The GF Resource Grammar Library. *Linguistics in Language Technology* 2.

A. Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

A. Ranta and K. Angelov. 2010. Implementing Controlled Languages in GF. In *Proceedings of CNL-2009, Marettimo*. volume 5972 of *LNCS*, pages 82–101.

A. Ranta, G. Détrez, and R. Enache. 2012. Controlled language for everyday use: the molto phrasebook. In *CNL 2012: Controlled Natural Language*. volume 7175 of *LNCS/LNAI*.

A. Ranta and P. Kolachina. 2017. From Universal Dependencies to Abstract Syntax. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies, 22 May, Gothenburg Sweden*. pages 107–116.

M. Rayner, D. Carter, P. Bouillon, V. Digalakis, and M. Wirén. 2000. *The Spoken Language Translator*. Cambridge University Press, Cambridge.

J. Saludes and S. Xambo. 2011. The GF mathematics library. In *THedu'11*.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88:191–229.

P. Tapanainen and A. Voutilainen. 1994. Tagging accurately - don't guess if you know. In *In Proceedings of ANLP '94*. pages 47–52.

B. Vauquois. 1968. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In *IFIP Congress (2)*. pages 1114–1122.

S. Virk, K. V. S. Prasad, A. Ranta, and K. Angelov. 2014. Developing an interlingual translation lexicon using WordNets and Grammatical Framework. In *Proceedings of the Fifth Workshop on South and Southeast Asian Natural Language Processing, Workshop at COLING 2014*. pages 55–64.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Bootstrapping dialogue systems: the contribution of a semantic model of interactional dynamics

Arash Eshghi
Interaction Lab
Heriot-Watt University
eshghi.a@gmail.com

Igor Shalyminov
Interaction Lab
Heriot-Watt University
o.lemon@hw.ac.uk

Oliver Lemon
Interaction Lab
Heriot-Watt University
o.lemon@hw.ac.uk

Abstract

We investigate an end-to-end method for automatically inducing task-based dialogue systems from small amounts of unannotated dialogue data. The method combines an incremental, semantic grammar formalism - Dynamic Syntax (DS) and Type Theory with Records (DS-TTR) with Reinforcement Learning (RL), where language generation and dialogue management are treated as one and the same decision problem. The systems thus produced are *incremental*: dialogues are processed word-by-word, shown in prior work to be essential in supporting more natural, spontaneous dialogue. We hypothesised that given the rich linguistic knowledge present within the grammar, our model should enable a combinatorially large number of interactional variations to be processed, even when the system is trained from only a few dialogues. Our experiments show that our model can process 70% of the Facebook AI bAbI data-set - a set of unannotated dialogues in a 'restaurant-search' domain even when trained on only 0.13% of the data-set (5 dialogues). This remarkable generalisation property results from the structural knowledge and constraints present within the grammar, and highlights limitations of recent state-of-the-art systems that are built using machine learning techniques only.

1 Introduction

Meaning is highly activity-specific, in that the action that a particular sequence of words is taken to perform, together with any perlocutionary effect that action might give rise to, is severely underdetermined in the absence of a particular overarching activity, or a 'language-game'. Wittgenstein famously argued that the structure of a language-game, or how actions fit together to form a coherent whole, is irreducible. Arguably, this is the most unyielding obstacle facing not only theoretical approaches to pragmatics, but also dialogue system developers today. This suggests that particular dialogue structures are emergent, learned, and very frequently adjusted during interaction (Mills and Gregoromichelaki, 2010; Mills, 2011; Healey, 2008)

Despite this, recent and ongoing work in formal dialogue modelling suggests that not only language processing mechanisms, but also certain basic principles of contextual dynamics in dialogue do generalise across domains (Ginzburg, 2012; Kempson et al., 2016; Eshghi et al., 2015; Kempson et al., 2015; Purver et al., 2010). Even in a simple domain, there's a lot of *interactional variation* that does not ultimately affect the overall communicative goal of a dialogue. For example, the dialogues in Fig. 1 (specifically the top two rows, where the lexicon is held constant) all lead to a context in which the user wants to buy a phone by LG. These dialogues can be said to be *pragmatically synonymous* for this domain. Arguably, a good model of interactional dynamics should be able to capture this synonymy.

In this paper, we show, using an implemented system (Anon; Anon) that given Dynamic Syntax and Type Theory with Records (DS-TTR) (Kempson et al., 2001; Eshghi et al., 2012; Eshghi et al., 2015) as a low-level, incremental model of interactional and contextual dynamics, one can see dialogue acts, together with their associated local dialogue structures and procedural conventions as emergent and learned from interaction; and thus that fully incremental dialogue systems can be bootstrapped from raw, unannotated example suc-

Interactional	USR: I would like an LG lap- err, phone SYS: okay.	USR: I would like a phone by LG. SYS: sorry a what? USR: a phone by LG. SYS: okay.	SYS: what would you like? USR: an LG phone SYS: okay.
	SYS: what would you like? USR: a phone SYS: by which brand? USR: LG SYS: okay	SYS: you'd like a ...? USR: a phone SYS: by what brand? USR: LG. SYS: okay	SYS: so would you like a computer? USR: no, a phone. SYS: okay. by which brand? USR: LG. SYS: okay.
Lexical	USR: I want an LG phone. SYS: okay.	USR: Can I have an LG phone? SYS: Sure.	SYS: What do you want to buy? USR: a phone SYS: by which make? USR: LG SYS: Okay.

Figure 1: Some Interactional and Lexical Variations in a Shopping Domain

cessful dialogues within a particular domain.

The model we present below combines DS-TTR with Reinforcement Learning for incremental word selection, where dialogue management and language generation are treated as one and the same decision/optimisation problem, and where *the corresponding Markov Decision Process is automatically constructed*. We show using the Facebook AI bAbI dataset, that the model can process 70% of the whole dataset, when trained on a very small fraction, 0.12% (5 dialogues), of the dataset. This remarkable generalisation power results from the linguistic knowledge and insight present within the grammar as a model of interactional dynamics and highlights the limitations of state-of-the-art, bottom-up machine learning techniques that do not exploit such models.

1.1 Dimensions of Pragmatic Synonymy

There are two important dimensions along which dialogues can vary, but nevertheless, lead to very similar final contexts: interactional, and lexical. Interactional synonymy is analogous to syntactic synonymy - when two distinct sentences are parsed to identical logical forms - except that it occurs not only at the level of a single sentence, but at the dialogue or discourse level - Fig. 1 shows examples. Importantly as we shall show, this type of synonymy can be captured by grammars/models of dialogue context.

Lexical synonymy relations, on the other hand, hold among utterances, or dialogues, when different words (or sequences of words) express meanings that are sufficiently similar in a particular domain or activity - see Fig 1. Unlike syntactic/interactional synonymy relations, lexical ones can often break down when one moves to another domain: lexical synonymy relations are domain specific. Here we do not focus on these, but merely note that lexical synonymy relations can be captured using Distributional Methods (see e.g. Lewis & Steedman (2013)), or methods akin to Eshghi & Lemon (2014) by grounding domain-general semantics into the non-linguistic actions within a domain.

2 Dynamic Syntax (DS) and Type Theory with Records (TTR)

Dynamic Syntax (DS) a is a word-by-word incremental semantic parser/generator, based around the Dynamic Syntax (DS) grammar framework (Cann et al., 2005) especially suited to the fragmentary and highly contextual nature of dialogue. In DS, words are conditional actions - semantic updates; and dialogue is modelled as the interactive and incremental construction of contextual and semantic representations (Eshghi et al., 2015) - see Fig. 2. The contextual representations afforded by DS are of the fine-grained semantic content that is jointly negotiated/agreed upon by the interlocutors, as a result of processing questions and answers, clarification requests, acceptances, self-/other-corrections etc. The upshot of this is that using DS, we can not only track the semantic content of some current turn as it is being constructed (parsed or generated) word by word, but also the context of the conversation as whole, with the latter also encoding the grounded/agreed content of the conversation (see e.g. Fig. 2, and see Eshghi et al. (2015); Purver et al. (2010) for details of the model). Crucially for our model below, the inherent incrementality of DS together with the word-level, as well as cross-turn, parsing constraints it provides, enables the word-by-word exploration of the space of grammatical dialogues, and the semantic and contextual representations

that result from them.

These representations are Record Types (RT, see Fig. 2) of Type Theory with Records (TTR, (Cooper, 2005)), useful for incremental specification of utterance content, underspecification, as well as richer representations of the dialogue context (Purver et al., 2010; Purver et al., 2011; Eshghi et al., 2012). For reasons of lack of space, we only note that the TTR calculus provides, in addition to other operations, the *subtype checking operation*, $\sqsubseteq$, among Record Types (RT), and that of the Maximally specific Common Supertype (MCS) of two RTs, which both turn out to be crucial for the automatic construction of our MDP model, and feature checking (for more detail on the DS-TTR Hybrid, see (Eshghi et al., 2012; Hough and Purver, 2014)).

3 The overall BABBLE method

We start with two resources: a) a DS-TTR parser *DS* (either learned from data (Eshghi et al., 2013), or constructed by hand), for incremental language processing, but also, more generally, for tracking the context of the dialogue using Eshghi et al.'s model of feedback (Eshghi et al., 2015; Eshghi, 2015); b) a set D of transcribed successful dialogues in the target domain.

Overall, we will demonstrate the following steps (see (Kalatzis et al., 2016) for more details):

1. Automatically induce the MDP state space, S, and the dialogue goal, G_D, from D;
2. Automatically define the state encoding function $F : C \rightarrow S$; where $s \in S$ is a (binary) state vector, designed to extract from the current context of the dialogue, the semantic features observed in the example dialogues D; and $c \in C$ is a DS context, viz. a pair of TTR Record Types: $\langle c_p, c_g \rangle$, where c_p is the content of the current, *PENDING* clause as it is being constructed, but not necessarily fully grounded yet; and c_g is the content already jointly built and *GROUNDED* by the interlocutors (loosely following the DGB model of (Ginzburg, 2012)).
3. Define the MDP action set as the *DS* lexicon L (i.e. actions are words);
4. Define the reward function R as reaching G_D, while minimising dialogue length.

We then solve the generated MDP using Reinforcement Learning, with a standard Q-learning method, implemented using BURLAP (MacGlashan, 2015): train a policy $\pi : S \rightarrow L$, where L is the DS Lexicon, and S the state space induced using F. The system is trained in interaction with a (semantic) simulated user, also automatically built from the dialogue data (see (Kalatzis et al., 2016) for details).

The state encoding function, F As shown in figure 2 the MDP state is a binary vector of size $2 \times |\Phi|$, i.e. twice the number of the RT features. The first half of the state vector contains the grounded features (i.e. agreed by the participants) ϕ_i, while the second half contains the current semantics being incrementally built in the current dialogue utterance. Formally:
$s = \langle F_1(c_p), \ldots, F_m(c_p), F_1(c_g), \ldots, F_m(c_g) \rangle$;
where $F_i(c) = 1$ if $c \sqsubseteq \phi_i$, and 0 otherwise. (Recall that $\sqsubseteq$ is the RT subtype relation).

4 Dialogue Turn Prediction

We have so far induced two prototype dialogue systems, one in an 'electronic shopping' domain (as exemplified by the dialogues in Fig. 1) and another in a 'restaurant-search' domain showing that incremental dialogue systems can be automatically created from small amounts of dialogue transcripts - in this case both systems were induced from a single successful example dialogue.

In this paper however, our focus is not on building dialogue systems per se, but on studying and empirically quantifying the interactional and structural generalisation power of the DS-TTR grammar, and that of symbolic, grammar-based approaches to language processing more generally.

