Recursive Neural Networks with Bottlenecks Diagnose (Non-)Compositionality

Verna Dankers[1] and **Ivan Titov**[1,2]
[1]ILCC, University of Edinburgh
[2]ILLC, University of Amsterdam
vernadankers@gmail.com, ititov@inf.ed.ac.uk

Abstract

A recent line of work in NLP focuses on the (dis)ability of models to generalise compositionally for artificial languages. However, when considering natural language tasks, the data involved is not strictly, or *locally*, compositional. Quantifying the compositionality of data is a challenging task, which has been investigated primarily for short utterances. We use recursive neural models (Tree-LSTMs) with bottlenecks that limit the transfer of information between nodes. We illustrate that comparing data's representations in models with and without the bottleneck can be used to produce a compositionality metric. The procedure is applied to the evaluation of arithmetic expressions using synthetic data, and sentiment classification using natural language data. We demonstrate that compression through a bottleneck impacts non-compositional examples disproportionately and then use the bottleneck compositionality metric (BCM) to distinguish compositional from non-compositional samples, yielding a compositionality ranking over a dataset.

1 Introduction

Compositional generalisation in contemporary NLP research investigates models' ability to compose the meanings of expressions from their parts and is often investigated with artificial languages (e.g. Lake and Baroni, 2018; Hupkes et al., 2020) or highly-structured natural language data (e.g. Keysers et al., 2019). For such tasks, the **local compositionality** definition of Szabó (2012, p. 10) illustrates how meaning can be algebraically composed:

> "The meaning of a complex expression is determined by the meanings its constituents have *individually* and the way those constituents are combined."

In natural language, there are fragments whose meaning can be composed as with arithmetic (e.g. "the cat is in the house"), while others carry contextual dependencies (e.g. "the kiwi grows on the farm"). Can we characterise whether an input's meaning arises from strictly local compositions?

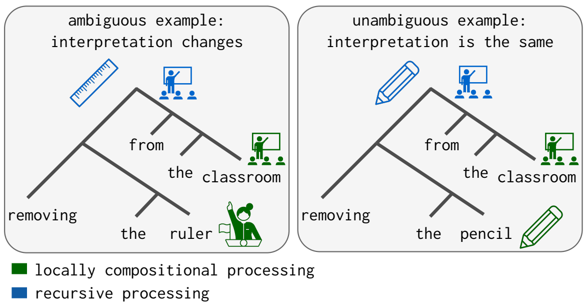


Figure 1: When processing this phrase, "the ruler" is interpreted differently when comparing recursive processing with local processing. We enforce local processing by equipping models with bottlenecks, and our **bottleneck compositionality metric (BCM)** then compares inputs' representations before and after compression through the bottleneck.

Existing work in that direction mostly focuses on providing a 'compositionality rating'[1] for figurative utterances since figurative language is assumed to be less compositional (Ramisch et al., 2016; Nandakumar et al., 2019; Reddy et al., 2011). Andreas (2018) suggests a general-purpose formulation for measuring the compositionality of examples using their numerical representations, through the *Tree Reconstruction Error* (TRE), expressing the distance between a model's representation of an input and a strictly compositional reconstruction of that representation. Determining how to compute that reconstruction is far from trivial.

Inspired by TRE, we use recursive neural networks, Tree-LSTMs (Tai et al., 2015), to process inputs according to their syntactic structure. We augment Tree-LSTMs with bottlenecks to compute

[1]We colloquially refer to the 'compositionality ratings' of phrases, but a more appropriate way to express the same would be to refer to 'the extent to which the meaning of a phrase arises from a compositional syntax and semantics'. After all, compositionality is a property of a language, not of a phrase.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4361–4378
December 7-11, 2022 ©2022 Association for Computational Linguistics

the task-specific meaning of an input in a more locally compositional manner. We use these models to distinguish more compositional examples from less compositional ones in a **bottleneck compositionality metric (BCM)**. Figure 1 provides an intuition for how a bottleneck can provide a metric. For fragments that violate the assumption that meanings of subexpressions can be computed locally (on the left side), one could end up with different interpretations when comparing a contextualised interpretation (in blue) with one locally computed (in green): disambiguating "ruler" requires postponed meaning computation, and thus local processing is likely to lead to different results from regular processing. For fragments that are non-ambiguous (on the right side) the two types of processing can yield the same interpretation because the interpretation of "pencil" is likely to be the same, with or without the context. The bottleneck hinders the model in postponing computations and more strongly impacts non-compositional samples compared to compositional ones, thus acting as a metric.

In the remainder of the paper, we firstly discuss the related work in §2. §3 elaborates on the models used that either apply a *deep variational information bottleneck* (DVIB) (Alemi et al., 2017) or compress representations through increased dropout or smaller hidden dimensionalities. In §4, we provide a proof-of-concept in a controlled environment where non-compositional examples are manually introduced, after which §5 elaborates on the natural language example of sentiment analysis. For both tasks, we (1) demonstrate that compression through a bottleneck encourages local processing and (2) show that the bottleneck can act as a metric distinguishing compositional from less compositional examples.

2 Related Work

Multi-word expressions The majority of the related work in the past two decades has discussed the compositionality of phrases in the context of figurative language, such as phrasal verbs ("to eat up") (McCarthy et al., 2003), noun compounds ("cloud nine" vs "swimming pool") (Reddy et al., 2011; Ramisch et al., 2016; Nandakumar et al., 2019), verb-noun collocations ("take place" vs "take a gift") (Venkatapathy and Joshi, 2005; McCarthy et al., 2007), and adjective-noun pairs ("nice house") (Guevara, 2010; Nandakumar et al., 2019). Compositionality judgements were obtained from humans, who indicated to what extent the meaning of the compound is that of the words when combined literally, and various computational methods were applied to learn that mapping. Those methods were initially thesaurus-based (McCarthy et al., 2003), relied on word vectors from co-occurrence matrices later on (Reddy et al., 2011), or employed deep neural networks (Nandakumar et al., 2019).

Compositionality by reconstruction TRE (Andreas, 2018) is a task-agnostic metric that evaluates the compositionality of data representations: TRE$(x) = \delta(f(x), \hat{f}_\eta(d))$. It is the distance between the representation of x constructed by f and the compositionally reconstructed variant $\hat{f}_\eta(d)$ based on the derivation of x (d). When employing the metric, one should define an appropriate distance function (δ) and define $\hat{f}_\eta$ parametrised by η. Andreas illustrates the TRE's versatility by instantiating it for three scenarios: to investigate whether image representations are similar to composed image attributes, whether phrase embeddings are similar to the vector addition of their components, and whether generalisation accuracy in a reference game positively correlates with TRE.

Bhathena et al. (2020) present two methods based on TRE to obtain compositionality ratings for sentiment trees, referred to as *tree impurity* and *weighted node switching* that express the difference between the sentiment label of the root and the other nodes in the tree. Zheng and Jiang (2022) ranked examples of sentiment analysis based on the extent to which neural models should *memorise* examples in order to capture their target correctly. While different from TRE, memorisation could be related to non-compositionality in the sense that non-compositional examples require more memorisation, akin to formulaic language requiring memorisation in humans (Wray and Perkins, 2000).

Other instantiations of the TRE are from literature on language emergence in signalling games, where the degree of compositionality of that language is measured. Korbak et al. (2020) contrast TRE and six other compositionality metrics for signalling games where the colour and shape of an object are communicated. Examples of such metrics are topographic similarity, positional disentanglement and context independence. These are not directly related to our work, considering that they aim to provide a metric for a *language* rather than single utterances. Appendix B.2 elaborates on topographic similarity and the metrics of Bhathena et al. (2020) and Zheng and Jiang (2022), comparing them to our metric for sentiment analysis.

Compositional data splits Recent work on compositional generalisation using artificial languages or highly-structured natural language data focuses on creating data splits that have systematic separation of input combinations in train and test data. The aim is to create test sets that should not be hard when computing meaning compositionally, but, in practice, are very challenging. An example compositionality metric for semantic parsing is *maximum compound divergence* (Keysers et al., 2019; Shaw et al., 2021), that minimises train-test differences in word distributions while maximising the differences in compound usage. This only applies to a data split as a whole, and – differently from the work at hand – does not rate individual samples.

More recently, Bogin et al. (2022) discussed a diagnostic metric for semantic parsing, that predicts model success on examples based on their local structure. Because models struggle with systematically assigning the same meaning to subexpressions when they re-appear in new syntactic structures, such structural deviation diagnoses generalisation failures. Notice that the aim of our work is different, namely identifying examples that are *not* compositional, rather than investigating generalisation failure for *compositional* examples.

3 Model

The model we employ is the *Tree-LSTM* (Tai et al., 2015), which is a generalisation of LSTMs to tree-structured network topologies. The LSTM computes symbols' representations by incorporating previous time steps, visiting symbols in linear order. A sentence representation is simply the final time step. A Tree-LSTM, instead, uses a tree's root node representation as the sentence representation, and computes the representation of a non-terminal node using the node's children.

