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.

$$\text{dog} \xleftarrow{\text{ARG1}} \text{chase} \xrightarrow{\text{ARG2}} \text{cat}$$
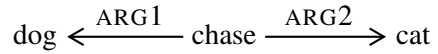
Figure 1: A simplified DMRS graph, illustrating the type of data observed during training. This graph would correspond to sentences like *The dog chased the cat*, or *Cats are chased by dogs*.
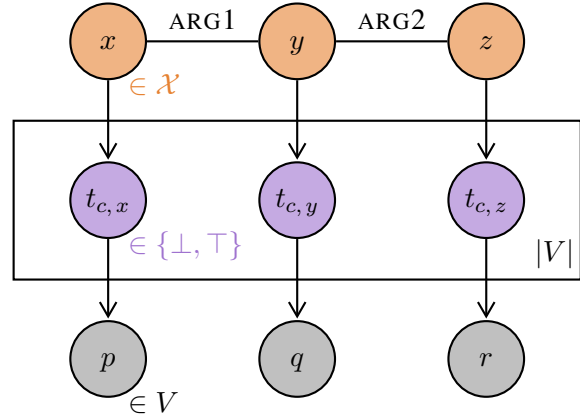

Figure 2: A probabilistic graphical model for Functional Distributional Semantics (E&C Fig. 3). Each node denotes a random variable, but only the bottom row is observed. The plate (middle row) denotes repeating variables across the vocabulary. **Top row:** latent pixies x, y, and z, lying in a semantic 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
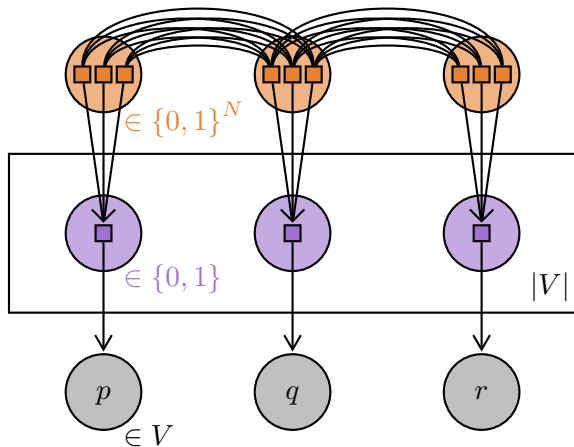
Figure 3: Implementation of the model in Fig. 2.
Top row: pixies are binary-valued vectors, forming a CaRBM. For each link, connections between the dimensions of the two pixies determine how likely they are to be active at the same time.
Middle row: each semantic function is a one-layer feedforward network, 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 $P(t_{b,x}|t_{a,x}) = 1$.

---

[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 $P(A) \neq 0$

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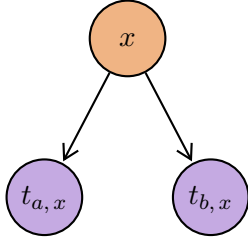
Figure 4: Logical inference for a single pixie x, and two predicates a and b. We have a joint distribution between x and the truth values $t_{a,x}$ and $t_{b,x}$, which lets us consider logical inferences in terms of conditional probabilities, such as $P(t_{b,x}|t_{a,x})$, the probability of b being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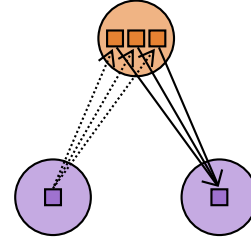

Figure 6: Variational inference for Fig. 4. Exactly calculating the distribution of x given $t_{a,x}$ is intractable, but we can use a mean-field approximation. The dotted lines indicate approximate inference, and the solid lines indicate inference from the mean-field pixie vector.
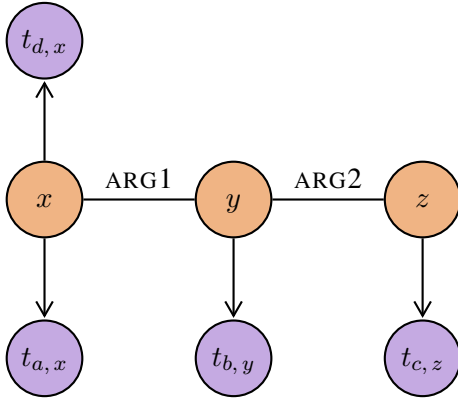

Figure 5: Logical inference for three pixies and four predicates of interest: we know whether a, b, c are true of x, y, z, respectively, and we would like to infer whether d is true of x. The distribution for $t_{d,x}$ depends on all the other truth values, because it is indirectly connected to them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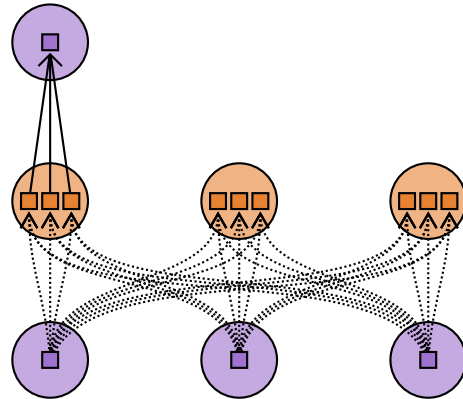

Figure 7: Variational inference for Fig. 5. The ARG1 and ARG2 links are not explicitly represented, but the mean-field probabilities are optimised to approximate the joint distribution induced by the links. Each dimension has an independent probability, but they are jointly optimised, so they depend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

---

[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.

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.

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

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 trans-

formed 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

---

[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

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:

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

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

$$P(A \cap C) = P(A) - P(A \setminus C)$$
$$= P(A) > 0$$

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

B Derivation of update rule

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

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

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

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

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

$$= \sum_{x|x_i=1} P(x|t_{c,x})\frac{1}{q_i} - \sum_{x|x_i=0} P(x|t_{c,x})\frac{1}{1-q_i}$$

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 $W_{jk}^{(l)}$ (and for incoming links, we simply take the transpose of this matrix). For clarity, we do not write bias terms.

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

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

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

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

$$\exp \sum W_{jk}^{(l)} \left(x_j^{(-i)} - x_j^{(+i)} \right) y_k$$

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

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

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

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