We therefore here set ourselves the task of predicting the system's turn in a particular dataset (see below) given the dialogue so far up to that turn. Our method for doing this is as follows:

Given a particular set of example successful dialogues, D, in a domain, the rules for predicting a particular interlocutor's turns - in this case, the system's turns - are *automatically* extracted from D using only the DS-TTR parser, and F, the state-encoding function described above: the dialogues in D are parsed and encoded using F incrementally. All the states immediately prior to a system's turn, $s_i = F(c)$ – where c is a DS context – are recorded, and mapped to what the system ends up saying in those (encoded) contexts - for more than one training dialogue there may be more than one

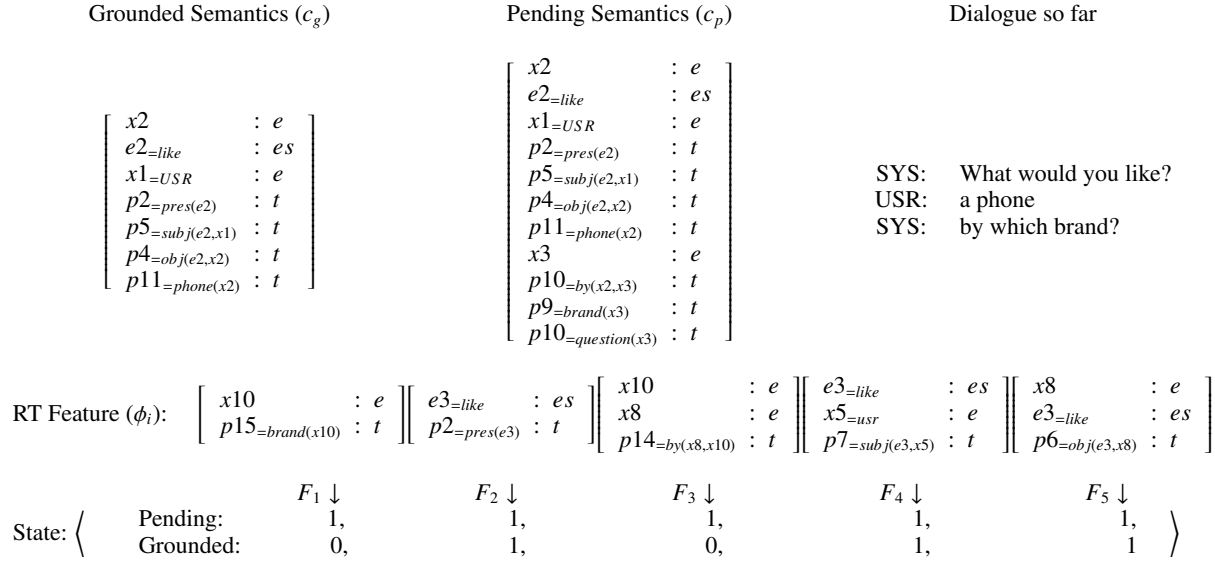


Figure 2: Semantics to MDP state encoding with RT features

candidate (in the same context/state). The rules thus extracted will be of the form:
$s_{trig} \rightarrow \{u_1, \ldots, u_n\}$, where u_i are user turns.

We also observe here that the above method respects the turn ordering encountered in the data, or more generally the order in which semantic increments are added to context. This is because states are composed not only of the semantic features of the current turn, but *also that of the history of the conversation*. And thus they capture *the contextual boundary* at which a user turn is being generated or a system turn monitored (e.g. in the bAbI 'restaurant-search' domain, a state might capture the fact that the user has already provided the cuisine type and the location of the restaurant).

5 Evaluation: measuring the generalisation power of the grammar

To measure the generalisation power of the DS-TTR dialogue model empirically, we use the above prediction method, rather than the full Reinforcement Learning method described earlier. To enable future comparison to other methods, the Facebook AI, bAbI dataset was chosen for this purpose - though we do not compare our results with others in this paper.

5.1 The bAbI data-set

This is a set of 4000, goal-oriented dialogues between two interlocutors in the domain of restaurant search. Here we tackle Task 1 only where in each dialogue the system asks the user about their preferences for the properties of a restaurant (slots, four of them in total), and each dialogue results in an API call which contains the values of each slot obtained. Other than the explicit API call notation, there are no annotations in the data whatsoever.

5.2 Experiment

As noted, rather than applying the full Babble method as described, we only apply the method described in section 4. We then use this to predict a system response, by parsing and encoding a test dialogue up to the point immediately prior to the system turn. This results in a triggering state, s_{trig}, which is then used as the key to look up the system's response from the rules constructed as per section 4. The returned response is then parsed word-by-word as normal, and this same process continues for the rest of the dialogue. This method uses the full machinery of DS-TTR & our state-encoding method - the DyLan model - and will thus reflect the generalisation properties that we are interested in.

5.3 Number of interactional variations captured

Here we establish, as an example of the power of the method implemented, a lower-bound on the number of dialogue variants that can be processed based on training from *only 1 example dialogue*. Consider the training dialogue (which has only 2 'slots' and 4 turns) below:

SYS: What would you like?
USR: a phone
SYS: by which brand?
USR: by Apple

Parsing this dialogue establishes (as described

above) a dialogue context that is required for success. The DS grammar is able to parse and generate many variants of the above turns, which lead to the same dialogue contexts being created, and thus also result in successful dialogues. To quantify this, we count the number of interactional variants on the above dialogue which can be parsed/generated by DS, and are thus automatically supported after training the system on this dialogue. Note that we do not take into account possible syntactic and lexical variations here, which would again lead to a large number of variants that the system can handle.

The *DS* grammar can parse several variants of the first turn, including overanswering ("I want an Apple laptop"), self-repair ("I want an Apple laptop, err, no, an LG laptop"), and ellipsis ("a laptop"), whose combinatorics give rise to 16 different ways the user can respond (not counting lexical and syntactic variations). These variations can also happen in the second user turn. If we consider the user turns alone, there are at least 256 variants on the above dialogue which we demonstrate that the trained system can handle. If we also consider similar variations in the two system turns (ellipsis, questions vs. statement, utterance completions, continuation, etc), then we arrive at a lower bound for the number of variations on the training dialogue of 8,192.

This remarkable generative power is due to the generalisation power of the DS grammar, combined with the system's DM/NLG policy which is created by searching through the space of possible (successful) dialogue variants.

6 Conclusion and ongoing work

We show how incremental dialogue systems can be automatically bootstrapped from small amounts of successful seed dialogues in a domain, combining Dynamic Syntax and Type Theory with Records with Reinforcement Learning. The method allows Dialogue Act conventions to emerge and be learned in (simulated) interaction, rather than be specified or annotated in advance. This method allows rapid domain transfer – simply collect some example (successful) dialogues in a 'slot-filling' domain, and retrain. Furthermore, as we have argued, it supports learning from very small datasets because of the generalisation power inherent in the DS-TTR grammar model. At present this is fully automated, and only requires checking that the DS lexicon covers the input data. We are currently applying this method to the problem of learning (visual) word meanings (groundings) from interaction.

References

Ronnie Cann, Ruth Kempson, and Lutz Marten. 2005. *The Dynamics of Language*. Elsevier, Oxford.

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation*, 15(2):99–112.

Arash Eshghi and Oliver Lemon. 2014. How domain-general can we be? Learning incremental dialogue systems without dialogue acts. In *Proceedings of Semdial 2014 (DialWatt)*.

Arash Eshghi, Julian Hough, Matthew Purver, Ruth Kempson, and Eleni Gregoromichelaki. 2012. Conversational interactions: Capturing dialogue dynamics. In S. Larsson and L. Borin, editors, *From Quantification to Conversation: Festschrift for Robin Cooper on the occasion of his 65th birthday*, volume 19 of *Tributes*, pages 325–349. College Publications, London.

Arash Eshghi, Julian Hough, and Matthew Purver. 2013. Incremental grammar induction from child-directed dialogue utterances. In *Proceedings of the 4th Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 94–103, Sofia, Bulgaria, August. Association for Computational Linguistics.

A. Eshghi, C. Howes, E. Gregoromichelaki, J. Hough, and M. Purver. 2015. Feedback in conversation as incremental semantic update. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*, London, UK. Association for Computational Linguisitics.

Arash Eshghi. 2015. DS-TTR: An incremental, semantic, contextual parser for dialogue. In *Proceedings of Semdial 2015 (goDial), the 19th workshop on the semantics and pragmatics of dialogue*.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press.

Patrick G. T. Healey. 2008. Interactive misalignment: The role of repair in the development of group sublanguages. In R. Cooper and R. Kempson, editors, *Language in Flux*. College Publications.

Julian Hough and Matthew Purver. 2014. Probabilistic type theory for incremental dialogue processing. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 80–88, Gothenburg, Sweden, April. Association for Computational Linguistics.

Dimitrios Kalatzis, Arash Eshghi, and Oliver Lemon. 2016. Bootstrapping incremental dialogue systems: using linguistic knowledge to learn from minimal data. In *Proceedings of the NIPS 2016 workshop on Learning Methods for Dialogue*, Barcelona.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell.

Ruth Kempson, Ronnie Cann, Arash Eshghi, Eleni Gregoromichelaki, and Matthew Purver. 2015. Ellipsis. In Shalom Lappin and Chris Fox, editors, *The Handbook of Contemporary Semantics*. Wiley-Blackwell.

Ruth Kempson, Ronnie Cann, Eleni Gregoromichelaki, and Stergios Chatzikiriakidis. 2016. Language as mechanisms for interaction. *Theoretical Linguistics*, 42(3-4):203–275.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

James MacGlashan. 2015. Burlap: Brown-umbc reinforcement learning and planning. In *http://burlap.cs.brown.edu/*.

G. Mills and E. Gregoromichelaki. 2010. Establishing coherence in dialogue: sequentiality, intentions and negotiation. In *Proceedings of SemDial (PozDial)*.

Gregory Mills. 2011. The emergence of procedural conventions in dialogue. In *IProceedings of the 33rd Annual Conference of the Cognitive Science Society*.

Matthew Purver, Eleni Gregoromichelaki, Wilfried Meyer-Viol, and Ronnie Cann. 2010. Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In P. Łupkowski and M. Purver, editors, *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*, pages 43–50, Poznań, June. Polish Society for Cognitive Science.

Matthew Purver, Arash Eshghi, and Julian Hough. 2011. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics*, pages 365–369, Oxford, UK, January.

Towards an annotation framework for incremental scope specification update

Asad Sayeed
Center for Linguistic Theory and Studies in Probability
University of Gothenburg
Box 100, 403 50 Gothenburg, Sweden
asad.sayeeed@gu.se

Abstract

This position paper outlines a framework for accommodating formal models of incremental scope processing in probabilistic terms through corpus annotation. Theories of scope ambiguity resolution encompass a number of overlapping and potentially conflicting strategies, including underspecification, quantifier raising, and employment of world-knowledge. Considering the conflicting evidence for the specific roles that these strategies play and the relationships between them, it may be possible to mediate their relationships through a probabilistic model that includes weights for each factor at each time step. This, however, requires the development of corpus resources. Here we propose the representation of scope specification operations as "decision tags" in future scope corpus annotation efforts.

1 Introduction

1.1 Motivation

The goal of this paper is to explore the possibility of a computational account of incremental scope ambuigity recognition and resolution via corpus annotation. The ability to handle ambiguity and multiple prediction is a core emphasis of present day natural language processing. Insofar as this intersects with psycholinguistic modeling, it most often involves using probabilistic reasoning to represent parsing "decisions" at the phonetic/orthographic, semantic, and syntactic levels. Depending on how theory-rich the psycholinguistic model is, one major question is how to represent probability inside a formalism and theory that constrains prediction. When applied to dialog systems, the formalism includes constraints for interaction with some relevant domain of knowledge.

Ambiguity, its identification, and its resolution cut across linguistic levels of representation. But the manner in which ambiguity appears is qualitatively different depending on the level of representation and the family of phenomena. Prepositional phrase attachment ambiguities operate on representations of surface syntax; resolution is achieved by settling on a particular syntactic connection. The characterization of scope ambiguities from linguistic theory tend to assume that the syntax is already decided; the ambiguity resides in connections drawn outside of syntactic structure and is resolved through selection of multiple semantic relations licensed by the same parse. The representation of scope ambiguity resolution therefore requires adaptation of existing techniques to scope-specific structure. From the perspective of strict incrementality, only a small amount of work has been done in this domain.

Recent advances in machine learning techniques, such as the trend towards deep learning and neural networks, attempt to acquire implicit representations of structure with increasingly minimal structure directly encoded in the input data, such as character-based approaches (Kim et al., 2016). Interpreting *post hoc* what structure is actually learned from the model weights is an open research question. Nevertheless, when the goal is to model a particular set of linguistic phenomena, some amount of phenomenon-specific structure may be required in the evaluation and training data. If the goal is to let the learning system acquire as much structure as it can, then the challenge is to find a sufficiently rich but theory-neutral data annotation scheme.

The phenomenon of ambiguous scope is interesting from both psycholinguistic and a natural language processing perspectives. Linguis-

tic theory has identified interpretive constraints over scope ambiguity that stem from either from syntactic structure or logical rules, depending on the perspective of the theorist (Ruys and Winter, 2011). These constraints are not readily visible in the surface string, but their operation may shed light on the processing difficulty phenomena of interest to psycholinguistics.