Equations 1 and 2 illustrate the difference between the LSTM and an N-ary Tree-LSTM for the input gate. The LSTM computes the gate's activation for time step t using input vector x_t and previous hidden state h_{t-1}. The Tree-LSTM does so for node j using the input vector x_j and the hidden states of up to N children of node j.

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{1}$$

$$i_j = \sigma(W^{(i)}x_j + \sum_{\ell=1}^{N} U_\ell^{(i)}h_{j\ell} + b^{(i)}) \tag{2}$$

In addition to the input gate, the Tree-LSTM's specification for non-terminal j (with its kth child indicated as h_{jk}) involves an output gate o_j (equation analogous to 2), a forget gate f_{jk} (Equation 3), cell input activation vector u_j (equation analogous to 2, with the σ function replaced by $\mathtt{tanh}$), and memory cell state c_j (Equation 4). Finally, c_j feeds into the computation of hidden state h_j (Equation 5).

$$f_{jk} = \sigma(W^{(f)}x_j + \sum_{\ell=1}^{N} U_{k\ell}^{(f)}h_{j\ell} + b^{(f)}) \tag{3}$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell} \tag{4}$$

$$h_j = o_j \odot \mathtt{tanh}(c_j) \tag{5}$$

We apply a *binary* Tree-LSTM to compute hidden state h_j and memory cell state c_j, that thus uses separate parameters in the gates for the left and right child.

Tree-LSTMs process inputs according to their syntactic structure, which has been associated with more compositional processing (Socher et al., 2013; Tai et al., 2015). Yet, although the topology encourages compositional processing, there is no mechanism to explicitly regulate how much information is passed from children to parent nodes – e.g. given enough capacity, the hidden representations could store every input encountered and postpone processing until the very end. We add such a mechanism by introducing a **bottleneck**.

1. Deep Variational Information Bottleneck The information bottleneck of Alemi et al. (2017) assumes random variables X and Y for the input and output, and emits a compressed representation Z that preserves information about Y, by minimising the loss $\mathcal{L}_{IB}$ in Equation 6. This loss is intractable, which motivates the variational estimate $\mathcal{L}_{VIB}$ provided in Equation 7 (Alemi et al., 2017) that we use to train the **deep variational information bottleneck (DVIB)** version of our model.

$$\mathcal{L}_{IB} = \beta I(X, Z) - I(Z, Y) \tag{6}$$

$$\begin{aligned} \mathcal{L}_{VIB} = & \underbrace{\beta \mathop{\mathbb{E}}_{x}[\mathrm{KL}[p_\theta(z|x), r(z)]] +}_{\mathtt{information\ loss}} \\ & \underbrace{\mathop{\mathbb{E}}_{z\sim p_\theta(z|x)}[-\log q_\phi(y|z)]}_{\mathtt{task\ loss}} \end{aligned} \tag{7}$$

In the information loss, $r(z)$ and $p_\theta(z|x)$ estimate the prior and posterior probability over z, respectively. In the task loss, $q_\phi(y|z)$ is a parametric approximation of $p(y|z)$. In order to allow an analytic computation of the KL-divergence, we consider Gaussian distributions $r(z)$ and $p_\theta(z|x)$, namely $r(z) = \mathcal{N}(z|\mu_0, \Sigma_0)$ and $p_\theta(z|x) = \mathcal{N}(z|\mu(x), \Sigma(x))$, where $\mu(x)$ and μ_0 are mean vectors, and $\Sigma(x)$ and Σ_0 are diagonal covariance matrices. The reparameterisation trick is used to estimate the gradients: $z = \mu(x) + \Sigma(x) \odot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$.

We sample z once per non-terminal node, and average the KL terms of all non-terminal nodes, where x is the hidden state h_j or the cell state c_j (that have separate bottlenecks), and $\mu(x)$ and $\Sigma(x)$ are computed by feeding x to two linear layers. β regulates the impact of the DVIB, and is gradually increased during training. During inference, we use $z = \mu(x)$.

2. Dropout bottleneck Binary **dropout** (Srivastava et al., 2014) is commonly applied when training neural models, to prevent overfitting. With a probability p hidden units are set to zero, and during the evaluation all units are kept, but the activations are scaled down. Dropout encourages distributing the most salient information over multiple neurons, which comes at the cost of idiosyncratic patterns that networks may memorise otherwise. We hypothesise that this hurts non-compositional examples most. We apply dropout to the Tree-LSTM's hidden states (h_j) and memory cell states (c_j).

3. Hidden dimensionality bottleneck Similarly, decreasing the number of **hidden units** is expected to act as a bottleneck. We decrease the number of hidden units in the Tree-LSTM, keeping the embedding and task classifier dimensions stable, where possible.

The different bottlenecks have different merits: whereas the hidden dimensionality and dropout bottlenecks shine through simplicity, they are rigid in how they affect the model and apply in the same way at every node. The DVIB allows for more flexibility in how compression is achieved through learnt $\Sigma(x)$ and by requiring an overall reduction in the information loss term, without enforcing the same bottleneck at every node in the tree.

From bottleneck to compositionality metric BCM compares Tree-LSTMs with and without a bottleneck. We experiment with two methods, inspired by TRE (Andreas, 2018). TRE aims to find η such that $\delta(f(x), \hat{f}_\eta(d))$ is minimised, for inputs x, their derivations d, distance function δ, a model f and its compositional approximation $\hat{f}_\eta$.

- In the **TRE training** (BCM-TT) setup, we include the distance (δ) between the hidden representations of f and $\hat{f}_\eta$ in the loss when training $\hat{f}_\eta$. When training $\hat{f}_\eta$ with TRE training, f is frozen, and f and $\hat{f}_\eta$ share the final linear layer of the classification module. In the arithmetic task, δ is the *mean-squared error* (MSE) (i.e. the squared Euclidean distance). In sentiment analysis, δ is the Cosine distance function.
- In the **post-processing** (BCM-PP) setup, we train the two models separately, extract hidden representations and apply *canonical correlation analysis* (CCA) (Hotelling, 1936) to minimise the distance between the sets of hidden representations. Assume matrices $A \in \mathcal{R}^{d_A \times N}$ and $B \in \mathcal{R}^{d_B \times N}$ representing N inputs with dimensionalities d_A and d_B. CCA linearly transforms these subspaces $A' = WA$, $B' = VB$ to maximise the correlations $\{\rho_1, \dots, \rho_{\min(d_A, d_B)}\}$ of the transformed subspaces. We treat the number of CCA dimensions to use as a hyperparameter.

4 Proof-of-concept: Arithmetic

Given a task, we assign ratings to inputs that express to what extent their task-dependent meaning arises in a locally compositional manner. To investigate the impact of our metric on compositional and non-compositional examples in a controlled environment, we first use perfectly compositional arithmetic expressions and introduce exceptions to that compositionality manually.

4.1 Data and model training

Math problems have previously been used to examine neural models' compositional reasoning (e.g. Saxton et al., 2018; Hupkes et al., 2018; Russin et al., 2021). Arithmetic expressions are suited for our application, in particular since they can be represented as trees. We use expressions containing brackets, integers `-10` to `10`, and `+` and `-` operators – e.g. "`( 10 - ( 5 + 3 ))`" (using data from Hupkes et al., 2018). The output is an integer. This is modelled as a regression problem with the MSE loss. The 'meaning' (the numerical value) of a subexpression can be locally computed at each node in

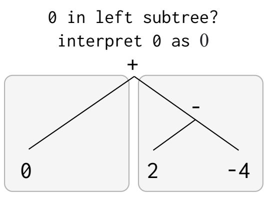
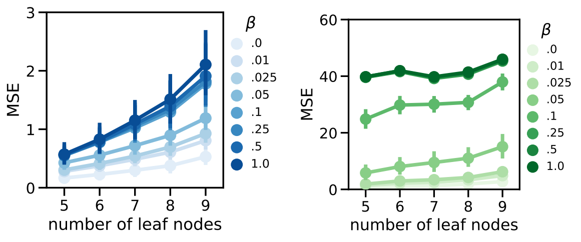
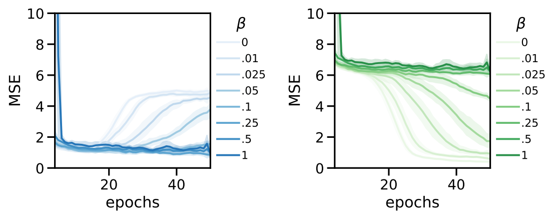

0 in left subtree? interpret 0 as 0
0 in right subtree? treat 0 as the 1st leaf node
+
-
0
2
-4
2
0
-4

Figure 2: Illustration of the 'exceptions' in the arithmetic task: the value of "0" depends on its position and on the value of the leftmost leaf node in the tree.



