Sentence Querying and Compositionality: Who is Doing What and Where?

Yke Rusticus
University of Amsterdam
yke.rusticus@student.uva.nl

Maarten Peters
University of Amsterdam
maarten.peters@gmail.com

Abstract

Within the field of machine translation neural networks tend to learn by building a probabilistic model of natural language, but struggle to generalize to unseen situations and fail to learn the compositional nature of language. Our research proposes a probabilistic context free grammar resembling natural language, training neural networks to answer a who/what/where question based of a generated sentence. We have found that testing for systematicity and productivity (Hupkes et al., 2020), networks seem to perform well above random baselines, but they fail to generalize consistently. Future research could be performed on the role of punctuation in sentences to facilitate hierarchy, as well as other networks such as Transformers which may be better suited in compositional learning tasks.

1 Introduction

One of the goals in AI is to develop systems that generalize well to unseen situations. However, if we look at tasks that seem trivial to humans, models surprisingly often fail in doing so. To follow an example given by Hupkes et al. (2020), if we know what 'brown dog' and 'black cat' mean, we should also be able to understand 'brown cat'. This is due to the notion of compositionality, which states that the meaning of a whole can be derived from the meaning of its parts and the way they are syntactically combined. The above example highlights one of five ways to test compositionality as listed by Hupkes et al. (2020). In this work we used two of these tests to evaluate the compositional properties of our models, which was trained on a simple artificial language. Different than previous work that uses arithmetic (Veldhoen et al., 2016), function-based (Hupkes et al., 2020), or other non-natural languages, we test our models on language that has a more similar structure to natural language. Using a probabilistic context free grammar (PCFG), we generated sentences that contain a *who*, *what*, *where* and possibly a *when*, which in turn consists of the same components. Our models are expected to interpret which *who*, *what* and *where* belong together in a sequence-to-sequence machine translation task. In order to test the models' compositional properties, we perform systematicity and productivity tests, which tell how well the model can combine known rules and how well it can generalizes to longer sequences than trained on. We have observed that our LSTM-based and GRU-based networks perform well above our random baseline on the main task. Before analyzing the results, we will discuss previous related work in the next section. Section 3 describes our approach in detail, and in Section 4 experiments and results are presented. We conclude and discuss this work in Section 5.

2 Related Work

In this section we discuss a few notable works in the field of compositionality that are relevant for this work. First of all, Bowman et al. (2015) compared tree-based models and recurrent models. Their task was to discover the semantic relations between bracketed logical expressions. They found that, even though the data was tree-structured, the recurrent model was able to learn the task. Veldhoen et al. (2016) focused mainly on the interpretation of how these models represent hierarchical information, using so-called diagnostic classifiers. They showed that their diagnostic classifier was effective in recovering the approach probably taken by the model to solve the problem. Similarly to our approach, Lake and Baroni 2017 trained recurrent models on a sequence-to-sequence machine translation task. They found that generalization by the models worked counter-intuitively. They struggle

S	→	WWW	0.25
S	→	WWWW	0.75
WWW	→	who WHAT WHERE	0.5
WWW	→	WHERE who WHAT	0.5
WWWW	→	WWW WHEN	0.25
WWWW	→	WHEN WWW	0.25
WWWW	→	who WHAT WHEN WHERE	0.25
WWWW	→	WHERE WHEN who WHAT	0.25
WHEN	→	func_when S	1.0
WHAT	→	func_what what	1.0
WHERE	→	func_where where	1.0
func_when	→	'when'	1.0
func_what	→	'went'	1.0
func_where	→	'at'	1.0
who	→	'person0' | 'person1' | ...	U
what	→	'doing0' | 'doing1' | ...	U
where	→	'location0' | 'location1' | ...	U

Table 1: The PCFG used in this work, with non-terminals and terminals as uppercase and lowercase words respectively. Sampling probabilities are given in the right column, where U stands for uniform.

to handle longer sequences than seen in training and they find it difficult to combine learned rules compositionally, in ways humans do not. Works discussed up till now all used more or less different definitions of compositionality. Therefore more recently, Hupkes et al. (2020) listed five tests for compositionality, to guide future work in the field. These tests include systematicity, productivity, substitutivity, localism and overgeneralization. In this work we focus on systematicity and productivity.