From an NLP perspective, the world is moving in the direction of increasingly domain-general conversational systems that will have to be able to handle the ambiguities that humans handle in order to reduce the human burden of interacting with these systems. Koller and Thater (2010) showed with a German newswire corpus that vast proliferations of possible scope readings can be extracted from sentences in isolation from pragmatic context.

In the remainder of this paper, we will explore the underlying modeling challenge of incremental scope ambiguity resolution, mention some psycholinguistic and formal approaches to representing it, and finally describe a corpus annotation scheme that represents scope ambiguity resolution as a sequence of interpretive "decisions".

1.2 Scope and processing

Consider the following sentences:

(1) a. Every child climbed a tree.
    b. Every jeweller appraised a diamond.

These sentences contain the "textbook" universal-existential quantifier scope ambiguity. (1-a) yields either the *linear* scope interpretation, in which for each child, there is a tree that that child climbed, or the *inverse* scope interpretation, in which there is a single tree that all the children climbed. (1-b) yields the same possibilities of interpretation. However, in (1-a), the pragmatics of trees and children suggests that the trees were distributed among the children, rather than there being a single tree. As diamonds are rare, (1-b) may support the inverse interpretation more easily.

Precisely when and how the processor makes decisions of quantifier scope interpretation has not been closely studied from a computational perspective to this point. Even from a psycholinguistic perspective, a focus on the time course of scope interpretation update has only a limited literature (section 2.1). But scope interpretation update requires many of the same elements that other forms of long-distance processing do (e.g., attachment ambiguity resolution) such as recall of prior constituents.

Now consider the presence of another NP after the subject.

(2) a. Every child a teacher picked climbed a tree.
    b. Every jeweller a customer selected appraised a diamond.

There are two possible effects, in terms of incremental processing, of inserting an additional intervener. One possible effect is that language users have more difficulty computing the pragmatics of the child-tree and diamond-jeweller relations because of the increased distance between them. Non-exclusively, another possible effect is that the intervening element itself changes the pragmatics of the listener's judgement, such as the reduction in the size of the set of children through selection by a teacher.

In theories such as Dependency Locality Theory (DLT; Gibson, 2000), the more intervening discourse referents, the higher the difficulty in processing the sentence. In this case, however, where language users may perceive a difference in pragmatics between "child-tree" and "jeweller-diamond", to what extent would an intervening element such as "teacher" or "customer" alter or interfere with these judgements? More specifically, how is this affected by the time-course of processing? How early are anticipated scopes selected? Answering these types of questions is part of the motivation for a computational approach to incremental scope ambiguity resolution.

1.3 Scope decisions

Traditionally, theoretical lingusitics has dealt with the question of syntactic and semantic ambiguity in terms of identifying the *available* interpretations of a given structure and has had less of a focus on the *chosen* interpretation that the language user has. Even the garden-path sentences that are used in psycholinguistic experimentation tend to produce the strongest effects when the sentences are given out of context. In the incremental resolution of representational ambiguities, both of these types of interpretive events have to be modeled: the point at which the parser decides an ambiguity may occur, and the point at which the ambiguity is resolved.

This may take place before all constituents are

even available. "Every child", for example, may already bias the processor to linear or inverse scope interpretations of quantifiers that follow, well before the quantifier has arrived, simply due to pragmatic biases in the kinds of situations that one tends to find children.

Quantifier scope phenomena are typically represented atop a general-purpose syntactic and semantic formalism. We propose to represent the process of scope interpretation in parallel with syntactic and semantic structure-building, as a series of "scope decisions" that indicate the establishment of scope domains and the update of scopes within them. This approach permits, among other things, an agnostic approach to the question of underspecification, representing when the knowledge is available to decide on precedence, but leaving it up to the learning system or the formalism to choose the timing of the decision. It also permits incremental scope processing to be converted to a sequence-labelling task, allowing a wide variety of approaches to be applied, from parsing based on rich formalism to n-gram tagging models.

2 Background

2.1 Experimental work

Experimental work in incremental scope processing has largely existed in response to theoretical work on scope resolution that emphasized the availability of interpretations from the parse of the sentence. This theoretical work, including theories such as Quantifier Raising (QR; e.g., Koster-Moeller et al., 2007), holds that complex "algorithmic" processes operate on sentences parses to allow the establishment of multiple interpretations. The experimental question is, therefore, the extent to which these processes can be observed in human linguistic behaviour, such as through processing difficulty measures—or, otherwise, the extent to which scope decisions are made by the system based on pragmatic constraints and lexically-encoded world-knowledge.

Early work (Kurtzman and MacDonald, 1993) explored the possibility that scope may be principally specified by the linear order of quantifier appearance, with results that dismissed a direct linear order preference. More recent work (Paterson et al., 2008) explored the possibility of competition between surface order and pragmatic factors in deciding scope precedence. Dwivedi (2013) did a series of self-paced reading studies that suggest that the processor actually only computes scope-relevant structure shallowly to start with and relies on an underspecified representation; instead, the processor employs world-knowledge and only uses algorithmic processing as a last resort. Dotlačil and Brasoveanu (2015) used a three-quantifier experiment design with eye-tracking to show that quantifiers are already assigned partly-specified relations by linear order.

This evidence paints a mixed picture of what drives scope processing, with the likely answer being that a combination of all the aforementioned factors are involved. Learning how to combine these factors is an advantage of applying machine learning to this task.

2.2 Formal work

Most efforts in developing incrementality-friendly semantic representations do not focus directly on scope. However, Dynamic Syntax includes a underspecified approach to representing the detection of scope constraints during the syntactic parse (Gregoromichelaki, 2006). Minimal Recursion Semantics (MRS; Copestake et al., 2005) uses a neo-Davidsonian approach (Parsons, 1990) for a fully underspecified representation of quantifier scope. Sayeed (2016) adapts a combination of a neo-Davidsonian and a QR-based approach to the possibility (Dotlačil and Brasoveanu, 2015) that scope relations may not be fully underspecified.

2.3 Corpora

Annotation of scopes in corpora is uncommon, but it has recently been accomplished by Ander-Bois et al. (2012) by annotating the LSAT Logic Puzzles, which gives 497 observations of ambiguous quantifier scopes and their resolutions as "narrow" or "wide". This annotation did not focus on the incremental or processing aspects of quantifier scope resolution. A similar effort was accomplished by Higgins and Sadock (2003), who tagged 893 sentences from the Penn Treebank. Manshadi et al. (2011) tagged 2500 sentences from a text editor instruction manual that includes annotations for sentences with multiple interacting quantifier scope domains.

Where the application requires it, researchers have annotated corpora for other types of scope interactions at a larger scale and used them in machine-learning efforts. For example, the BioScope corpus (Vincze et al., 2008) contains de-

tailed annotations of negation and uncertainty in biomedical texts considered 20,000 sentences and found and annotated 10% of them for scope cues that had a potential effect on meaning. Councill et al. (2010) used product reviews and annotated 679 sentences with negation annotations, using these and the BioScope sentences to successfully train a negation scope detection system.

3 Annotation proposal

3.1 Scope operations as decision tags

Since corpora are lifeblood of probabilitic modeling, the remainder of this paper will focus on proposing a general framework for annotation of ambiguous scope that takes into account incrementality.

In order to do this we identify a set of relevant operations in a model of incremental scope processing. One or more of these operations is activated whenever a processing unit (either a word or a phrase, depending on the granularity of the incremental parsing model) enters the system. From an annotation perspective, the application of one of these operations is represented as a "decision tag".

Decision tags take one of three forms: Δ, $\Delta(\Gamma)$, or $\Delta(\Gamma, \Psi)$. Δ is the operation itself, one of:

- **Quantifier introduction** (T) – a quantified phrase enters the system, requiring a label for the quantifier and variable.
- **Relation creation** (R) – two scope operators enter into a (possibly underspecified) scope precedence relationship.
- **Specification** (S) – an underspecified relation is given a specified precedence is selected between two scope operators. Specification subsumes relation creation when they happen at the same time. Specification can also hold multiple times for the same set of operators
- **Null** (N) – the word or phrase is not involved in a scope relation.

Γ is one of:

- a quantifier along with an identifier for the variable being quantified, usually as a subscript. E.g., $\forall_1$. This applies to the trigger introduction (T) operation.
- two quantifiers being brought into a possible relation. E.g., $\forall_1 = \exists_2$ (underspecified relation). This applies to the relation creation (R) operation.
- the relation between two quantifiers being specified. E.g., $\forall_2 > \exists_2$ (specified relation, where $\forall_2$ precedes $\exists_2$ in the scope order). This applies to the specification (S) operation.

Ψ is a term that represents the factor that "justifies" the decision and applies to the establishment of the scope relation (R) or its specification (S). This is the most flexible one, and depends on the theoretical biases of the annotation exercise. We propose the following two to start, but these are non-exhaustive and can include, e.g., weighted combinations of them:

- Syntactic/structural (X) – the scope operation was applied because algorithmic or formal constraints require an interpretation to hold at that point.
- Pragmatic/knowledge-based (P) – the scope operation was applied because of information about the world or discourse context applied by the processor.

This form of annotation has the advantage of being flexible: perhaps we only need or can obtain the decisions being made, but not their contents.

3.2 Illustrating the process

These operations can be illustrated by the following sentence. Using words as our incremental units, we take the basic example sentence above and step through it word by word. At the beginning of the derivation, the scope database is empty.

(3) || Every child climbed a tree
||

When the first word enters the system, we get:

(4) a. Every || child climbed a tree
T($\forall_1$) ||

The system can commit to the existence of a participant in the scope relation. The actual noun does not activate the scope system, although it will provide information during later decision making: it will receive an N, which does not imply that it is irrelevant, since most machine learning systems will also use features of the string:

(5) a. Every child ‖ climbed a tree
T($\forall_1$) N ‖

The verb "climbed" is transitive, so it heralds the presence of another NP that could potentially enter into a scope relation with "every child". However, once again, this annotation is not exclusive of other data sources (parses, treebanks), and only augments that information with what is explicitly known about scope.

(6) a. Every child climbed ‖ a tree
T($\forall_1$) N N ‖

The appearance of the definite article *after* the transitive verb, however, is evidence that something has a scope relation with "every child", but there is not enough evidence to make a full decisions. This is annotated as relation creation (R), which subsumes T:

(7) a. Every child climbed a ‖ tree
T($\forall_1$) N N R($\forall_1 = \exists_2$,X) ‖

Finally, "tree" enters the system, giving enough information to specify the relation. In this case, pragmatics favours the linear scope, hence the P justification term:

(8) a. Every child climbed a
T($\forall_1$) N N R($\forall_1 = \exists_2$,X)
tree ‖
S($\forall_1 > \exists_2$,P) ‖

We now have a record for the most constrained way in which a parser can use given scope and lexical information to decide incrementally on the scopes in the given sentence.

Consider the intervening element introduced in (2-a). This may be annotated as:

(9) a. Every child a teacher picked
T($\forall_1$) N S($\forall_1 > \exists_2$,X) N N
climbed a tree
N R($\forall_1 = \exists_3$,X) S($\exists_3 > \forall_1$,P)

In other words, the annotator may decide that on syntactic grounds, the parser could favour a linear scope interpretation of "a teacher", and then because of the restriction placed on "every child", an inverse scope for "a tree" may be required.

Annotation may thus proceed on an experimental basis, where annotators are treated as subjects and their differences considered factors in an experimental model, or a corpus containing gold standard data may be required, wherein annotators must strive for high agreement. This approach to annotation is flexible enough to handle both requirements.

4 Conclusions and future work

Quantifier scope ambiguity resolution is a long-distance, within-sentence semantic phenomenon that offers opportunities for research into the process of incremental human sentence comprehension as well as challenges in building domain-general, increasingly realistic conversational systems. A corpus annotation format that specificallly focuses on incremental update is necessary in order to exploit recent techniques in statistical NLP that can be applied to this type of modeling task.

In this position paper, we proposed one such annotation approach after justifying the need for such with some background in the recent history of research into scope processing. We recognize that developing corpus resources is labour-intensive and sometimes expensive, so it is best to propose a scheme for discussion in the relevant research community and then use that feedback to move on to the next steps: collecting relevant texts, constructing annotation interfaces, training annotators, and so on. We also had the goal of developing an annotation scheme that had multiple levels of detail, so that even if lower levels of detail have data sparsity problems, we would still have rich annotation at the higher level of scope decision-making.