(a) Regular examples (b) Exceptions

Figure 3: Performance (MSE) on the arithmetic task for the Tree-LSTM with the DVIB (darker colours correspond to higher β). Exceptions have a contextual dependency and cannot be computed bottom up.



(a) Compositional targets (b) Adapted targets

Figure 4: Training dynamics for the Tree-LSTM with the DVIB: for all test examples we compute the MSE over the course of training on the validation set using (a) the compositional targets, or (b) the targets from the adapted dataset, of which a subset is not compositional.

the tree: there are no contextual dependencies.

In this controlled environment, we introduce exceptions by making "0" ambiguous. When located in the subtree headed by the root node's left child, it takes on its regular value, but when located in the right subtree, it takes on the value of the leftmost leaf node of the entire tree (see Figure 2). The model is thus encouraged to perform non-compositional processing to keep track of all occurrences of "0" and store the first leaf node's value throughout the tree. 88% of the training data are the original arithmetic expressions, and 12% are such exceptions. We can thus track what happens to the two categories when we introduce the bottleneck. The training data consist of 14903 expressions with 1 to 9 numbers. We test on expressions with lengths 5 to 9, using 5000 examples per length. The Tree-LSTMs trained on this dataset have embeddings and hidden states of sizes 150 and are trained for 50 epochs with learning rate 2e$-$4 with AdamW and a batch size of 32. The base Tree-LSTMs in all setups use the same architecture, namely the Tree-LSTM architecture required for the DVIB, but with $\beta = 0$. All results are averaged over models trained using ten different random seeds. In the Tree-LSTM, the numbers are leaf nodes and the labels of non-terminal nodes are the operators.[2]

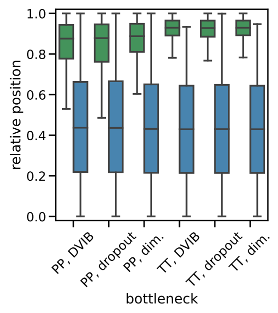
4.2 Task performance: Hierarchy without compositionality?

Figures 3a and 3b visualise the performance for the regular examples and exceptions, respectively, when increasing β for the DVIB. The DVIB disproportionately harms the exceptions; when β is too high the model cannot capture the non-local dependencies. Appendix A.1 shows how the hidden dimensionality and dropout bottlenecks have a similar effect. Figure 4 and Appendix A.2 provide insights in the training dynamics of the models: initially, all models will treat "0" as a regular number, independent of the bottleneck. Close to convergence, models trained with a low β have learnt to capture the ambiguities, whereas models trained with a higher β will remain in a more locally compositional state.[3]

Bottlenecks restrict information passed throughout the tree. To process an arithmetic subexpression, all that is needed is to pass on its outcome, not the subexpression itself – e.g. in Figure 2, one could simply store the value 6 instead of the subexpression "2 - -4". The former represents *local processing* and is more efficient (i.e. it requires storing less information), while the latter leads to information from the leaf nodes being passed to non-terminal nodes higher up the tree. Storing information about the leaf nodes would be required to cope with the exceptions in the data. That the model could get close to accurate predictions for these exceptions in the first place suggests Tree-LSTMs can process inputs according to the hierarchical structure without being locally compositional. Increasing compression using bottlenecks enforces local processing.

[2]Appendix C further elaborates on the experimental setup.

[3]Comparing 'early' and 'late' models may yield similar results as comparing base and bottleneck models. Yet, without labels of which examples are compositional, it is hard to know when the model transitions from the early to the late stage.


(a) All BCM variants

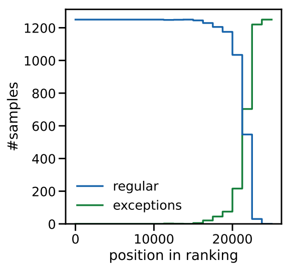

(b) Ranking for $\beta = 0.25$

Figure 5: Rankings of arithmetic examples. (a) shows the relative position of regular examples and exceptions in the rankings of all setups, where 0 corresponds to the start of the ranking and 1 to the end. (b) illustrates the result of BCM-TT with the DVIB, $\beta = 0.25$.

4.3 The Bottleneck Compositionality Metric

The bottleneck Tree-LSTM harms the exceptions disproportionately in terms of task performance, and through BCM we can exploit the difference between the base and bottleneck model to distinguish compositional from non-compositional examples. As laid out in §3, we use the TT or PP method to compare pairs of Tree-LSTMs: the base Tree-LSTM with $\beta = 0$, no dropout and a hidden dimension of 150 (**base model**) is paired up with Tree-LSTMs with the same architecture, but a different β, a different dropout probability p or a different hidden dimensionality d (**bottleneck model**). All Tree-LSTMs have a classification module that consists of two linear layers, where the first layer maps the Tree-LSTM's hidden representation to a vector of 100 units, and the second layer emits the predicted value. The 100-dimensional vector is used to apply the BCM:

- In BCM-PP the vector feeds into the CCA computation, that compares the hidden representation of the base model and the bottleneck model using their Cosine distance. We rank examples according to that distance, and use all CCA directions.
- In BCM-TT, the vector feeds into the TRE loss component. We train the base model, freeze that network, and add the MSE of the hidden representations of the bottleneck model and the base model to the loss. After training, the remaining MSE is used to rank examples.

Both setups have the same output, namely a compositionality ranking of examples in a dataset. A successful ranking would put the exceptions last.

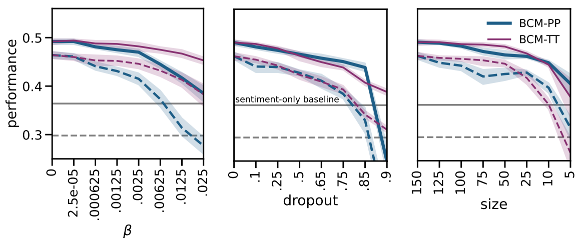

(a) DVIB (b) Dropout (c) Hidden dim.

Figure 6: The accuracy (solid) and macro-averaged F_1-scores (dashed) for the SST test set, for base models, bottleneck models and a sentiment-only baseline.

Figure 5a illustrates the relative position of regular examples and exceptions for all bottlenecks and BCM variants, for $\beta = 0.25$, $p = 0.5$ and $d = 25$. The change in MSE observed in §4.2 is reflected in the quality of the ranking, but the success does not depend on the specific selection of β, p or d, as long as they are large (β, p) or small enough (d). Figure 5b illustrates one of the rankings.

Summarising, we illustrated that recursive models can employ strategies that do not locally compose the meaning of arithmetic subexpressions but carry tokens' identities throughout the tree. We can make a model more locally compositional using bottlenecks and can use a model's hidden states to infer which examples required non-local processing afterwards, acting as our compositionality metric.

5 Sentiment analysis

We apply the metric to the task of sentiment analysis, for which Moilanen and Pulman (2007, p. 1) suggest the following notion of compositionality:

> "For if the meaning of a sentence is a function of the meanings of its parts then the global polarity of a sentence is a function of the polarities of its parts."

Sentiment is quasi-compositional: even though the sentiment of an expression is often a straightforward function of the sentiment of its parts, there are exceptions – e.g. consider cases of sarcasm, such as "I love it when people yell at me first thing in the morning" (Barnes et al., 2019), which makes it a suitable testing bed.

5.1 Data and model training

We use the SST-5 subtask from the Stanford Sentiment Treebank (SST) (Socher et al., 2013), that contains sentences from movie reviews collected

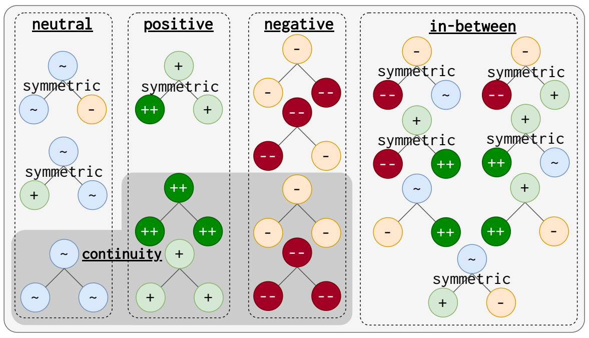


Figure 7: Illustration of the predictions of a sentiment-only baseline model. We indicate the predicted sentiment given two inputs. The labels range from very negative ('- -') to neutral ('∼') to very positive ('++').

by Pang and Lee (2005). The SST-5 subtask requires classifying sentences into one of five classes ranging from very negative to very positive. The standard train, development and test subsets have 8544, 1101 and 2210 examples, respectively. The sentences were originally parsed with the Stanford Parser (Klein and Manning, 2003), and the dataset includes sentiment labels for all nodes of those parse trees. Typically, labels for all phrases are included in training, but the evaluation is conducted for the root nodes of the test set, only.