3 Approach

Samples of our artificial language consist of two parts: question(s), and a sentence. The sentences are generated using the PCFG given in Table 1. We keep track of the relations in the sentence, i.e. which *who*, *what* and *where* belong together according to the hierarchy of the (sub)sentences. These relations are then used to generate the question(s). Possible questions start with 'who', 'what' or 'where', and are followed either by a person, action, or a location that appears in the sentence. 'who' is never followed by a person, 'what' never by an action and 'where' never by a location. The general question-answer interpretation rules can be found in Table 2. Per sentence, we sampled up to N questions uniformly. For each question we expect the model to answer it according to the order in which they are asked. An example sequence is: [questions] *what location1 who doing4* [sentence] *at location1 when person3 went doing4 at location10 person6 went doing9*. The correct interpretation would be: *doing9 at location1 person3 went doing4*. It could happen that the same word appears multiple times in a sentence. If a question is asked regarding that word, the model is expected to concatenate answers to that question according to the hierarchy of the sentence structure. For example, if an answer can be formed with the main sentence, it would be in front of an answer that is formed based on a sub-sentence (indicated by the word 'when').

Question format	Answer format
who doing	person went doing
who location	person at location
what person	person went doing
what location	doing at location
where person	person at location
where doing	at location doing

Table 2: General question and answer formats.

The choice of this language is motivated by the kind of sentences we encounter on a daily basis. When we hear sentences like these (obviously with more realistic names for people, locations and actions) we automatically keep track of the relations, regardless of what names are used. This language therefore allows us to investigate whether this same ability is recognized in neural models, through the systematicity test. We can perform this test by letting models train on sequences that exclude a few names, locations and actions from their questions, while still have them be present in the sentence. When evaluating, we include the left out samples in the test set and measure the performance. Additionally, we can test productivity by splitting samples based on their number of *whens* or their number of questions. We base our experiments on a data set of overall 40k unique samples. This set is split for training (30k), development (5k), and testing (5k) to evaluate the main task. The data was generated with 20 unique names for *who*, *what* and *where*. Samples consist of up to 3 questions and 5 whens. We used the same data set to make the systematicity and productivity test splits, the sizes of which are given in the next section.

4 Experiments and Results

We implemented all of the following using the publicly available neural machine translation tool OpenNMT[1]. Two bi-directional recurrent neural networks (RNN) were used throughout the experiments, one based on Gated Recurrent Units (GRU) and one on Long Short-Term Memory (LSTM).

[1] https://github.com/OpenNMT/OpenNMT-py

Model	Accuracy
Random	0.444 ± 0.002
GRU	0.583 ± 0.005
LSTM	$\mathbf{0.614} \pm 0.003$

Table 3: Main task accuracies. After further finetuning the LSTM scored an accuracy of 0.695 on a single run. We used optimizer Adam with learning rate 0.001, 25000 steps and a batchsize of 64.

Similar to Hupkes et al. (2020), we used two layers, a 512 word embedding space and 512 hidden units. After analysis of the results, we further finetuned and manually experimented with the hyperparameters. We used the development set to tune parameters. For both the embedding and the hidden dimensions we searched over [128, 256, 384, 512]. We varied the number of layers (2-4) and the number of attention heads (6-10). As the LSTM model showed most promise, we settled its best parameters at 384 for word embedding and hidden dimensions, with 3 layers and its default 8 attention heads. The performances of the models are compared to a random *rule-based* baseline, which translates samples correctly according to word order and structure, but randomly samples the corresponding person, action or location names from the sentence. All of the following results are averaged over three runs and errors are given as one standard deviation.

The performances on the main task are given in Table 3. It shows that both models significantly outperform the random baseline. In order to investigate that this was not only the case for "easy" samples, we looked at accuracies for different search distances. We defined search distance as the distance two words are apart that together form the answer to a question. Figure 1 shows that for samples with search distances up to 20, both models either outperform, or shows similar results with the random baseline.

For the systematicity test, we excluded samples with questions regarding person0-2, doing0-2 and location0-2 from the training set. The upper left subplot in Figure 2 shows a large performance drop if we subsequently test the models including these samples. Interestingly though, the GRU scores significantly better than the LSTM. This might be due to the differences between the structure of the cells in both recurrent models, which we will discuss further in Section 5. In Figure 3 we compare the performances specifically on either person0-2, doing0-2 or location0-2. This figure suggests that

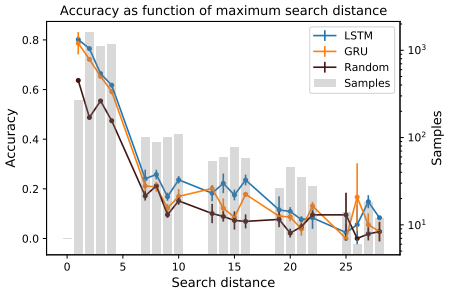

Figure 1: Accuracies and number of samples per search distance. If multiple questions are asked, the maximum search distance is taken.