Insofar as previous efforts at scope annotation have been successful, we are optimistic that an incrementality-focused annotation process can be likewise successful. Annotation interface design is the key element in successfully developing this type of corpus. Another area of future work is connecting this annotation to general-purpose syntactic or semantic representations such as MRS or Dynamic Syntax. An important challenge in annotation development will be an assessment of the reliability of the annotation, considering the subtlety of the scope annotation task; a major part of this challenge is the choice of the right agreement measure and the text bounds over which agreement is to be assessed.

Acknowledgements

This research was funded in part by the German Research Foundation (DFG) as part of SFB 1102 "Information Density and Linguistic Encoding" as well as EXC 284 Cluster of Excellence "Multi-

modal Computing and Interaction" during the author's previous employment. The work reported in this paper was also supported by a grant from the Swedish Research Council for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

References

Scott AnderBois, Adrian Brasoveanu, and Robert Henderson. 2012. The pragmatics of quantifier scope: A corpus study. In *Proceedings of Sinn und Bedeutung*. volume 16, pages 15–28.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation* 3(2-3):281–332.

Isaac G Councill, Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*. Association for Computational Linguistics, pages 51–59.

Jakub Dotlačil and Adrian Brasoveanu. 2015. The manner and time course of updating quantifier scope representations in discourse. *Language, Cognition and Neuroscience* 30(3):305–323.

Veena D Dwivedi. 2013. Interpreting quantifier scope ambiguity: Evidence of heuristic first, algorithmic second processing. *PloS one* 8(11):e81461.

Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain* pages 95–126.

Eleni Gregoromichelaki. 2006. *Conditionals in Dynamic Syntax*. Ph.D. thesis, University of London.

Derrick Higgins and Jerrold M Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics* 29(1):73–96.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 2741–2749.

Alexander Koller and Stefan Thater. 2010. Computing weakest readings. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 30–39.

Jorie Koster-Moeller, Jason Varvoutis, and Martin Hackl. 2007. Processing evidence for quantifier raising: The case of antecedent contained deletion. In *Proceedings of SALT*.

Howard S Kurtzman and Maryellen C MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition* 48(3):243–279.

Mehdi Manshadi, James Allen, and Mary Swift. 2011. A corpus of scope-disambiguated english text. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 141–146.

T. Parsons. 1990. *Events in the semantics of English*. MIT Press, Cambridge, MA, USA.

Kevin B Paterson, Ruth Filik, and Simon P Liversedge. 2008. Competition during the processing of quantifier scope ambiguities: Evidence from eye movements during reading. *The Quarterly Journal of Experimental Psychology* 61(3):459–473.

EG Ruys and Yoad Winter. 2011. Quantifier scope in formal linguistics. In *Handbook of philosophical logic*, Springer, pages 159–225.

Asad Sayeed. 2016. Representing the effort in resolving ambiguous scope. In *Sinn und Bedeutung 20*. Tübingen, Germany.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC bioinformatics* 9(11).

Stretching the Meaning of Words: Insights for Context-Sensitive Lexical Semantic Models

Elisabetta Jezek
Università di Pavia
Dipartimento di Studi Umanistici
Pavia, Italy
jezek@unipv.it

Abstract

We offer an evidence-based linguistic analysis of the phenomenon of semantic coercion, i.e. the phenomenon through which the meaning of a word is stretched in context due to the meaning of the co-occurring selecting words. We focus on examples of predicate-argument combinations, in which the stretching of meaning is triggered by the predicate and targets the semantic type of its arguments. We claim that semantic coercion phenomena are both constrained and graded and we validate this assumption through a data-driven investigation based on both corpus evidence and human judgements. We contend that the evidence of the existence of constraints on coercion phenomena is an argument in favour of the view that there exists a context-independent lexical meaning distinct from conceptual knowledge and pragmatic inference. Overall, our study supports the view that logically driven and machine learning methods based on distribution and probability are both equally essential to model the interplay between semantics, pragmatics and cognition in natural language phenomena such as the context-sensitivity of word meaning. We focus on the Italian language but our findings and results are valid crosslinguistically.

1 Introduction

In this paper we offer an evidence-based linguistic analysis of the phenomenon of semantic coercion[1], i.e. the phenomenon through which the meaning of a word is stretched in context due to the meaning of the co-occurring selecting words. We focus on examples of predicate-argument composition, in which the stretching of meaning targets the semantic type of the argument. We claim that semantic coercion phenomena are both constrained and graded and we validate this assumption through a data-driven investigation based on both corpus evidence and human judgements. We contend that the evidence of the existence of constraints on coercion phenomena is an argument in favour of of the view that there exists a context-independent lexical meaning distinct from conceptual knowledge and pragmatic inference. Overall, our study supports the view that logically driven and machine learning methods based on distribution and probability are both indispensable to model the interplay between semantics, pragmatics and cognition in natural language phenomena such as the context-sensitivity of word meaning. We focus on the Italian language but our findings and results are valid crosslinguistically.

The structure of the paper is the following. After providing evidence of the variety of context-dependency on the interpretation of words (section 2), and offering a survey of the different views with respect to the interplay between lexicon and cognition (section 3), in section 4 and 5 we present the claims that coercion phenomena are both constrained and graded, and corroborate them with corpus evidence and with data of human judgments we collected for this purpose (section 6). We conclude in section 7 by making a plea for moderate minimalisms for lexical semantic models and by advocating that modelling the interplay between semantics, pragmatics and cognition in

[1] Coercion as a tool of theoretical analysis has been used in several areas of grammar and frameworks, starting from aspectual coercion in Moens and Steedman (1988). We refer here most directly to the work on semantic type coercion by Pustejovsky (2011).

natural language phenomena such as the context-sensitivity of word's meaning requires a hybrid architecture which combines logically driven and machine learning methods based on distribution and probability.

2 Evidence for context-sensitive lexical semantics

Words are able to take on a different meaning depending on the context in which they are used. The coexistence of many possible meanings for a word is traditionally referred to as polysemy, and it is conceived as a list of established senses stored in the lexical entry. This is also the standard way dictionaries are put together.

The checklist theory on meaning, however, has long proved inadequate to account for the range and types of contextual variations in interpretation that words display in actual use. Focusing on the class of nouns, and on local linguistic context, these variations include - and are not limited to - properties of objects denoted by words coming into the foreground as in (1a), where a specific part of the *car* is referenced by the predicate *screech* (the wheels), contextual coercions as in (1b), where an event (*flight*) is contextually interpreted as the artefact through which it is performed (*aircraft*), hidden events as in (1b), where *before* introduces a temporal requirement related to *dessert*, which is resolved by adding *eating* to the default interpretation, and inherent polysemy as in (1d), where two interpretations of the same word (*book*) are triggered in the same linguistic context by two different predicative expressions, *on the shelf* (pointing at the physical aspect of book) and *boring* (pointing at the informational content).

(1) a. The car screeched down the road.
b. The flight landed safely at about 9 a.m.
c. We took a break before dessert.
d. The book on the shelf is boring.

In particular, metonymy shifts of the kind in (1b), appear to be widespread and systematic across languages. Extensive corpus data collected for Italian within the T-PAS project[2] using the corpus pattern analysis methodology (Hanks, 2013) shows that argument coercions are spread unevenly among verb classes, i.e. certain verbs tend to be more coercive than others, , and that certain kinds of shifts are more widespread than others. A selection of shifts is summarized in Table 1:

Source Type	Target Type
Artifact	Event
Artifact	Human
Artifact	Location
Event	Location
Human	Artifact (Vehicle)

Table 1: Coercion Shifts

The data below provides corpus evidence for the shifts indicated in Table 1. First, the verb class of the target predicate is provided (for example, aspectual verbs); second, the typed predicate-argument structure (T-PAS) of the verb is specified, based on which the shift was detected. Finally, both an example of matching and an example of mismatching are given; in the examples, the relevant syntactic role is marked in italics (for example, the role of object in the case of *interrompere*). *Mismatch* is used as a neutral term that registers the lack of correspondence between the type specified in verb template (corresponding to a specific verb sense), and the semantic type of the noun in the instance. Formulae such as $Artifact$ as $Event$ mark that the source is interpreted as the target in the context of use.

(2) *Aspectual Verbs*
[Human]-subj *interrompe* [Event]-obj
Arriva Mirko e interrompe *la conversazione*.
'Mirko arrives and interrupts the conversation' (matching)
Luisa ha interrotto *la pillola*.
Luisa interrupted the pill' (mismatch, $Artifact$ as $Event$)

(3) *Communication Verbs*
[Human]-subj *annuncia* [Event]-obj
Lo speaker annuncia la partenza.
'The speaker announces the departure' (matching)
L'altoparlante annunciava l'arrivo del treno.
'The loudspeaker announced the arrival of the train' (mismatch, $Artifact$ as $Human$)

(4) *Directed motion verbs*
[Human]-subj *raggiunge* [Location]-obj
Abbiamo raggiunto *l'isola* alle 5.

[2]The T-PAS project builds a resource of corpus-derived predicate-argument structures for Italian, such as [Human]-subj *raggiunge* 'reaches' [Location]-obj) for linguistic analysis and NLP tasks (Jezek et al., 2014). The repository consists of 4241 T-PASs for a total of 1000 average polysemy verbs. The project includes the construction of an inventory of corpus-derived type mismatches in argument positions.

'We reached the island at 5' (matching)
Ho raggiunto *il semaforo* e ho svoltato a destra.
'I reached the traffic light and turned right' (mismatch, $Artifact$ as $Location$)

(5) *Directed motion verbs*
[Human]-sub arriva (Adv[Location])
Ormai col buio sono arrivata *a una radura*.
'It was already dark when I arrived at a clearing" (matching)
Gli invitati arrivano *alla cerimonia* in ritardo.
'The guests arrive late at the ceremony' (mismatch, $Event$ as $Location$)

(6) *Motion using a vehicle*
[Flying Vehicle]-subj *atterra* (Adv[Location])
Il nostro aereo atterra alle 21.
'Our plane lands at 9pm' (matching)
Il pilota è regolarmente atterrato senza problemi.
'The pilot landed regularly with no problems' (mismatch, $Human$ as $Artifact$)

In order to account for the pervasiveness and heterogeneity of contextual meaning variations such as those shown in (1)-(6), it is evident that a context-sensitive model of lexical semantics is needed, incorporating two main standpoints: first, semantic flexibility is a property of natural language; the meaning of each word is expected to vary from occurrence to occurrence as a function of the interaction with the other words it combines with, and of the situation of utterance (Recanati, 2012); second, context-sensitivity is not confined to words with functional roles (traditionally verbs and adjectives), but extends e.g. to nouns (see Pustejovsky's *qualia* theory).

But how does such a model look like? And what is the best model (formal, distributional, probabilistic) to predict the observed contextual variation in word meanings? Can a single model serve both linguistic and computational purposes? To what extent statistics about word context and exploitation of co-occurrence information (distributionally-represented knowledge) can serve as a proxy for semantic grounding – and how can it inform us about compositionality in language? We argue that a basic requirement of a context-sensitive lexical semantic model is, above all, a clear standpoint with respect to the interplay between the lexicon, cognition and pragmatic processes, which we address below.

3 Lexicon, cognition and pragmatic processes: an overview

By common consent, words denote classes of entities and are associated with conceptual categories, for example a *dog* denotes an $animal$, a *table* denotes an $artifact$, *bread* denotes a kind of $food$, a *park* denotes a $location$, *run* denotes a $process$, and so forth. A conceptual category may be analyzed as a set of salient attributes or properties (Baroni and Lenci, 2008), for example the concept dog has properties: breathes, barks, wags its tail, has fur, and so forth. But which properties of a concept are genuinely distinctive and enter into the *lexical make-up* of a word and which ones do not? There are deep controversies regarding what piece of information associated with a word should enter into its definition, and constitute what is called its lexical information. Traditionally, it is assumed that so-called encyclopedic/commonsense/real world knowledge should be excluded (Marconi, 1997).

The distinction between lexical information and real world knowledge is intuitive but difficult to draw. According to some scholars, it is not even necessary. Others believe it should be conceived as a continuum rather than a dichotomy. In our view, opinions differ because there is no consensus about what criteria must be satisfied for a piece of information to qualify as encyclopedic knowledge instead of linguistic meaning, or vice versa.