Following Tai et al. (2015), we use GloVe word embeddings (Pennington et al., 2014), that we freeze across models.[4] The Tree-LSTM has 150 hidden units and is trained for 10 epochs with a learning rate of 2e$-$4 and the AdamW optimiser. During each training update, the loss is computed over all subexpressions of 4 trees. Training is repeated with 10 random seeds.[5] Figure 6 provides the performance on the test set for the base and bottleneck models, using the accuracy and the macro-averaged F_1-score. Tai et al. (2015) obtained an accuracy of 0.497 using frozen embeddings.

In sentiment analysis, as in our pseudo-arithmetic task, a successful model would often have to deviate from local processing. After all, the correct interpretation of a leaf node is often unknown without access to the context – e.g. in the case of ambiguous words like "sick" which is likely to refer to being ill, but could also mean "awesome". Being successful at the task thus requires a recursive model to keep track of information about (leaf) nodes while recursively processing the input, and more so for non-compositional examples than for compositional examples. As with the arithmetic task, local processing – enforced in the bottleneck models – should disproportionately hinder processing of non-compositional examples.

[4]The notion of local compositionality, relied on in this work, assumes that tokens are not disambiguated, which is why we refrain from using state-of-the-art contextualised representations. The focus of this work is on developing a compositionality metric rather than on improving sentiment analysis.

[5]Appendix C further elaborates on the experimental setup.

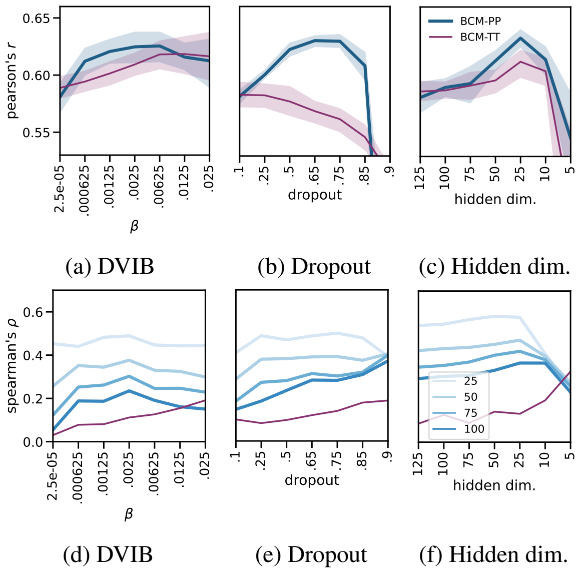


Figure 8: Pearson's r for the predictions of sentiment-only baselines and bottleneck models (a-c) and Spearman's ρ for the SST validation set compositionality ranking of the baselines and bottleneck models (d-f), when varying the number of CCA dimensions used.

5.2 A sentiment-only baseline

To assert that the bottlenecks make the models more compositional, we create a sentiment-only baseline that is given as input not words but their sentiment, and has a hidden dimensionality $d = 25$. Non-compositional patterns that arise from the composition of certain words rather than the sentiment of those words (e.g. "drop dead gorgeous") could hardly be captured by that model. As such, the model exemplifies how sentiment can be computed more compositionally. Figure 7 illustrates the default sentiment this model predicts for various input combinations. Its predictions can be characterised by i) predicting **positive** for positive inputs, ii) predicting **negative** for negative inputs, iii) predicting **neutral** if one input is neutral, iv) predicting a class **in between** the input classes or as v) predicting the same class as its inputs (**continuity**).

The performance of the model is included in Figure 6, and Figure 8 (a-c) indicates the Pearson's r between the sentiment predictions of bottleneck models and baseline models. Generally, a higher β or dropout probability, or a lower hidden dimen-

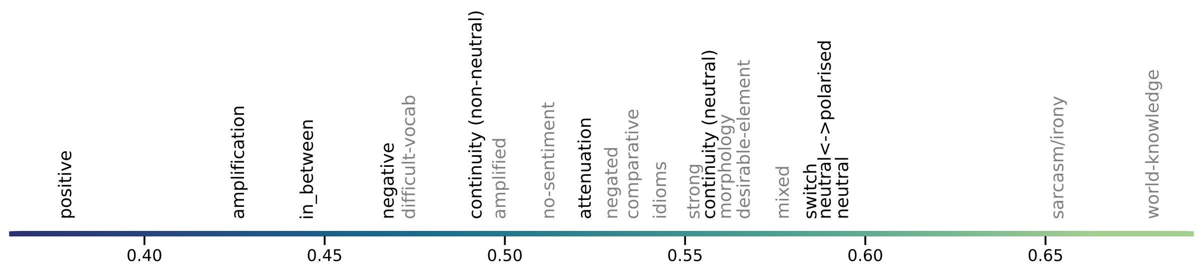


Figure 9: Categories of SST examples and their average position on the compositionality ranking visualised for the BCM-PP with the hidden dimensionality bottleneck and $d = 25$. Categories in black are assigned by us; categories in gray are from Barnes et al. (2019). The categories are further explained in the main text and Appendix B.1. Jittering was applied to better visualise overlapping categories.

sionality, leads to predictions that are more similar to this sentiment-only model, unless the amount of regularisation is too extreme, debilitating the model (e.g. for dropout with probability 0.9).

5.3 The Bottleneck Compositionality Metric

Now we use BCM to obtain a ranking over the SST dataset. We separate the dataset into four folds, train on those folds for the base and bottleneck model, and compute the cosine distances for the hidden representations of examples in the test sets (using BCM-PP or BCM-TT). We merge the cosine distances for the different folds, averaged over models trained with 10 random seeds, and order examples based on the resulting distance. We select the values for β, dropout and the hidden dimensionality, as well as the number of CCA directions to use, based on rankings computed over the SST validation data. Figure 8 (d-f) illustrates how the rankings of bottleneck models correlate with rankings constructed using the sentiment-only baseline. We select 25 directions, $\beta = 0.0025$, dropout $p = 0.65$ and a hidden dimensionality of 25 to compute the full ranking. Contrary to the arithmetic task, BCM-TT underperforms compared to BCM-PP.

Different from the arithmetic task, it is unclear which examples *should* be at the start or end of the ranking. Therefore, we examine the relative position of categories of examples in Figure 9 for the BCM-PP with the hidden dimensionality bottleneck, and in Appendix B.3 for the remaining rankings. The categories include the previously introduced ones, augmented with the following four:

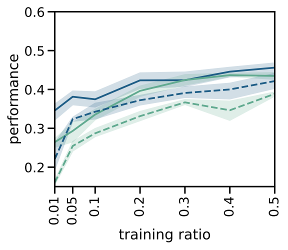
- **amplification**: the root is even more positive/negative than its top two children;
- **attenuation**: the root is less positive/negative than its top two children;
- **switch**: the children are positive/negative, but the root node flips that sentiment;
- **neutral↔polarised**: the inputs are sentiment-laden, but the root is neutral, or vice versa.

We also include characterisations from Barnes et al. (2019), who label examples from the SST test set, for which state-of-the-art sentiment models cannot seem to predict the correct label, including, for example, cases where a sentence contains mixed sentiment, or sentences with idioms, irony or sarcasm. Appendix B.1 elaborates on the meaning of the categories. Figure 9 illustrates the relative positions of our sentiment characterisations and those of Barnes et al. on that ranking. Patterns associated with more compositional sentiment processing, such as 'positive', 'negative' and 'in between' lead to hidden representations that are more similar between the base model and bottleneck models than the dataset average (the mid point, 0.5). Atypical patterns like 'switch' and 'neutral↔polarised', on the other hand, along with the characterisations by Barnes et al. lead to less similar hidden representations. Appendix B.3 presents the same results for all six rankings considered, along with example sentences from across the ranking, to illustrate the types of sentences encountered among the most compositional and the least compositional ones.

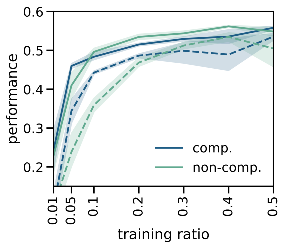
5.4 Example use cases

Compositionality rankings can be used in multiple manners, of which we illustrate two below.

When data is scarce: use compositional examples Assuming that most of natural language *is* compositional, one would expect that when limiting the training data, selecting compositional examples yields the best test performance. To investigate this, we train models on various training dataset sizes, and evaluate with the regular test set. The training data is taken from the start of the ranking for the 'compositional' setup, and from the end of the ranking for the 'non-compositional' setup



(a) LSTM



(b) Roberta

Figure 10: Change in SST test set accuracy (solid) and macro-averaged F_1-score (dashed) as the training set size increases, for LSTM and Roberta models. The examples are from the most (in blue) or the least compositional (in green) portion of the ranking from the BCM-PP metric with the hidden dimensionality bottleneck.

Model	**Comp.**		**Non-comp.**		**Random**	
	Acc.	F_1	Acc.	F_1	Acc.	F_1
Roberta	.546	.535	.516	.487	.565	.549
LSTM	.505	.485	.394	.310	.478	.447

Table 1: Performance on the new SST compositionality splits, generated using the ranking from the BCM-PP metric with the hidden dimensionality bottleneck.