Data sets	Train split	Test split(s)
Systematicity	25824	Excl.: 4000; Incl.: 10176
Productivity test 1	22674	Each (1 - 3 questions): 2000
Productivity test 2	22673	Same sets as test 1
Productivity test 3	24075	Each (0 - 4 whens): 2000
Productivity test 4	27704	Same sets as test 3
Productivity test 5	18556	Same sets as test 3

Table 4: Number of samples per test.

questions regarding location0-2 were in general the hardest. This was as expected, as our data was structured such that the location could be placed anywhere in the sentence, while the person and its action always follow each other.

We performed five productivity tests, which are also visualized in Figure 2. In tests 1 and 2, the models were trained on samples containing 1 or 2 and 1 or 3 questions respectively. In both cases the models fail to generalize well to an unseen number of questions, and show accuracies below the random baseline. In tests 3 to 5, we varied the number of *when*s the models were trained on. In test 3, the models were trained on samples with up to 2 *when*s; in test 4 on samples up to 3 *when*s; and in test 5 on samples with 0, 2 or 4 *when*s. Surprisingly, the results for all three tests are highly similar, and in each case the models outperform the random baseline. The number of samples that were used for each test are given in Table 4.

5 Discussion

Although the performances in general were not very high on this task, we found that our LSTM- and GRU-based models outperformed a "smart" random baseline. We confirmed that the models were not only performing well on easy samples, but also on samples where relevant words lie far from each other in the sentence. The fact that they outperformed the baseline for most samples up

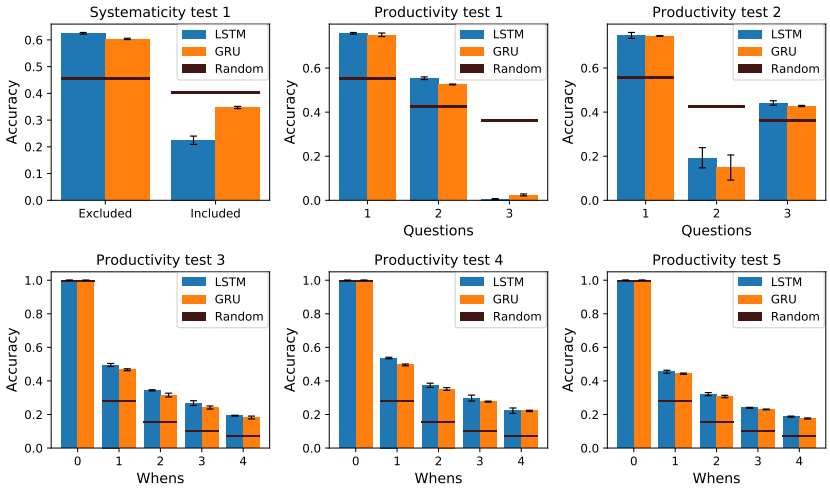

Figure 2: The systematicity and productivity tests for the models compared to the random baseline.

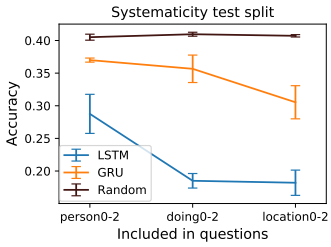

Figure 3: Zoom-in on the systematicity test. This figure shows the accuracies per entity left out from training.

to search distances of 20, suggests that the models have learned to capture some relevant relations within the sentences, rather than just copying the closest words they could find to form an answer.

The LSTM and GRU performed similarly on most tests, except for the systematicity test. This could be due to the fact that a GRU has no explicit forget gate, as opposed to an LSTM. As specific names were excluded from questions in training, the LSTM model might have been actively ”forgetting” these names when they were encountered in a sentence, restricting itself from learning relevant embeddings for these words. The simplicity of the GRU, as compared to the LSTM, might have been a benefit in this specific case. For future research, it is suggested to investigate how much left-out samples are needed in training to make the models outperform the random baseline on this systematicity test.

The productivity tests showed relatively intuitive results. In tests 1 and 2 we found that the models struggled more with extrapolating than with interpolating. Furthermore, by varying the number of questions we implicitly varied the expected output length of the models, which clearly had a large effect on the behaviour of the models. In tests 3 to 5 we only varied the input length of the models, which had very little effect on their behaviour. This suggests that the models were capable of focusing on relevant subsets of the sentences, regardless of the input lengths they were trained on.

As the results showed only slight differences over all between GRU and LSTM we have experimented with Transformers (Vaswani et al., 2017), as they have shown promise in related works (Hupkes et al., 2020). Experimentation showed no decisive results, as training and validation of Transformers required larger datasets and specific tuning. We have, however, observed above random results with modified training data and accuracies over longer search distances (visualized in Figure 7).