Those who make a distinction take different positions on the subject (Jezek, 2016). According to the *minimalist* position, nothing of what we know about, say, the entity called *dog* is part of the context-independent lexical information associated with the word *dog*, except for those features that are necessary to define it as a domestic animal (as opposed to a wild one) and allow us to distinguish it from other entities falling into the same category. According to the maximalist position, the opposite is instead true, that is, the lexical information associated with the word *dog* incorporates our knowledge that dogs can be aggressive (and therefore bite and attack), that they have an acute sense of smell, that they like to chase cats, and so on. This additional knowledge about dogs is what we know from our individual experience and perception. A radical position is that taken by

those who hold that the distinction between lexical information and encyclopedic knowledge is artificial or useless, and should be eliminated. According to this position, what words do is give us access to concepts, and all the properties that enter into the constitution of a concept can in principle be exploited in language through the use of words. The contexts in which words are used determine which property/ies of the concept is/are activated in the specific case. The lexicon is interpreted as the access node into the vast repository of information associated with conceptual categories. This position is dominant in cognitive semantics and pragmatics, such as Relevance Theory (Sperber and Wilson (1995) and Carston (2002)), where context-dependency is dealt with at the conceptual level. Finally, meaning eliminativism - the most radical version of contextualism discussed in (Recanati, 2004) - maintains that we don't need abstract schemes in the form of context-independent linguistic meaning as input to the composition process. This can proceed without the help of conventionalized context-independent word meanings. Meaning eliminativism gets rid of abstract meaning in favour of observed occasion of particular uses.

4 Stretching the meaning of words

We argue that the position according to which there is no distinction between lexical meaning and conceptual content and that the construction of interpretation is entirely a matter of context, is not tenable, and language models which follow this view are deemed to fail in accounting for semantic composition in natural language.[3] There are at least three different types of arguments for this claim.

First, if we allow that context does all the work required to obtain the assignment of explicit semantic values to word occurrences, the range of interpreted values assignable to a given lexical entry is in principle unlimited; in this perspective, there is nothing preventing speakers from uttering a word instead of another in their speech, which is obviously not the case.

As for argument two, there clearly is stability in the assignment of semantic values to lexical items across speakers. Moreover, language users converge in their judgments regarding conditions of applications of words. Both these aspects support the idea that lexical meanings have a robust psychological reality.

Third, there are constraints to the way we can stretch the meaning of words in the context of use; these constraints do not appear to be systematically predictable on the basis of conceptual knowledge, suggesting that constraints operate not only at the cognitive level but also at the lexical semantics level, and there exists a distinction between the two.

To clarify this claim, consider the the concepts expressed by the following words and the relations existing among them: *museo* 'museum', *quadro* 'painting' and *collezione* 'collection':[4] *museo* denotes both the LOCATION where paintings are stored, and the INSTITUTION which is in charge of exhibiting them; *quadro* is the prototypical OBJECT associated with the EXHIBIT event; specifically, the participant playing the role of Theme; finally, *collezione* refers to a GROUP of accumulated paintings, usually considered as a whole because of the way it was put together (either by the same owner, or according to a particular property). The concepts expressed by these three words are clearly related: specifically, there exists a containment relation between *quadro* and *museo*, a part_of relation between *quadro* and *collezione*, and a relation of participation between *quadro* and the relational concept expressed by EXHIBIT.

Consider now the linguistic contrasts in (7), where $\rightarrow$ signals a successful coercion:

(7) a. "Il museo apre alle alle 9.00".
'The museum opens at 9.00
b. $\rightarrow$"La collezione apre alle 9.00.
'The collection opens at 9.00.
b. *"I quadri aprono alle 9.00".
'The paintings open at 9.00'.

In (7a) the noun *museo* 'museum' is contextually interpreted as the institution; the co-occuring verb *aprire* selects for an underlying agent enabling an activity (making the service of exhibit available, in this case).[5] We retrieved 1627 of hits

[3] See Asher and Pustejovsky (2005) for previous literature backing up this claim, with focus on modelisation; on meaning eliminativism, Gasparri (2013).

[4] In the following, words are reported in italics, whereas concepts, following an established tradition in cognitive linguistics - cf. Carston (2002) - are in small caps. We restrict ourself to signalling the concepts and relations we deem important for the present discussion.

[5] We do not analyse this contextual interpretation of *museo* as a coercion, because we assume that the type associated

of such examples in the ItTenTen 2016 corpus, consisting in 4.9 billion words[6]. The number of hits retrieved from the corpus confirms that combinations associated with this interpretation of museum are not only attested but also conventional in the use of the language.

Also the noun *collezione* in (7b) is interpreted as an institution, although the number of corpus instances retrieved in this case are much lower.[7] The analysis we propose for this example is that the contextual interpretation of collection as INSTITUTION hosting exhibitions is a successful coercion triggered by the verb *aprire*.

Consider now (7c), in which the noun *quadro* does not appear to licence the interpretation of institution, that is, coercion to Institution is not successful. We retrieved no hits of such cases from the corpus[8]. While this interpretation cannot be ruled out exclusively based on absence in corpus evidence, it is certainly not the *norm* (Hanks, 2013), as is the case with *museum* and, to a lesser extent, with *collection*.

The analysis of the sets of examples discussed here provide us with several pieces of evidence. First, there exist constraints to the way we can stretch the meaning of words, which are not necessarily motivated cognitively: indeed, words which are cognitively related such as *collezione* and *quadro* (being collection a word that denotes both the act and the result of gathering together a number of painting) exhibit in the data different semantic behaviors in composition with respect to the extent to which their meaning can be stretched. One might argue that this is related to conventions in language; we content that this is precisely the argument in favour of distinguishing conceptual knowledge from the actual knowledge associated with lexical items of a specific language. Second, it is plausible to assume that different types of constraints are active in semantic composition, an issue we address in the following.

with the noun *museo* is a *complex type*, that is, a single type made up by different components (Asher and Pustejovsky, 2005), and that the underlying compositional operation in this case differs from ordinary coercions. This position can of course be a matter of debate.

[6]The hits are obtained by the following CQL query [lemma="museo"][]0,4[lemma="aprire"][tag="PRE.*"] run on the corpus through the Sketch Engine tool (Kilgarriff et al., 2014).

[7]446 hits retrieved from the ItTenTen 2016 corpus using the query [lemma="collezione"][]0,4[lemma="aprire"].

[8]0 hits in the EnTenTen corpus, consulted through the following query: [lemma="collezione"][]0,4[lemma="aprire"].

4.1 Types of constraints operating on word combinations

We claim that there are at least three types of constraints on word combinations, even if when violated they produce the same result, namely word combinations which are ruled out as odd or interpreted only if the meaning of one of the members is stretched contextually, as in the case of coercion.

Conceptual or ontological constraints. These constraints on word combinations derive from the inherent properties of the word's referents - and the way we conceptualize them - of which we are aware as a result of our experience of the world, experience that we tend not to contradict when we speak. It is on the basis of this type of constraint that expressions such as "Bananas chase democracy" are not interpretable, at least in the actual world we live in.[9] A combination of words that violates this type of constraints expresses a conceptual conflict, which cannot be resolved in any way, because it is inconsistent from an ontological point of view, that is, from the point of view of how the world is and how we perceive and conceptualize it. Conceptual/ontological restrictions determine which words are semantically compatible with each other and which ones are not, i.e. they are the basic level at which compatibility among words is established.

Lexical constraints. Lexical constraints are similar to ontological constraints - their violation results in conflicting word combinations - but their nature is different, because they are based on a lexical conflict rather than an ontological one. A lexical conflict concerns the way in which a certain concept is lexicalized in a language. There exist a number of examples in different languages that support this claim (Jezek (2016), 190). For our current purposes consider It. "Luca calzava una cravatta rossa" 'Luca "calzava" a red tie', a combination which is not acceptable in Italian (0 hits in It TenTen. 2016 corpus) because it violates the lexical constraints that links the verb *calzare*, hyponym of *indossare* 'wear', to a specific subtype of clothes accessories, namely *scarpe* 'shoes" and *guanti* 'gloves"; in other

[9]We set aside in this discussion speculations about possible words, which certainly constitute an interesting perspective on the phenomenon of stretching outlined here.

words, *calzare* can be said of shoes and gloves (as in "Luca calzava gli scarponi da sci" 'Luca was wearing his ski boots') but not of ties. The conflict that arises from the violation of a lexical constraint, as in the example above, can be easily resolved by substituting the "conflicting" verb for a hypernym/super-ordinate with a looser restriction. For example, in the case of *calzare*, with the hyperonym *indossare* ('to wear' used for clothing in general).

Constraints institutionalized by conventionality of use. This type of constraint can neither be directly ascribed to ontological constraints (therefore, it is not conceptual), nor to obvious relations between word meanings which tend to co-occur.[10] That is to say, word combinations on which these constraints apply are not predictable only on the basis of the conceptual category or of the meaning associated with the words involved, as the other two types above. Rather, they are restricted by a constraint rooted in language use, that is, in the tendency of languages to express a given content by means of preferential word pairs, although in several cases other combinations are in principle possible from a semantic perspective. These conventionalized or institutionalized word pairs are often referred to as "collocations" in the linguistic literature, or "multiwords", if lexical substitution and syntactic modification of the constituents is blocked (Sag et al., 2002); they are felt by speakers as a typical way of saying a certain thing, i.e. as combinations characterized by a certain degree of conventionality. An example is It. *avere paura*, lit. 'to have fear' (140.283 hits in the ItTenTen corpus)[11] vs. the less likely *avere tristezza* (45 hits in the same corpus) lit. 'to have sadness': the most appropriate combination in latter case would be *provare tristezza* 'to feel sadness' (289 hits).

5 Graded coercion phenomena

We claim that besides being constrained, coercion phenomena are graded, that is, they vary depending on how much the lexical content is exploited or enriched in the context of use.[12] Following Pustejovsky and Jezek (2008), gradability can be approached in terms of "span" of coercion mechanisms, i.e., what semantic shifts are possible (given a certain starting point); what can be coerced into what else; how easily this may occur etc. In predicate-argument composition this span can be 'measured' by comparing the type expectation of a given predicate with the list of argument types it occur with in the data. Consider the following example of the It. verb *ascoltare* 'listen' and assume that in its basic sense it selects for $sound$. Corpus data (from Pustejovsky and Jezek (2008)) show that *ascoltare* combines with a variety of argument fillers, only a small subpart of which are pure $sound$s (we restrict ourselves to a selection of these types):

(8) *ascoltare* 'listen' ($sound$)
Object
$sound$: *voce* 'voice', *rumore* 'noise', *eco* 'echo'
$sound$ and $information$: *musica* 'music', *canzone*, 'song', *jazz* 'jazz'; *concerto* 'concert'
$event$ (natural): *vento* 'wind', *onda* 'wave'
$event$ (involving sound production): *respiro* 'breathing', *battito* 'beat' , *pianto* 'cry'
$event$ and $information$ (speech act): *annuncio* 'announcement', *conversazione* 'conversation', *lezione* 'lecture'
$media\ artifact$: *radio* 'radio'
$music\ artifact$: CD, *disco* 'disk', album
$sound\ makers\ artifact$: *campana* 'bell', *sirena* 'siren'
$human$ (who writes music): Mozart, Bach
$human$ (emitting sound): *ragazzo* 'boy', *cliente* 'costumer'; *cantante* 'singer' $body\ parts$ (emitting sound): *corpo* 'body'

What is interesting is that all nouns in (8) which are neither sounds nor types of sounds are reinterpreted as such when composed with *ascoltare*: media artifacts (*radio*), music artifacts (*disc*), sound makers artifacts (*bell*), events involving sound production (*cry*), speech acts (*announcement*), animals (*bird*), humans (*singer, Mozart*),

[10] This type of constraint is not directly relevant in our present discussion, but we briefly introduce it for completeness.

[11] The corpus was queried through the Sketch Engine with the following CQL: [lemma="avere"][lemma="paura"].

[12] The idea that there exists different types of coercion operations has already been explored and formalized in the literature. In Pustejovsky's lexical semantic framework, for example (cf. Asher and Pustejovsky (2005) and Pustejovsky (2011), coercion is modelled as a two-layered mechanism: *coercion by exploitation*, in which part of the context available in the lexical item is exploited, and *coercion by introduction*, where the argument's type is wrapped with the type the predicate requires, enriching the information of the argument denotation. The notion of gradience introduced here is related to but not overlapping this distinction.

body parts (*body*) and so on. It is clear that although all the coercions in (8) entail recomputing, they do not all involve the same amount of computation; it appears necessary to assume that the exploitation of the semantic content of the arguments involves different degrees, and that coercion is gradable; some shifts are easier than others; it is easier to shift from a source which is conceptually/ontologically close to the target than from one which is far. Conversely, source-target shifts in which the distance is bigger are cognitively more complex and less frequent.