(excluding test data), while ensuring equal distributions over input lengths and output classes. We train a two-layer bidirectional LSTM with 300 hidden units, and Roberta-base (Liu et al., 2019), using batch size 4. The models are trained for 10 and 5 epochs, respectively, with learning rates $2e-4$ and $5e-6$. Because the ranking is computed over full sentences, and not subexpressions, we train the models on the sentiment labels for the root nodes. Figure 10 presents the results, consolidating that when data is scarce, using compositional examples is beneficial.

Non-compositional examples are challenging For the same models, Table 1 indicates how performance changes if we redistribute train and test data such that the test set contains the most compositional examples, or the least compositional ones (keeping length and class distributions similar). The non-compositional test setup is more challenging, with an 11 percentage point drop in accuracy for the LSTM, and a 3 point decrease for Roberta.

In conclusion, applying the BCM to the sentiment data has confirmed findings previously observed for the arithmetic toy task. While it is harder to understand whether the method actually filters out non-compositional examples, both comparisons to a sentiment-only baseline, and the average position of cases for which the composition of sentiment is known to be challenging (e.g. for 'mixed' sentiment, 'comparative' sentiment or 'sarcasm'), suggest that compression acts as a compositionality metric. We also illustrated two ways in which the resulting ranking can be used.

6 Conclusion

This work presents the Bottleneck Compositionality Metric, a TRE-based metric (Andreas, 2018) that is task-independent and can be applied to inputs of varying lengths: we pair up Tree-LSTMs where one of them has more compressed representations due to a bottleneck (the DVIB, hidden dimensionality bottleneck or dropout bottleneck), and use the distance between their hidden representations as a per-datum metric. The method was applied to rank examples in datasets from most compositional to least compositional, which is of interest due to the growing relevance of compositional generalisation research in NLP, which assumes the compositionality of natural language, and encourages models to compose meanings of expressions rather than to memorise phrases as chunks. We provided a proof-of-concept using an arithmetic task but also applied the metric to the much more noisy domain of sentiment analysis.

The different bottlenecks lead to qualitatively similar results. This suggests that, while DVIB might be better motivated (it directly optimises an estimate of the Shannon information passed across the network), its alternatives may be preferable in practice due to their simplicity.

Because natural language itself is not fully compositional, graded metrics like the ones we presented can support future research, such as i) learning from data according to a compositionality-based curriculum to improve task performance, ii) filtering datasets to improve compositional generalisation, or iii) developing more and less compositional models depending on the desiderata for a task – e.g. to perform well on sentences with idioms, one may desire a more non-compositional model. In addition, the formulation of the metric was general enough to be expanded upon in the future: one could pair up other models, such as an LSTM and a Tree-LSTM, or a Transformer and its recursive variant, as long as one keeps in mind that the compositional reconstruction itself should not be too powerful. After all, even Tree-LSTMs could capture the exceptions in the arithmetic dataset despite their hierarchical inductive bias.

Limitations

We identify three types of limitations for the work presented:

- A **conceptual limitation** is that we work from a very strict definition of compositionality (*local* compositionality), which essentially equates language with arithmetic. While overly restrictive, current datasets testing compositional generalisation follow this notion. The framework might be extensible to more relaxed notions by allowing for token disambiguation by using contextualised token embeddings and only enforcing a bottleneck on the amount of further contextual integration within the model added on top of the token embeddings.
- In terms of **methodological limitations**, the use of only Tree-LSTMs – although well-motivated from the perspective of compositional processing – is a major limitation. Tree-LSTMs are most suited for sentence classification tasks, limiting the approach's applicability to sequence-to-sequence tasks. Nonetheless, the bottlenecks can be integrated in other types of architectures that process inputs in a hierarchical manner, such as sequence-to-sequence models inducing latent source and target trees (Kim, 2021) to yield an alternative implementation of the BCM.
  Our work also assumes that an input's tree structure is known, which might not always be the case. Therefore, the compositionality ranking obtained using BCM always depends on the trees used: what is non-compositional given one (potentially inadequate) structure might be more compositional given another (improved) structure.
- Lastly, the **evaluation** of our approach is limited in the natural domain through the absence of gold labels of the compositionality of examples in the sentiment analysis task, but for other tasks that could have been considered, the same limitation would have applied.

Acknowledgements

We thank Chris Lucas for his contributions to this project when it was still in an early stage, Kenny Smith for his comments on the first draft of this paper, and Matthias Lindemann for excellent suggestions for the camera-ready version. VD is supported by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) and the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences. IT acknowledges the support of the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO Vidi 639.022.518).

References

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2017. Deep variational information bottleneck. In *ICLR*.

Jacob Andreas. 2018. Measuring compositionality in representation learning. In *International Conference on Learning Representations*.

Jeremy Barnes, Lilja Øvrelid, and Erik Velldal. 2019. Sentiment analysis is not solved! Assessing and probing sentiment classification. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 12–23.

Hanoz Bhathena, Angelica Willis, and Nathan Dass. 2020. Evaluating compositionality of sentence representation models. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 185–193.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. *arXiv preprint arXiv:2201.05899*.

Henry Brighton and Simon Kirby. 2006. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial life*, 12(2):229–242.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 33–37. Citeseer.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intellgence Research*, 67:757–795.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Yoon Kim. 2021. Sequence-to-sequence learning with latent neural grammars. *Advances in Neural Information Processing Systems*, 34:26302–26317.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics*, pages 423–430.

Tomasz Korbak, Julian Zubek, and Joanna Rączaszek-Leonardi. 2020. Measuring non-trivial compositionality in emergent communication. In *NeurIPS 2020 workshop on Emergent Communication.*

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. 2018. Emergence of linguistic communication from referential games with symbolic and pixel input. In *International Conference on Learning Representations*.

Xiang Lisa Li and Jason Eisner. 2019. Specializing word embeddings (for parsing) by information bottleneck. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2744–2754.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach.

Diana McCarthy, Bill Keller, and John A Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment*, pages 73–80.

Diana McCarthy, Sriram Venkatapathy, and Aravind Joshi. 2007. Detecting compositionality of verb-object combinations using selectional preferences. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 369–379.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of the Recent Advances in Natural Language Processing International Conference*, pages 378–382. Association for Computational Linguistics.

Navnita Nandakumar, Timothy Baldwin, and Bahar Salehi. 2019. How well do embedding models capture non-compositionality? A view from multiword expressions. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 27–34.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Carlos Ramisch, Silvio Cordeiro, Leonardo Zilio, Marco Idiart, and Aline Villavicencio. 2016. How naked is the naked truth? A multilingual lexicon of nominal compound compositionality. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 156–161.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 210–218.

Jacob Russin, Roland Fernandez, Hamid Palangi, Eric Rosen, Nebojsa Jojic, Paul Smolensky, and Jianfeng Gao. 2021. Compositional processing emerges in neural networks solving math problems. In *Annual Conference of the Cognitive Science Society (CogSci)*, volume 2021, page 1767.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2018. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938.

Richard Socher, Alex Perelygin, Jean 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 2013 conference on* *empirical methods in natural language processing*, pages 1631–1642.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Zoltan Szabó. 2012. The case for compositionality. *The Oxford handbook of compositionality*, 64:80.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Sriram Venkatapathy and Aravind Joshi. 2005. Measuring the relative compositionality of verb-noun (vn) collocations by integrating features. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 899–906.

Alison Wray and Michael R Perkins. 2000. The functions of formulaic language: An integrated model. *Language & Communication*, 20(1):1–28.

Xiaosen Zheng and Jing Jiang. 2022. An empirical study of memorization in NLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6265–6278.

A Bottlenecks for arithmetic

A.1 Performance for other bottlenecks

Figures 11 and 12 display the MSE for models with the hidden dimensionality bottleneck and dropout bottleneck.

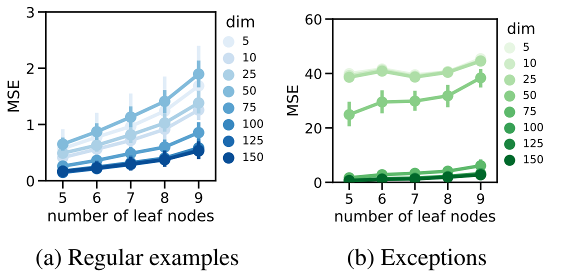

(a) Regular examples (b) Exceptions

Figure 11: Performance on the arithmetic task for the Tree-LSTM with varying hidden dimensionalities.