Another interesting research area would be to investigate the effects of punctuation on the models' performance. Even though commas are often used to disambiguate sentences in natural language, they could also be used to provide a clearer overview of sentence structure. In our specific task, commas could be inserted as additional tokens in places where a sub-sentence starts (i.e. before ”when”) and where it ends.

Our research shows promise that models form a internal representation of syntax and grammar that performs well above random baselines. We propose more research could be done on the role of punctuation and other NLP models, as these might help towards a comprehension of compositionality in sequence to sequence translation tasks.

References

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. *CoRR*, abs/1506.04834.

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

Brenden M. Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *CoRR*, abs/1711.00350.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sara Veldhoen, Dieuwke Hupkes, Willem H Zuidema, et al. 2016. Diagnostic classifiers revealing how neural networks process hierarchical structure. In *CoCo@ NIPS*.

A Additional figures

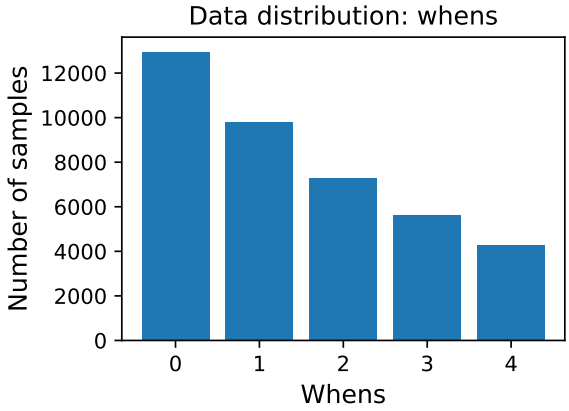

Figure 4: Distribution of the number of *when*s per sample in the data that was used.

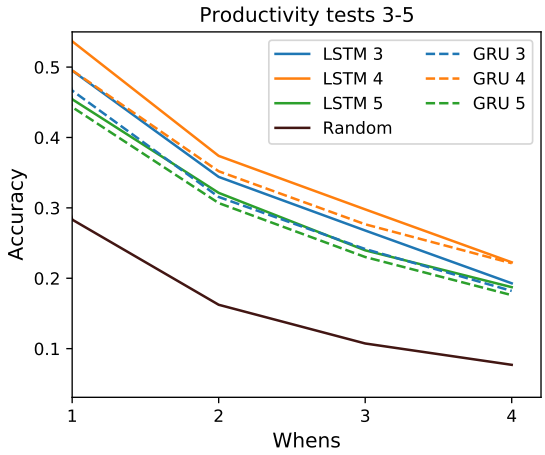

Figure 5: An alternative visualization of the bottom row of Figure 2 in order to better distinguish differences between the tests. The differences are small, and possibly mostly related to the size of the training sets.

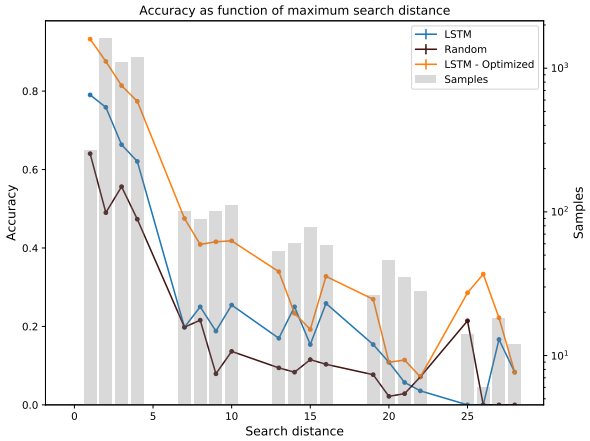

Figure 6: An overview of performance over search distance for LSTM, comparing the results baseline to the single run optimized network.

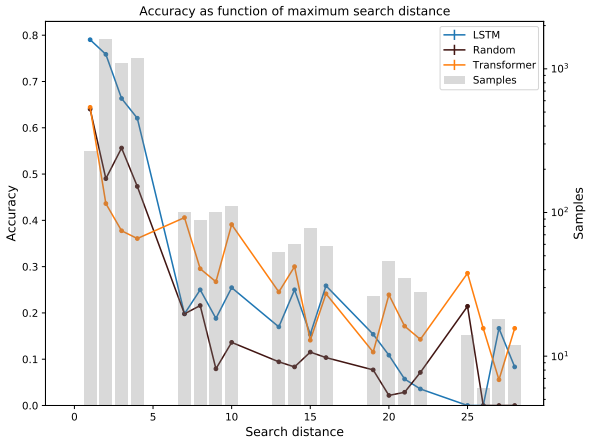

Figure 7: As an indication of the potential of a Transformer trained on a bigger dataset, we can see it too can perform above random baselines on longer search distances. The data is however too inconsistent to include as a valid result and requires more research.