6 Human judgments on gradability and constraints

In order to assess whether gradience of coercion phenomena is reflected in judgments of native speakers,[13] we run an experiment to obtain human judgements on a total of 100 verb-object and subj-verb partially contextualised dyads, where 2 annotators were asked to rate how literal the interpretation of the highlighted noun in the given dyad was, within a span of 1 (literal) to 5 (shifted in context), and a tag for semantically not acceptable (odd). We proposed the task as follows: how literal is the interpretation of the highlighted word in the contexts below? Rate it from 1 (literal) to 5 (shifted in context) and use the last column if you think the example is semantically odd or non interpretable. Drawing from Shutova et al. (2010), we provided the annotators with the following additional guidelines: For each phrase, establish the meaning of the noun in the context of the phrase. Try to imagine a more "basic" meaning of the noun in other contexts. If you can establish a basic meaning distinct from the meaning of the noun in this context, it is likely to be used non literally. The dyads consist of corpus-derived examples of matching (42) and mismatching (58) taken from the study in section 2 and from the Italian section of the dataset of the SemEval 2010 Task 7 on argument coercion (Pustejovsky et al., 2010). Additional examples were constructed manually with the goal of testing acceptability. Verbs included in the data are coercive in at least one of their meanings. Table 1 presents the sample of the annotated data for the verb *annunciare* 'announce': the noun under focus is the one in subject position that is assumed to be typed as [[Human]] by the selecting verb.

13	lo speaker annuncia la partenza del treno
14	Una tromba annuncia un parlamentare
15	Questa tromba annuncia una sventura
16	i telegiornali annunciano la ricomparsa della madre
17	una telefonata annuncia la presenza di un ordigno
18	L' altoparlante annuncia ritardi a catena
19	La voce metallica dell'altoparlante ha annunciato che il regionale sarebbe arrivato in ritardo
20	il premier spagnolo annuncia il ritiro immediato delle truppe
21	Washington ha annunciato un programma di aiuti
22	Il governo coreano ha annunciato la vendita della Dae-woo
23	Il comunicato annuncia la nomina del consiglio di amministrazione
24	Una voce dalla radio annuncia lo sciopero generale
25	Gli organizzatori annunciano una sorpresa

Table 2: Sample of the annotated data

Interannotator Agreement. The two annotators marked the same judgements in 57 cases out of the 100 proposed (57% of observed agreement). The agreement (IAA) we computed using *Cohen's Kappa*[14] is 0.38.[15] Given the variability of the possibility, we believe that this value is fair. We also tried to collapse extreme categories, merging the two mostly literal categories (1-2) and the least literal (4-5). We obtained an observed agreement of 68% and a K equal to 0.39.

Qualitative analysis. Overall, the results show that the two annotators express a variety of graded judgments on coercions when allowed to do so by the task.

Both annotators make extensive use of tag 1 (52 tags for ANN 1 and 49 tags for ANN 2), and their agreement on this tag - identifying the most literal reading - is higher than on any other tag (41 agreement on tag 1; 6 on tag 2; 2 on tag 3; 4 on tag 4, 1 on tag 5; 3 on odd).

Importantly, agreements on tag 1 do not include conventionalized coercions such as the ones in (9)[16]:

(9) a. "aprire il *vino* rosso in anticipo"
'open the red wine in advance' (tag 3, 4)
b. "finire il *bicchiere* prima di andarsene"
'finish the glass before leaving' (tag 4, 4)
c. "divorare *Asterix*"
'devour Asterix' (tag 3, 5)

[13] We agree with Lau et al. (2016) that although the mere existence of gradient human judgments cannot be taken as conclusive evidence against a categorical classification of linguistic knowledge: nevertheless we regard them as indicators that certain aspects of linguistic knowledge might be better represented as a probabilistic rather than as a categorical system.

[14] As is well known, Cohen's kappa (K) takes into account the possibility of the agreement occurring by chance; in fact the formula subtracts the probability of agreement by chance from the observed agreement.
For details see (Artstein and Poesio, 2008).

[15] I am indebted to Anna Feltracco for this calculus.

[16] In the example, we specify the tags used by the two annotators

d. "*Freuds* e' in edicola"
'Freud is at the newsstand' (tag 5, 5)
e. "*Washington* ha annunciato un programma di aiuti"
'Washington announced an aid campaign' (tag 4,5)
f. "L'*altoparlante* annuncia ritardi a catena"
'The loudspeaker announces successive delays' (tag 5,5)

This suggest that despite their frequency, conventionalized coercions such as those in (9) are still recognized - albeit to a different degree - as non literal uses of the word by native speakers.

On the other hand, in the context of *raggiungere* 'reach', the artefact *semaforo* 'traffic light' is tagged as literal by both annotators, supporting the hypothesis that this is a very "light" form of coercion, if at all; the property of being localized is inherent to all physical objects and under "coercive" contexts such as directed motion verbs they are easily interpreted as locations (see example 4).

The largest variation between the two annotators is to be found in tags higher than 1. In this respect we noted that compared to ANN 2, ANN1 underused tag 3 (total of 5 annotations), perhaps under the suggestion that it is a "neutral" score and does not disambiguate clearly between literal and non literal.

Finally, there is agreement between the annotators on the oddness (tag: odd) of the following two expressions, validating the corpus-based analysis in section 4, according to which the It. noun *quadro* is not successfully used as coercion to Institution.

(10) a. "i quadri aprono alle 9.00".
'The paintings open at 9.00'
b. "visitare i quadri".
'visit the paintings'

Perhaps the most notable observation, when looking at the entire dataset, is that although the annotations span over all degrees, their total between 1 and 3 (151) is much higher than the total between 3 and 5 (55). In light of the ratio between selection and coercion in the dataset (42/58), this result suggest that coercion mechanisms is perhaps not perceived as highly non literal by speakers, whereas this could not be the case with metaphor uses; this insight is left for future investigation.

7 Conclusions: A plea for Moderate Minimalism

We have claimed that stretching phenomena in semantics are both graded and constrained, and supported this claim with corpus evidence and data about human judgements. We have argued that the presence of constraints on coercion phenomena constitutes linguistic evidence that points towards a rejection of meaning eliminativism and towards moderate minimalism in lexical semantics. Although context can stretch the meaning of words, some combinations are uninterpretable, and others are highly unlikely, because words do carry a meaning on their own, and the construction of interpretation is not entirely a matter of context. According to this view, a word is a collection of "pointers"[17] to "fragments" of conceptual knowledge; however, the way conceptual knowledge is packed into lexical items and available for exploitation in actual use presupposes the existence of a specific mental entity, lexical meaning, which acts as interface between concepts and words. Finally, in this paper we have used the dataset we created mainly for purposes of linguistic analysis. From the point of view of machine learning, however, this dataset represents a gold standard which can be employed to train and test models aiming at detecting and classifying stretching phenomena. Finally, the linguistic study we have presented supports the view that hybrid architectures merging logically driven and machine learning methods based on distribution and probability are necessary to model the interplay between semantics, pragmatics and cognition in natural language phenomena such as the context-sensitivity of word's meaning.

Acknowledgments

We thank the anonymous reviewers and the participants to the Conference on Logic and Machine Learning in Natural Language (LaML), CLASP, University of Gothenburg, June 12-14, 2017, for their useful comments.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

[17] I owe this formulation to Katrin Erk.

Nicholas Asher and James Pustejovsky. 2005. Word meaning and commonsense metaphysics. *Semantics Archive*.

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

Robin Carston. 2002. *Thoughts and Utterances: The Pragmatics of Explicit Communication*. Wiley-Blackwell.

Luca Gasparri. 2013. A look at meaning eliminativism. *Philosophy Study*, 3(11):1018.

Patrick Hanks. 2013. *Lexical analysis: Norms and exploitations*. The MIT Press.

Elisabetta Jezek, Bernardo Magnini, Anna Feltracco, Alessia Bianchini, and Octavian Popescu. 2014. T-pas: A resource of corpus-derived typed predicate-argument structures for linguistic analysis and semantic processing. In *Proceedings of LREC 2014*, pages 890–895.

Elisabetta Jezek. 2016. *The lexicon: An Introduction*. Oxford University Press.

Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlỳ, and Vít Suchomel. 2014. The sketch engine: ten years on. *Lexicography*, 1(1):7–36.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2016. Grammaticality, acceptability, and probability: a probabilistic view of linguistic knowledge. *Cognitive Science*, pages 1–40.

Diego Marconi. 1997. *Lexical competence*. MIT press.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational linguistics*, 14(2):15–28.

James Pustejovsky and Elisabetta Jezek. 2008. Semantic coercion in language: Beyond distributional analysis. *Italian Journal of Linguistics*, 20(1):175–208.

James Pustejovsky, Anna Rumshisky, Alex Plotnick, Elisabetta Jezek, Olga Batiukova, and Valeria Quochi. 2010. Semeval-2010 task 7: Argument selection and coercion. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 27–32.

James Pustejovsky. 2011. Coercion in a general theory of argument selection. *Linguistics*, 49(6):1401–1431.

François Recanati. 2004. *Literal meaning*. Cambridge University Press.

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

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. *Computational Linguistics and Intelligent Text Processing*, pages 189–206.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1002–1010. Association for Computational Linguistics.

Dan Sperber and Deirdre Wilson. 1995. *Relevance: Communication and Cognition*. Blackwell.

Experimental Results on Exploiting Predicate-Argument Structure for Verb Similarity in Distributional Semantics

Benjamin Blundell
Information Technology Services
Queen Mary University of London
b.blundell@qmul.ac.uk

Mehrnoosh Sadrzadeh
School of Electronic Engineering and Computer Science
Queen Mary University of London
m.sadrzadeh@qmul.ac.uk

Elisabetta Jezek
Department of Humanities
University of Pavia
jezek@unipv.it

Abstract

An insight from formal semantics is applied to distributional semantics by building verb vectors and tensors that do take into account argument information associated with verbs. Four different argument combination models are presented and used to augment the verb vectors in two different conjunctive and disjunctive ways. The resulting representations are evaluated on a verb similarity task in three different vector spaces. Three different subsets of the similarity dataset are identified and the performance of the models are analysed on them. The overall findings show that the argument-augmented models and in particular a conjunctive model based on point wise multiplication and the Kronecker tensor product performed better than the base line of verb-only vectors and the other operations.

1 Introduction

Standard distributional semantics focuses on the degree to which words co-occur in context rather than the syntactic structure in which they appear. Formal semantics takes an opposite view: meanings of words emerge in syntactic structure: adjectives and intransitive verbs are represented by e.g. unary predicates, transitive and ditransitive verbs by binary and ternary predicates, and so forth. As a result, in distributional semantics, where the representation of a word is its context vector, verbs with the same contexts but different argument structure will be come out similar. For example, 'stroke' and 'purr' both occur frequently in the context of 'pet', but whereas 'purr' is an intransitive verb and only has one argument, 'stroke' is transitive and has two arguments. This distinction does not show itself in this context: the two verbs are considered to be similar when they occur in the context of 'pet'. In formal semantics, on the other hand, verbs that have the same arguments will have identical representations An example is a formal semantics model wherein the meaning of the two verbs 'like' and 'sit beside' is the pair ('John', Mary'). This model cannot distinguish between these two verbs since they have the same arguments.

In this paper, we argue that exploiting the syntactic structure of the verb - particularly the information associated to its argument slots - when building a distributional representation for it, improves the much used distributional task of similarity. We implement this idea by *augmenting* the distributional representation of a verb with the distributional representations of its arguments with the goal of providing a better distributional representation for the verb itself. We present different models for combining the argument vectors and augmenting the vector and tensor of the verb with them. We experiment with different spaces with different vocabulary, dimension, and window sizes. We implement three different vector construction algorithms, CBOW, Skip-gram (both from the Word2Vec package but by using Google's TensorFlow package[1]), and the PPMI-normalised co-occurrence counts. We evaluate our hypothesis by applying it to the SimVerb-3500 verb similarity dataset of (Gerz et al., 2016). The results support our hypothesis; in the majority of models and parameters the augmented vectors do better than the vectors of the verbs alone and the vectors of the arguments alone in the similarity task. Moreover, the combination of the Kronecker tensor product of the arguments augmenting the Kronecker tensor of the verb provides best results both in the de-

[1] https://www.tensorflow.org/

velopment and in the test set. The best method of augmentation was a conjunctive operation: pointwise multiplication.