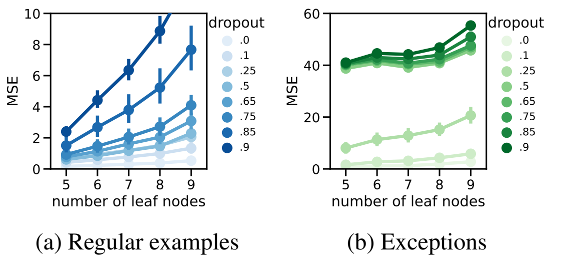

(a) Regular examples (b) Exceptions

Figure 12: Performance on the arithmetic task for the Tree-LSTM with varying dropout probabilities.

A.2 Bottleneck training dynamics

The exceptions in the arithmetic task had both a compositional and non-compositional interpretation, essentially giving us two sets of targets for measuring the models' performance during training using the validation data. Figures 4 (in the main paper), 13, and 14 illustrate how early on during training, the MSE is lowest for the compositional targets for all hyperparameters used for the bottlenecks: the models *overgeneralise* the regular interpretation of "`0`", applying it to all inputs. Later on, the models that have a high β, a high dropout probability or a small dimension stay in that 'compositional state', whereas the remaining models learn to capture the ambiguity.

B Sentiment analysis

B.1 Categories of Barnes et al. (2019)

Barnes et al. collect examples from sentiment analysis datasets that three state-of-the-art classifiers struggle with, and annotate them for the presence of 18 (para-)linguistic phenomena. We include the following categories from the SST test set in the visualisation of the SST ranking:

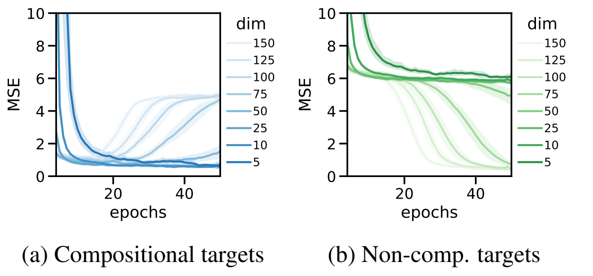

(a) Compositional targets (b) Non-comp. targets

Figure 13: Training dynamics for the hidden dimensionality bottleneck.

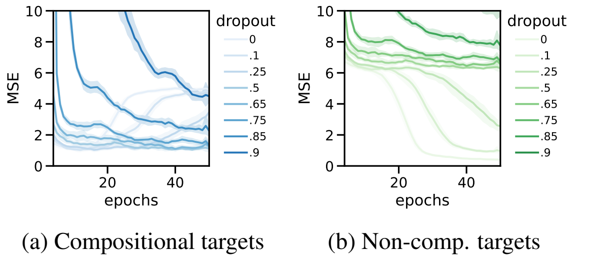

(a) Compositional targets (b) Non-comp. targets

Figure 14: Training dynamics for the dropout bottleneck.

1. *Negated*: phrases or sentences that are negated, where Barnes et al. identify a pattern of irrelevant negation throwing models off.
2. *Amplified*: cases where neutral modifiers act as strong contextual valence shifters.
3. *Strong*: *very* positive or *very* negative cases.
4. *Desirable element*: sentiment dominated by one 'desirable element', such as "pool".
5. *Comparative*: cases that express sentiment by comparison (e.g. using "better than").
6. *Idioms*: cases containing idioms, for which the mapping from the words to the sentiment is often not straightforward.
7. *Mixed*: mixed positive & negative sentiment.
8. *Difficult-vocab*: e.g. "engrossing and psychologically resonant suspenser".
9. *World-knowledge*: for instance comparisons between entities, where the entities imply a certain sentiment.
10. *Sarcasm/irony*: sarcasm is often present in negative examples, where the speaker is intending the opposite of what is said.
11. *No-sentiment*: neutral labelled examples.
12. *Morphology*: examples with morphological features that affect sentiment very positively or very negatively.

B.2 Alternative metrics

Tree impurity score (TIS) Bhathena et al.

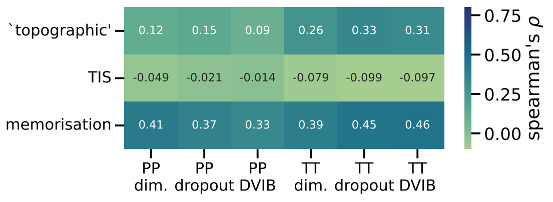


Figure 15: Illustration of how BCM rankings over the SST data correlate with baseline metrics that ought to be correlated: topographic similarity, tree impurity score and memorisation as per Spearman's ρ.

(2020) propose basic compositionality metrics that rely on the difference between the label of the root node, and labels of subexpressions. We report their *tree impurity score* (TIS): a simple metric that measures the absolute difference between the label of the root node and the average of all labels in a tree.

'Topographic' similarity A compositionality metric from language emergence literature is *topographic similarity* (Brighton and Kirby, 2006), that given a set of objects, their meanings and the associated signals computes the correlation of the distances between corresponding pairs of signals and meanings. For instance, Lazaridou et al. (2018) compare symbolic signals in a referential game using the Levenshtein edit distance, and compare vector meaning representations through cosine distance. Topographic similarity assumes that in a compositional language, similar signals should yield similar meaning representations. However, the use of edit distance to directly compare sentences does not readily transfer to sentiment analysis, where one word changed in the input space can yield a large change in the predicted sentiment.

To approximate topographic similarity of meanings and signals, we instead manipulate the input in ways that should yield a similar prediction, and then rate examples based on the change in the predictions observed. We replace nouns, verbs, adjectives or adverbs with a word that in SST has the same POS tag and sentiment label. If a change is observed, the example is more likely to be a non-compositional example.

We apply this to the base models from the main paper, by randomly replacing one token that is a verb, noun, adjective or adverb with a different token of the same POS tag and sentiment label. We make 50 such modifications per sentence per model seed. We record the average change in the predicted sentiment, and use this as a metric akin to topographic similarity.

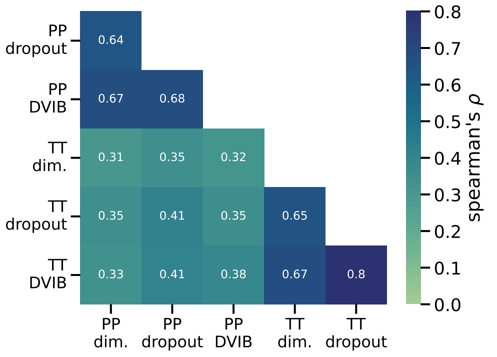


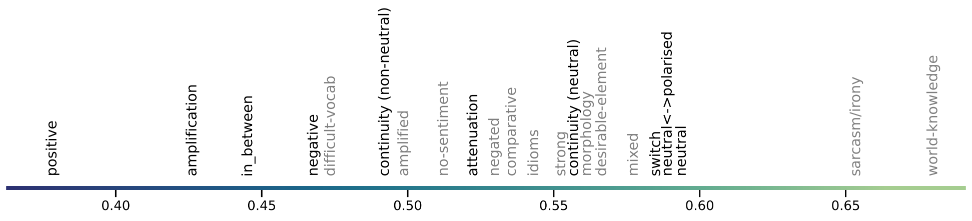
Figure 16: Illustration of how the BCM rankings over the SST dataset correlate as per Spearman's ρ.

Memorisation score Zheng and Jiang (2022) empirically evaluate the *long tail theory* posed by Feldman and Zhang (2020) (who validate their hypotheses using computer vision tasks), that states that for data distributions with a long tail, memorisation of training examples is required for near-optimal performance on the test data. Zheng and Jiang put this theory to the test for sentiment classification. The metric of Zheng and Jiang expresses how the likelihood of the target changes when an example is down-weighted during training. That change should be larger for memorised examples. Intuitively, non-compositionality and memorisation are related: non-compositional patterns in data require memorising the atypical interpretation of words in specific contexts. The metric is expected to be different from our metric – ours is measured using *test* data, while memorisation occurs during training – but a positive correlation is expected between the two, nonetheless. Zheng and Jiang use the binary SST subtask and report their scores on preprocessed versions of SST sentences. We report the correlation for the examples for which we could find matching surface forms only.

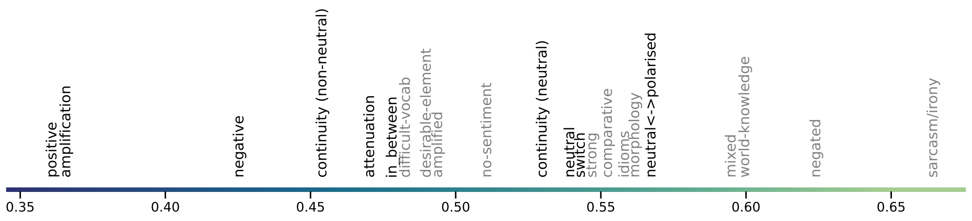
Figure 15 indicates how the different BCM rankings correlate with these three alternative metrics. Surprisingly, TIS negatively correlates with our rankings, but only weakly. This may be due to the simplicity of that metric, that ignores the tree structure and simply averages all sentiment of all nodes. The devised 'topographic' similarity positively correlates with our rankings, but only up to $\rho = 0.33$. The memorisation score has a moderate correlation with our rankings, of up to $\rho = 0.46$ for the BCM-TT. Figure 16 illustrates how the rankings correlate with each other. The figure suggests that TRE training yields rankings that are still quite different from the post-processing ones.

B.3 Additional results for SST rankings

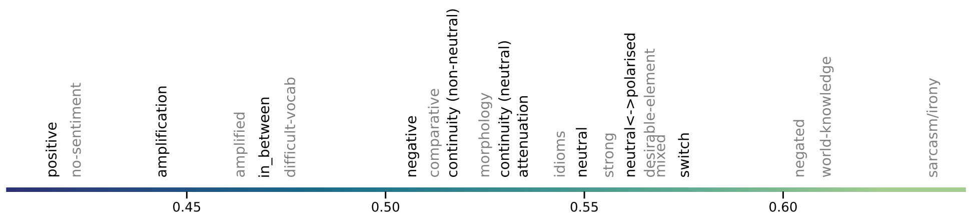
Here, we present further results for the BCM rankings over the SST datasets: firstly, Figure 17 provides the rankings for the six different BCM setups. They have commonalities, but also differences, e.g. in the BCM-TT setups the 'neutral' sentiment is closer to the end of the ranking.



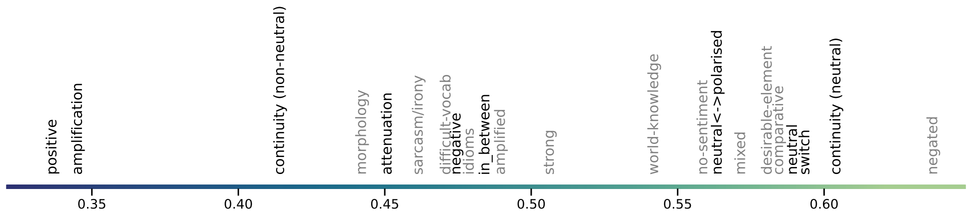
(a) BCM-PP, Hidden dimensionality bottleneck



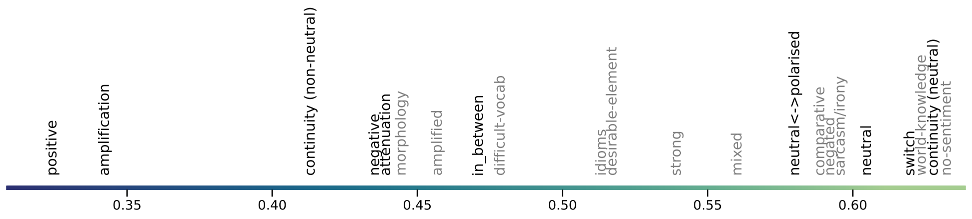
(b) BCM-PP, DVIB bottleneck



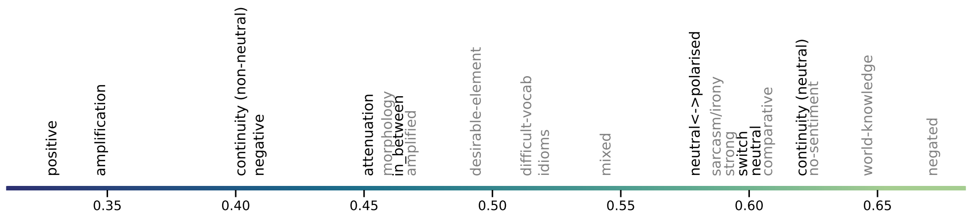
(c) BCM-PP, Dropout bottleneck



(d) BCM-TT, Hidden dimensionality bottleneck



(e) BCM-TT, DVIB bottleneck



(f) BCM-TT, Dropout bottleneck

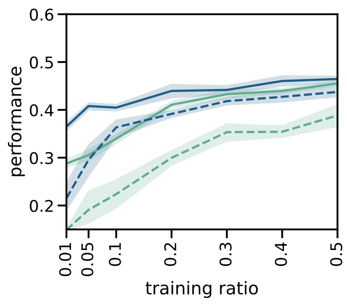
Figure 17: Example categories for SST test set examples, and their average position on the compositionality ranking for the hidden dimensionality bottleneck with $d = 25$, the dropout bottleneck with $p = 0.65$ or the DVIB with $\beta = 0.0025$. The categories in black are assigned by us, and the categories in gray are from Barnes et al. (2019). Jittering was applied to better visualise overlapping categories.

Secondly, Table 2 provides example sentences from different parts of the rankings, randomly sampled. 0 indicates the most compositional examples, and 1 the least compositional ones.

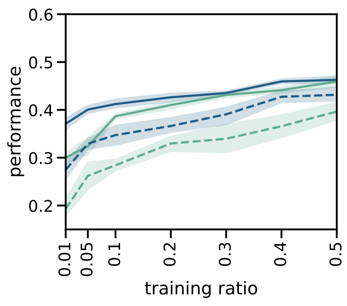
Relative position	Target	Sentence
- BCM-PP, Hidden dim. bottleneck		
0.07	3	tsai convincingly paints a specifically urban sense of disassociation here .
0.17	0	the film desperately sinks further and further into comedy futility .
0.21	3	a cop story that understands the medium amazingly well .
0.40	3	one scarcely needs the subtitles to enjoy this colorful action farce .
0.42	1	the entire movie is about a boring , sad man being boring and sad .
0.53	2	not everyone will play the dark , challenging tune taught by the piano teacher .
0.66	3	this is the stuff that disney movies are made of .
0.75	3	daughter from danang sticks with its subjects a little longer and tells a deeper story
0.86	2	not kids , who do n't need the lesson in repugnance .
0.98	0	lacks heart , depth and , most of all , purpose .
- BCM-PP, Dropout bottleneck		
0.09	4	it 's a cool event for the whole family .
0.14	0	done in mostly by a weak script that ca n't support the epic treatment .
0.20	3	some movies are like a tasty hors-d'oeuvre ; this one is a feast .
0.37	4	my oh my , is this an invigorating , electric movie .
0.41	3	its director 's most substantial feature for some time .
0.51	4	the modern master of the chase sequence returns with a chase to end all chases
0.62	3	like its bizarre heroine , it irrigates our souls .
0.78	1	sadly , ‘ garth ’ has n't progressed as nicely as ‘ wayne . ’
0.86	2	you 're too conscious of the effort it takes to be this spontaneous .
0.95	0	a film of empty , fetishistic violence in which murder is casual and fun .
- BCM-PP, DVIB		
0.07	4	highly recommended viewing for its courage , ideas , technical proficiency and great acting .
0.18	3	it manages to squeeze by on angelina jolie 's surprising flair for self-deprecating comedy .
0.25	1	goes on and on to the point of nausea .
0.38	3	the best part about “ gangs ” was daniel day-lewis .
0.44	2	a dopey movie clothed in excess layers of hipness .
0.51	2	a perplexing example of promise unfulfilled , despite many charming moments .
0.64	0	the entire film is one big excuse to play one lewd scene after another .
0.79	1	the problematic characters and overly convenient plot twists foul up shum 's good intentions .
0.82	3	the jabs it employs are short , carefully placed and dead-center .
0.92	2	then nadia 's birthday might not have been such a bad day after all .
- BCM-TT, Hidden dim. bottleneck		
0.00	3	the story is smart and entirely charming in intent and execution .
0.02	2	for single digits kidlets stuart little 2 is still a no brainer .
0.17	1	outer-space buffs might love this film , but others will find its pleasures intermittent .
0.29	3	although shot with little style , skins is heartfelt and achingly real .
0.37	3	a knowing look at female friendship , spiked with raw urban humor .
0.50	3	there 's an energy to y tu mamá también .
0.57	2	a piquant meditation on the things that prevent people from reaching happiness .
0.63	3	too daft by half ... but supremely good natured .
0.71	1	the feature-length stretch ... strains the show 's concept .
0.88	1	a thriller without thrills and a mystery devoid of urgent questions .
0.97	1	the movie does n't generate a lot of energy .
- BCM-TT, Dropout bottleneck		
0.03	3	an extremely funny , ultimately heartbreaking look at life in contemporary china .
0.14	1	the pretensions – and disposable story – sink the movie .
0.22	4	that rara avis : the intelligent romantic comedy with actual ideas on its mind .
0.39	2	admirable , certainly , but not much fun to watch .
0.50	1	the end result is a film that 's neither .
0.59	2	throwing it all away for the fleeting joys of love 's brief moment .
0.63	4	kids should have a stirring time at this beautifully drawn movie .
0.77	3	the obnoxious title character provides the drama that gives added clout to this doc .
0.87	0	stitch is a bad mannered , ugly and destructive little **** .
0.98	0	there is no pleasure in watching a child suffer .
- BCM-TT, DVIB		
0.03	4	quite simply , a joy to watch and – especially – to listen to .
0.14	1	a fake street drama that keeps telling you things instead of showing them .
0.22	1	it tries too hard , and overreaches the logic of its own world .
0.34	1	plays less like a coming-of-age romance than an infomercial .
0.48	4	it 's one of the most honest films ever made about hollywood .
0.52	3	a low-key labor of love that strikes a very resonant chord .
0.69	3	it 's never laugh-out-loud funny , but it is frequently amusing .
0.71	3	a meditation on faith and madness , frailty is blood-curdling stuff .
0.81	1	a beautifully shot but dull and ankle-deep ‘ epic . ’
0.91	2	it uses the pain and violence of war as background material for color .