Our work sits within the emerging field of *compositional distributional semantics*, which aims to combine insights from distributional semantics with formal semantics models of meaning in order to build compositional vectors for meanings of phrases and sentences of natural language. Different unifying models have been proposed and experimented with tasks involving phrases and sentences, e.g. see (Erk and Padó, 2008; Baroni and Zamparelli, 2010; Coecke et al., 2010; Clark, 2015; Maillard et al., 2014; Krishnamurthy and Mitchell, 2014); for a recent take and more models see the articles of the recently edited volume by (Boleda and Aurelie, 2016).

Our work is different from existing work in the field in that it is the first time that the focus is not on building vectors for phrases and sentences but on representations of words themselves. For instance, experiments of previous work focused on improving tasks such as phrase/sentence disambiguation in the work done by (Mitchell and Lapata, 2010; Grefenstette and Sadrzadeh, 2011; Polajnar et al., 2014), phrase/sentence paraphrasing in (Mitchell and Lapata, 2010; Kartsaklis and Sadrzadeh, 2013), phrase reconstruction in (Baroni and Zamparelli, 2010), phrase/sentence entailment in (Lewis and Steedman, 2013), and parsing in (Krishnamurthy and Mitchell, 2014). Our focus is on building enriched representations for verbs and the experimentation is on a verb similarity task. We use the SimVerb-3500 dataset, which was designed to represent the complexity and diversity of verb meanings and to gain a better understanding of verb semantics in distributional models. Our thesis holds for any word with a functional role, e.g. adjectives and prepositions. Experimenting with similarity of these words can also be a future direction.

The idea that distributional representations of meanings of some words can be improved when their syntactic structure is used has been investigated in some of previous work and when building vectors for phrases and sentences containing them, e.g. in (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011) and more directly in (Erk and Padó, 2008). A further difference of our work with the former, i.e. the work of (Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Polajnar et al., 2014; Kartsaklis and Sadrzadeh, 2013), is that they only focus on combinations of arguments and do not combine these with the vector of the verb. Further, often only one model is used, e.g. (Baroni and Zamparelli, 2010) use linear regression on the phrase vectors of adjective-nouns and nouns to harvest matrices for adjectives, (Grefenstette and Sadrzadeh, 2011) only work with sums of Kronecker products of arguments to build matrices for transitive verbs and moreover they do not combine these with the vector of the verb. The work of (Erk and Padó, 2008) only combines vectors of arguments with that of the verb using a conjunctive multiplication model and does not investigate disjunctive operations. There are further theoretical differences, for example that following work carried out at the lexical semantics/syntactic interface in linguistics ((Pustejovsky, 1995), (Van Valin, 2005), (Levin and Rappaport Hovav, 2005)) we assume that argument structure is part of the meaning of the verb and not external to it, which is what most previous work seem to follow. In this paper, we mainly present experimental results and leave the theoretical expansion to a sequel paper.

2 Verb Similarity Task and Dataset

We evaluate our models on the 3500 verb similarity evaluation resource SimVerb-3500 of (Gerz et al., 2016). This is a dataset of 3,500 verb pairs with human rankings for the similarity of each pair, divided to 500-3000 development and test sets. It contains 827 verb types covering all the normed verb types of the University of South Florida Free Association Norms (USF), the largest database of free association for English. The verbs were paired with each other using the norming process of USF and VerbNet classes (Kipper et al., 2008); leading to similar (*respond/reply*) and non-similar (*run/seat*) pairs.[2]

In our experiment, we first computed vectors for each verb using a set of models, to be presented in Section 4; then we computed the cosine similarity between the vectors in each pair and calculated the degree of correlation between this the similarity so obtained and human rankings from the SimVerb-3500 using Spearman's ρ. Human annotations were obtained from 843 native English

[2]Besides synonymy, the resource includes human rankings for three additional types of verb relations: antonyms, hypernym/hyponyms, cohypomyms: we return to this in section 6.2.

speakers within the age range of 18-50. In order to restrict the annotators to this range, the special crowed sourcing platform of PA[3]) was used. This platform provides demographic information. For more details on annotators and the dataset in general, we refer the read to the original paper on the development of this dataset, that is to (Gerz et al., 2016).

3 Intransitive vs Transitive

Some of the verbs of the SimVerb-3500 dataset are transitive, some are intransitive, some are both. In order to assign a valence to each verb of the dataset, we work with the following hypothesis: verbs that occurred with an object more than 50% of the total number of times that they occurred in total, are treated as transitive and verbs that had an object less than that, were treated as intransitive. For example `hesitate` occurred with a total of 19,504, 0.033% with an object and 0.966% without one, and was treated as intransitive accordingly. Similarly, `refrain` occurred a total of 7,725, 0.044% with an object and 0.95% with none. Both were treated as intransitives. Examples of transitive cases were `attract` with 125,544 occurrences, 0.78% with an object and 0.21% with none. Another example is `lend` with a total occurrence of 3,1741, 0.76% with an object and 0.23% with none. Examples of borderline cases were `combine` and `dismiss` with a 51%-48% division between their object and no-object occurrences. These were mostly verbs that have both transitive and intransitive roles. For the purpose of this paper, we applied a strict above/under 50% threshold and each verb was provided with a transitive/intransitive tag. In further work, we plan to apply a more fine grained filter using linguistic resources to identify the type of the verb.

4 Vector and Tensor Combination Operations

For intransitive verbs, we consider one argument type per verb: subject. A verb occurs in a corpus in different phrases and sentences, acquiring many subjects. Each subject has a vector representation. We combine these vectors to obtain the matrix for the subject argument using two different sets of operations:

1. Disjunctive Operations: summation, point-wise maximum

2. Conjunctive Operations: point-wise multiplication, point-wise minimum, and Kronecker tensor product.

Recall that the operations minimum and maximum act point-wisely on their inputs: they take two vectors as input, take the minimum/maximum of each entry therein and return a vector as output. The Kronecker tensor product takes two vectors and returns a matrix whose elements are multiplications of elements of vectors. For a demonstrations, see below where these operations are defined on vectors in a two dimensional space:

$$\min(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix}) = \begin{pmatrix} \min(a,c) \\ \min(b,d) \end{pmatrix}$$

$$\max(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix}) = \begin{pmatrix} \max(a,c) \\ \max(b,d) \end{pmatrix}$$

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac & ad \\ bc & bd \end{pmatrix}$$

Since our numbers are probabilities, we are working in a sub-vector space of a vector space over the field of reals. This subspace is restricted to positive reals; therein disjunctive operations accumulate the information of the arguments, conjunctive ones sift through and take the common part of them. In this setting, disjunctive operations on vectors are analogous to taking union on sets and conjunctive operations are analogous to taking intersection. Hence, taking maximum acts similarly to summation and taking minimum to multiplication. For a demonstration, suppose we have two vectors, one element of one of which is a zero and for the other element we have $a \leq c$, recall $a, c, d > 0$ we have

$$\begin{pmatrix} a \\ 0 \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a+c \\ d \end{pmatrix}$$

$$\max(\begin{pmatrix} a \\ 0 \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix}) = \begin{pmatrix} c \\ d \end{pmatrix}$$

whereas

$$\begin{pmatrix} a \\ 0 \end{pmatrix} \odot \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ 0 \end{pmatrix}$$

[3] `https://prolific.ac/`

Label	Formula	Label	Formula
Arguments Only		**Verbs Augmented by Arguments**	
Sum	$\overrightarrow{Verb}^{+}_{tv} = (\sum_i^n \overrightarrow{Sbj}_i) + (\sum_i^n \overrightarrow{Obj}_i)$	Sum-Sum	$\overrightarrow{Verb}^{+}_{tv} + \overrightarrow{Verb}$
Kronecker	$\overrightarrow{Verb}^{\otimes}_{tv} := \sum_i^n \overrightarrow{Sbj}_i \otimes \overrightarrow{Obj}_i$	Sum-Mult	$\overrightarrow{Verb}^{+}_{tv} \odot \overrightarrow{Verb}$
		Kron-Sum	$\overrightarrow{Verb}^{\otimes}_{tv} + (\overrightarrow{Verb} \otimes \overrightarrow{Verb})$
		Kron-Mult	$\overrightarrow{Verb}^{\otimes}_{tv} \odot (\overrightarrow{Verb} \otimes \overrightarrow{Verb})$

Table 1: Subjects/ objects combination formulae

$$\min(\begin{pmatrix} a \\ 0 \end{pmatrix}, \begin{pmatrix} c \\ d \end{pmatrix}) = \begin{pmatrix} a \\ 0 \end{pmatrix}$$

In the first set of operations, the zero element disappears from the result, as we are accumulating positive information. The contrary is true for the second case where the zero survives as we are intersecting information. In the same line, a zero will survive the Kronecker product:

$$\begin{pmatrix} a \\ 0 \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac & ad \\ 0 & 0 \end{pmatrix}$$

In our experiment, we augment the vector of the verb with each of these combinations by summing and multiplying them. As illustrated above, the summation provides a disjunctive augmentation and the multiplication, a conjunctive one. As a technical side, in order to augment the Kronecker combination, we need to make the type of the verb match the type of the result of the Kronecker product of its argument, the former is a matrix, the latter a vector. We resolve the problem by working with the Kronecker product of the vector of the vector of the verb with itself, thus obtain a matrix encoding of the original vector of the verb. All the operations we performed are listed in Table 2. The number of the subject's occurrences of an intransitive verb are denoted by the variable k. For a transitive verb, the number of subjects and objects are the e number of times the verb has occurred in a transitive sentence, these are denoted by variable n.

Transitive verbs have at least two argument types per verb: subject and object. They occur in the corpus n many times, at each occurrence, we collect their subjects and objects and combine them by summing the subject and object vectors and summing their Kronecker products. The vector of the verb is augmented with these as for the intransitive case. The corresponding formulae are listed in Table 1.

Not every verb pair in the SimVerb-3500 dataset had a coherent type of verb; there were cases were intransitive verbs were paired with transitive ones. In such cases, we worked with the combination formulae that led to a representation in the same space. These were Sum and Kron for the argument only models, and Sum-Sum, Sum-Mult, Kron-Sum, and Kron-Mult for the augmented argument models.

5 Vector Space Model Parameters

We implemented three different spaces; with the skip-gram algorithm implemented in TensorFlow, with the original C version of the continuous bag of word algorithm implemented in Word2Vec, and with a count-based space. All spaces were trained on the parsed version of the UKWacky corpus[4], which is a concatenation of UKWac[5] and a 2009 dump of English Wikipedia, tagged with the TreeTagger and parsed with the MaltParser. In the case of Word2Vec, we worked with a 50k and a 500k vocabulary size, the vector embedding sizes were 128, 256, and 500, the window sizes were 3 and 5. The parameters of the skip-gram space where the same, except the maximum embedding size was 256, due to the large training times. Finally, in the count-based space, the window sizes were 3 and 5 and the dimension was the most frequently occurring 5k lemmas minus the top 100k; vectors were weighted with PPMI.

Amongst the UKWacky's tags are the subjects and objects of sentences. We used the tags for subjects and objects to detect the role of a word in a sentence. UKWacky also contains the parsed structure of the sentences, using this structure we could determine which words the subjects and objects were operating on.

Different machines were used for the implementation and evaluation, among these was a cluster of 8 Core Intel Xeon for the 500 development set and an 18 machine cluster of 16 Core Intel Xeon for the 3000 test set.

[4] http://wacky.sslmit.unibo.it/doku.php?id=corpora
[5] https://www.sketchengine.co.uk/ukwac-corpus/

6 Subsets of the Dataset

6.1 Number of Subjects and Objects

We probed the number of subjects and objects of the verbs to test whether this parameter affected the results. Some verbs had a large number of subjects/objects; more precisely, subjects for the case of intransitive verbs and subjects and objects for transitive verb. We wanted to know wether these verbs made better or worse contributions to our models. Our conjecture was that these verbs would provide inappropriate candidates for the argument-augmented models since their arguments are plentiful and in principle can range over all eligible nouns; thus the information provided by them is more resemblant of noise. We decided to verify this idea by removing these verbs from the dataset and test whether the results improved or not. Our findings showed that about 95% of the verbs have very few subjects/objects and very few of the verbs have a large number of subjects/objects. Based on this, verb pairs that had the top 5% and the low 5% of the total number of subjects/objects of all verb pairs were retrieved. The experiment was repeated separately for each case. The performance of the argument-augmented model on the omitted top 5% of verbs was improved.

Our intuition was backed with this experimentation: when the set of verbs with top 5% of subjects/objects where removed from the model, the amount of noise from the model was reduced and the augmentation worked better.

6.2 Synonyms vs Antonyms

We analysed the correlation between the relationship of the verbs in a pair and how well the argument augmented models worked. As referenced in section 1, each verb pair of the SimVerb-3500 dataset came labelled with one of $\{$antonym, synonym, hyper/hyponym, none$\}$. Our findings showed that the subset of the dataset with synonymy relationship performed best and dropping the antonyms from the dataset improved the results. The conclusion is similar to that of (Gerz et al., 2016): argument-augmented models work best for synonyms and become worse when antonyms are included.

6.3 Intransitive vs Transitive

We experimented with subsets of the dataset restricted to pairs where verbs are either both intransitive or both transitive. A preliminary analysis did not show a clear improvement and we could not find much regularity in the results; some argument combinations together with certain verb augmentations worked better for some implementations and we some did not. The argument-based results performed better than the verb-only ones in the majority of cases. For reasons of space we do not present these results here.

7 Results

The results are presented in tables 3 and 4. Table 3 reports the results on the 500 development set; Table 3 reports the result of the best model of the 3000 test set on the test set.

The results of the 500 development set are for the following space parameters: window size 5, vocabulary size 500k, dimension 500 for CBOW, 256 for skip-gram and 5k for the count-based model. The **Original Setting** is the full dataset before any classification was applied. The **Verbs with Top 5% of Subjects/Objects Removed** is the classification described in subsection 6.1, where the verb pairs that had the top 5% of the number of subjects/objets were removed from the dataset . The **Synonyms Only** and **All But Antonyms** are the result of the classifications outlined in subsection 6.2, where, respectively, only verb pairs that had a synonymous relationship were experimented on and where all the verb pairs but the ones that had the antonymous relationship were experimented on . As a baseline, we are comparing the results with a model with a non-augmented verb, that is, the vector of the verb without considering any of its arguments. The results of the verb-only model are presented in the first column of Table 3. To make it clear that this is our baseline, we have underlined the label of this column.

In the 500 development set, the highest degree of correlation was achieved by the count-based space and was 0.42. In all but the synonym-only subset of the dataset, the argument-only and the argument-augmented models did better than the verb-only model. The verb-only model did better in two out of three cases of the synonym-only subset. Here, the Kron-Sum and Kron-Mult operations were the best ways of augmenting the verb vectors with vectors of arguments; they performed best in 6 out of 12 cases. They were followed by Sum-Mult, which was the winner in 3 out of

Label	Formula	Label	Formula
Arguments Only		**Verbs Augmented by Arguments**	
Sum	$\overrightarrow{Verb}^{+}_{itv} := \sum_i^k \overrightarrow{Sbj}_i$	Sum-Sum	$\overrightarrow{Verb}^{+}_{itv} + \overrightarrow{Verb}$
Minimum	$\overrightarrow{Verb}^{\min}_{itv} := argmin(\overrightarrow{Sbj}_1, \cdots, \overrightarrow{Sbj}_k)$	Sum-Mult	$\overrightarrow{Verb}^{+}_{itv} \odot \overrightarrow{Verb}$
Maximum	$\overrightarrow{Verb}^{\max}_{itv} := argmax(\overrightarrow{Sbj}_1, \cdots, \overrightarrow{Sbj}_k)$	Min-Sum	$\overrightarrow{Verb}^{\min}_{itv} + \overrightarrow{Verb}$
Kronecker	$\overrightarrow{Verb}^{\otimes}_{itv} := \sum_i^k \overrightarrow{Sbj}_i \otimes \overrightarrow{Sbj}_i$	Min-Mult	$\overrightarrow{Verb}^{\min}_{itv} \odot \overrightarrow{Verb}$
		Max-Sum	$\overrightarrow{Verb}^{\max}_{itv} + \overrightarrow{Verb}$
		Max-Multiply	$\overrightarrow{Verb}^{\max}_{itv} \odot \overrightarrow{Verb}$
		Kron-Sum	$\overrightarrow{Verb}^{\otimes}_{itv} + (\overrightarrow{Verb} \otimes \overrightarrow{Verb})$
		Kron-Multiply	$\overrightarrow{Verb}^{\otimes}_{itv} \odot (\overrightarrow{Verb} \otimes \overrightarrow{Verb})$

Table 2: Subject combination Formulae

Model	Verb	Sum	Kronecker	Sum-Sum	Sum-Multiply	Kronecker-Sum	Kronecker-Multiply
The Original Setting							
Skip-gram	0.0094	0.0076	0.0348	0.0188	0.0224	**0.0465**	0.0187
CBOW	0.1497	**0.2013**	0.1374	0.2008	0.1684	0.1383	0.1767
Count	0.2382	0.1270	0.1457	0.1398	**0.2773**	0.1516	**0.2657**
Verbs with Top 5% of Subjects/Objects Removed							
Skip-gram	0.0141	0.0109	0.0501	0.0108	0.0188	**0.0503**	0.0223
CBOW	0.1757	0.1865	0.1003	0.1861	**0.1909**	0.1014	0.1864
Count	0.3613	0.2792	0.2772	0.2792	0.3880	0.2772	**0.4206**
Synonyms Only							
Skip-gram	**0.4119**	0.0784	0.1029	0.0736	0.3409	0.1029	0.3495
CBOW	0.1437	0.0491	0.0085	0.0491	0.3587	0.0085	**0.3832**
Count	**0.3613**	0.1270	0.1457	0.1398	0.2773	0.1516	0.2657
All But Antonyms							
Skip-gram	0.0143	0.0281	0.0669	0.0280	0.0146	**0.06721**	0.0110
CBOW	0.1693	**0.2059**	0.1338	0.2049	0.1706	0.1346	0.1790
Count	0.2309	0.1331	0.1484	0.1336	**0.2787**	0.1486	0.2650

Table 3: Degrees of correlation between human rankings and cosine distances on the **500 development set** of SimVerb-3500. Baseline is the verb-only model: the first column on the models row.

12 cases. Overall; the argument-augmented subset performed best in 6, the argument-only models in 2, and the verb-only models in 2 cases. Thus, the argument-based models performed best in 10 out of 12 cases; we take this as a support for our hypothesis.

As a second experiment, we picked the parameters of the best performing 500 development set and ran it on the 3000 test set. The best results were for the **count space**, where window size was 5, vocabulary size 500k, and dimension 5k. The best performance of the development set was the model where verbs with Top 5% of subjects/objects were removed. We repeated the experiment with the original as well as the other subsets of the dataset in the 3000 test set and obtained the same results: the best performance was provided by the argument-augmented model and in the subset where verbs with Top 5% of subjects/objects were removed. The best combination operation was again a tensor-based one: **Kronecker-Multiply**. The results are presented in Table 4. Here again and in consistency with the results of the 500 development set, multiplication proved to be the best augmentation method: the **Kronecker-Multiply** and **Sum-Multiply** operations achieved the top two best correlation scores and were thus our top two best performers.

Similar to the results of the original SimVerb-3500 paper (Gerz et al., 2016), the degrees of correlation are in general low. For the development set, their best set of results in a distributional space are obtained using a dependency based vector space, where the highest achieved degree of correlation was 0.401. Our best model (Kronecker-Multiply) reached a degree of correlation of 0.4206 and improves on that. The results reported in (Gerz et al., 2016) were then improved by using non-distributional linguistic resources. These models obtained a degree of correlation of 0.632. As these used non-distributional resources, it would not make sense to compare our models with these. In the test set, however, their best distributional model, which was again a dependancy

Model	Verb	Sum	Kronecker	Sum-Sum	Sum-Multiply	Kronecker-Sum	Kronecker-Multiply
The Original Setting							
Count	0.1490	0.1183	0.1014	0.1135	**0.1632**	0.1014	**0.1623**
Verbs with Top 5% of Subjects/Objects Removed							
Count	0.1558	0.1557	0.1414	0.1568	**0.2055**	0.1415	**0.2046**
Synonyms Only							
Count	0.1503	0.0245	0.0093	0.0251	**0.0265**	0.0093	**0.0236**
All But Antonyms							
Count	0.1533	0.1208	0.1074	0.1209	**0.1682**	0.1075	**0.1681**

Table 4: Degrees of correlation between human rankings and cosine distances on the **3000 test set** of SimVerb-3500 for the best model of the development set. Baseline is the verb-only model: the first column on the models row.

based vector space, obtain a degree of correlation of 0.351, which beats our best model. Nevertheless, their best count-based model only achieved a degree of correlation of 0.186, whereas the performance of our best count-based model was 0.2046 and thus improves on that.

8 Conclusions and Work in Progress

We conjectured that the distributional vector representations of words with functional roles, such as adjectives and verbs, will improve when their argument structure is taken into account. We focused on verbs and implemented this conjecture by augmenting the vectors of verbs with vectors of their arguments. We provided a variety of different models for combining the arguments with each other and for augmenting the vector of the verb with them. We worked with three different vector spaces using a parsed version of the UKWacky corpus: the Tensor Flow Skip-gram algorithm, the Word2Vec CBOW algorithm, and a count-based model, normalised with PPMI. We evaluated these models on the SimVerb-3500 verb similarity dataset. We analysed the dataset in three different ways: based on the number of subjects/objects of the verb pairs, on the relation between the verbs in a pair, and the transitive/intransitive types of the verbs. The former two provided improvements, the results of the latter did not lead to a clear conclusion.

Overall, the argument-augmented models provided the best results, confirming our hypothesis. The best combination operations were sum and multiplication of sums of Kronecker tensor products of arguments. The best overall degree of correlation was achieved by the multiplication of sums of Kronecker products in the count-based space when the verb pairs with the top 5% of the number of subjects/objects were removed. The results are the same in the 500 development and 3000 test set.

The correlations were overall quite low. One reason is the imprecision of the tags in the UKWaC. Our hypothesis relies on the recognition and retrieval of arguments of verbs, which in turn relies on the syntactic dependencies identified by the UKWaC tags. A preliminary analysis revealed a good number of mistakes. For instance, ”drivers” was identified as the object of the intransitive verb ”snooze” in the phrase ”snoozing drivers”; this was among the most common error patterns. Further, as the number of verb pairs increased, so did the number of general purpose verbs; on these our models did not performed well, due to the presence of noise in the subjects/objects. Bettering these faults by using a better parser to tag the UKWaC corpus and rerunning the models is a future direction. Working with tensors other than the Kronecker one and where the verb tensors are matrices that are directly built using data, as in previous work of (Grefenstette and Sadrzadeh, 2011) (Kartsaklis and Sadrzadeh, 2013) for verbs and of (Baroni and Zamparelli, 2010) for adjectives is another direction we believe is worth pursuing. Finally, studying the theoretical side of this work by exploring the syntactic and lexical structure of the verbs and its connections to its formal and distributional semantics, e.g. the internal versus external structure of the verb and the connections of this with the meaning of the verb, is our work in progress.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. pages 1183–1193.

Gemma Boleda and Herbelot Aurelie. 2016. Special

issue on formal distributional semantics. *Computational Linguistics* 42(4):619?635.

Stephen Clark. 2015. *Vector Space Models of Lexical Meaning*, Wiley-Blackwell, pages 493–522.

B. Coecke, M. Sadrzadeh, and S. Clark. 2010. Mathematical Foundations for a Compositional Distributional Model of Meaning. Lambek Festschrift. *Linguistic Analysis* 36:345–384.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 897–906.

Daniela Gerz, Ivan Vulic, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 2173–2182.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1394–1404.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNL)*. pages 1590–1601.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation* 42(1):21–40.

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Beth Levin and Malka Rappaport Hovav. 2005. *Argument realization*. Cambridge University Press.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192.

J. Maillard, S. Clark, and E. Grefenstette. 2014. A type-driven tensor-based semantics for CCG. In *Proceedings of the Type Theory and Natural Language Semantics Workshop, EACL 2014*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science* 34(8):1388–1439.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2014. Using sentence plausibility to learn the semantics of transitive verbs. *CoRR* abs/1411.7942. http://arxiv.org/abs/1411.7942.

James Pustejovsky. 1995. *Generative Lexicon*. The MIT Press.

Robert D Van Valin. 2005. *Exploring the syntax-semantics interface*. Cambridge University Press.