Table 2: Examples from across the six rankings, randomly sampled.

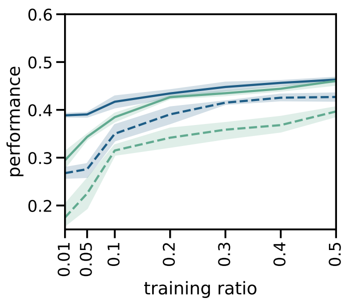
Thirdly, Figure 18 illustrates how training on different subsets of the ranking leads to different performance on the test set. In general, it is better to train on the compositional examples.



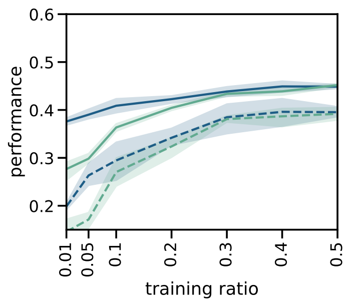
(a) LSTM, Hidden dim., BCM-PP



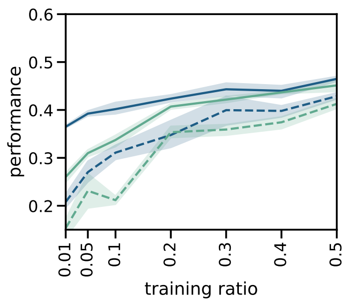
(b) LSTM, dropout, BCM-PP



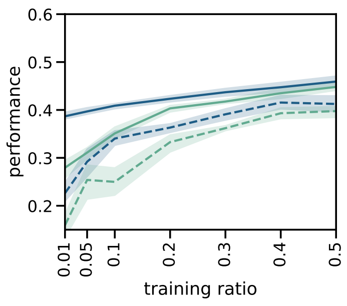
(c) LSTM, DVIB, BCM-PP



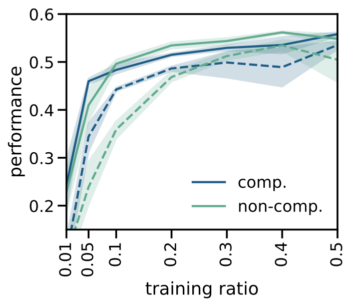
(d) LSTM, Hidden dim., BCM-TT



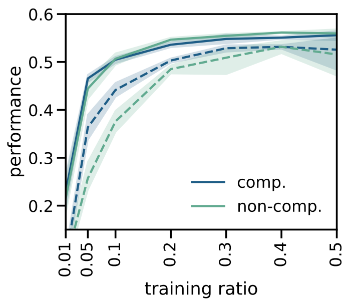
(e) LSTM, dropout, BCM-TT



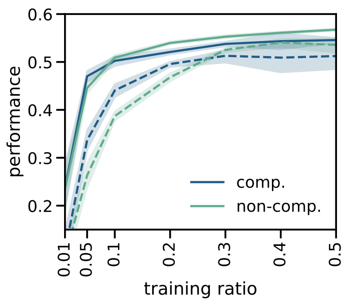
(f) LSTM, DVIB, BCM-TT



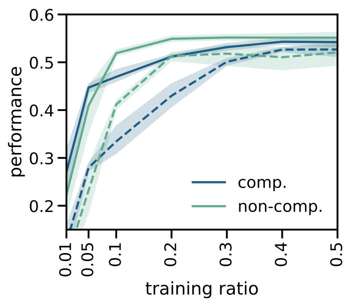
(g) Roberta, Hidden dim., BCM-PP



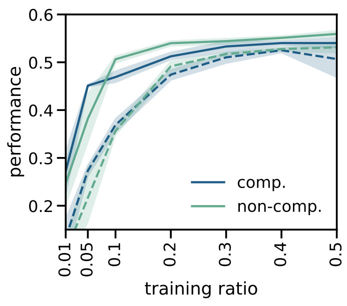
(h) Roberta, dropout, BCM-PP



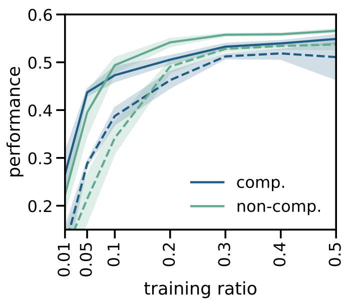
(i) Roberta, DVIB, BCM-PP



(j) Roberta, hidden dim., BCM-TT



(k) Roberta, dropout, BCM-TT



(l) Roberta, DVIB, BCM-TT

Figure 18: Change in SST test set accuracy (solid) and macro-averaged F_1-score (dashed) as the training set size increases, for LSTM and Roberta models.

C Reproducibility details

Visit https://github.com/vernadankers/bottleneck_compositionality_metric for the code and data. Below we collect an overview of the settings involved in the model training and model evaluation:

- **Model specifications**: Following Tai et al. (2015) we use 150 hidden dimensions, and 300-dimensional word embeddings for the sentiment analysis task. For consistency, we adopt the 150 dimensions in the arithmetic task, as well, but reduce the size of the word embeddings to 150. The arithmetic base models have 426k trainable parameters, and the sentiment base models have 514k trainable parameters (excluding the frozen word embeddings).
- **Training procedure**: The sentiment and arithmetic models are trained for 10 and 50 epochs, respectively, based on model convergence. These numbers are fixed across models, to ensure that when multiple models are combined in the compositionality metric, they have been trained for the same amount of time. For sentiment, a batch size of 4 was used, and batch sizes {1, 4, 8} were experimented with. For arithmetic, batch sizes {16, 32, 64} were experimented with across 5 seeds, where 32 was selected. Selection is based on validation performance across five seeds and training speed (e.g. while 1 slightly outperformed 4 for sentiment analysis, we opted for 4 for computational efficiency). For both tasks, learning rates {0.001, 2e-4, 1e-4} were experimented with, and 2e-4 was selected based on model performance on the validation set across 5 seeds. In these hyperparameter trials, the accuracy was used for sentiment analysis and the MSE was used for the arithmetic task.
- **Hidden dimensionality bottleneck**: To increase compression, the dimensionality should clearly become smaller, so we manually select 8 values to run ranging from 150 (the standard dimension) to 5.
- **Dropout bottleneck**: For dropout, the closer to 1, the more compression there is. We manually select 8 values to run from 0 (the standard amount) to 0.9.
- **DVIB**: For DVIB, preliminary experiments indicated that $\beta > 1$ for arithmetic or $\beta > 0.1$ for sentiment debilitates the model. We manually selected 8 values to run for β based on that information. We use the implementation of Li and Eisner (2019) for the DVIB, available at: https://github.com/XiangLi1999/syntactic-VIB.
- **Evaluation metrics**: The evaluation metric used for arithmetic is the MSE. The evaluation metrics for sentiment analysis are the accuracy of the predicted sentiment class, and the F_1-score, macro-averaged.
- **Number of runs & run time**: The results for both tasks are averaged over ten seeds. Training one model with one seed on CPU lasts up to 40 minutes for the sentiment analysis models, and up to 30 minutes for the arithmetic task. The TRE-training setup typically takes twice as long. We utilise CPUs from the icelake partition of the CSD3 cluster.

For the datasets used, the following are relevant details in terms of their size and preprocessing:

- **Arithmetic task**: The arithmetic task was generated using the implementation of Hupkes et al. (2018), available at https://github.com/dieuwkehupkes/processing_arithmetics. We augment the data with the ambiguous examples ourselves. The training data consist of 14903 expressions with 1 to 9 numbers. We test on expressions with lengths 5 to 9, using 5000 examples per length. 2100 additional examples are used as validation data, to track the model's behaviour during training.
- **Stanford Sentiment Treebank**: We collect the SST data from the pytreebank package (https://github.com/JonathanRaiman/pytreebank). For the (Tree-)LSTM models, we further preprocess the inputs by lowercasing the sentences. We use SST-5, that classifies sentences using five classes ranging from very negative to very positive. The standard train, validation and test subsets have 8544, 1101 and 2210 examples, respectively. Each node in the input trees has its own sentiment label.

For the experiments contained in Section 5.4, we did not run an extensive hyperparameter search. Those results are averaged over five seeds, and the models were trained on one NVIDIA A100-SXM-80GB, where training one seed lasted up to 15 minutes.