Evaluating morphological generalisation in machine translation by distribution-based compositionality assessment

Anonymous Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Anonymouser Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Anonymousest Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

Compositional generalisation refers to the ability to understand and generate an infinite number of novel meanings using a finite group of known primitives and a set of rules of how to combine them. The degree to which artificial neural networks can possess this ability is an open question. Recently, many evaluation methods and benchmarks have been proposed to test compositional generalisation, but not many have focused on the morphological level of language. We propose an application of the previously developed *distribution-based compositionality assessment* method to assess compositional generalisation on the level of morphology in NLP tasks, such as machine translation or paraphrase detection. We demonstrate the use of our method by comparing the morphological generalisation ability of translation models with different BPE vocabulary sizes. The evaluation method we propose suggests that small vocabularies help with morphological generalisation in NMT.[1]

1 Introduction

Natural languages usually adhere to the *principle of compositionality*, with the exception of idiomatic expressions. Partee et al. (1995) phrased this principle as "The meaning of a whole is a function of the meanings of the parts and of the way they are syntactically combined". Deriving from this principle, *compositional generalisation* (CG) refers to the capacity to understand and generate an infinite number of novel meanings using a finite group of known primitives and a set of rules of how to combine them. In the case of language, morphemes are combined into words and words in turn into phrases and sentences, using the syntactical rules of the language.

Neural networks have long been argued to lack the ability to generalise compositionally the way humans do (Fodor and Pylyshyn, 1988; Marcus, 1998). After the rapid improvement of neural NLP systems during the previous decade, this question has gained renewed interest. Many new evaluation methods have been developed to assess whether the modern sequence-to-sequence (seq2seq) architectures such as Transformers exhibit CG, since they certainly exhibit increasingly competent linguistic behaviour. For instance, in one of the seminal CG evaluation methods, called SCAN (Lake and Baroni, 2018), a seq2seq system has seen certain natural language commands in training and needs to combine them in novel ways in testing.

CG is a general capacity that can be seen as a desideratum in many NLP tasks, and in machine learning more generally. Furthermore, CG is a multifaceted concept that can be, and should be, decomposed into narrower, more manageable aspects that can be tested separately (Hupkes et al., 2020). For example, NLP systems should be able to generalise compositionally both on the level of words and on the level of morphology.

Although many aspects of CG have recently been evaluated in NLP (an extensive review is offered by Hupkes et al. (2022)), some aspects have remained without an evaluation method. We identify (see Section 2) a lack of methods to evaluate *compositional morphological* generalisation using only *natural*, non-synthetic, data. To fill this gap, we propose an application of the *distribution-based compositionality assessment* (DBCA) method (Keysers et al., 2020) (henceforth *Keysers*) to generate adversarial data splits to evaluate morphological generalisation in NLP systems.

Specifically, we split natural language corpora

[1] A link to the Github repository anonymised.

while controlling the distributions of lemmas and morphological features (*atoms* in the terminology of *Keysers*) on the one hand, and the distributions of the combinations of atoms (*compounds*, not to be confused with compound words) on the other hand. By requiring a low divergence between the atom distributions of the train and test sets, and a high divergence between the compound distributions, we can evaluate how well a system is able to generalise its morphological knowledge to unseen word forms.

For example, if our corpus included as atoms the lemmas "cat" and "dog", and the morphological tags `Number=Sing` and `Number=Plur`, a low divergence between the atom distributions would mean that both the training and test sets included all four of the atoms, and a high compound divergence would mean that the sets include different combinations of them, for instance training set {`cat, dogs`} and test set {`cats, dog`}.

Our main contributions are the following: **firstly**, we describe an application of DBCA to evaluate morphological generalisation in any NLP task in which the train and test data consist of sentences for which morphological tags are available. **Secondly**, we demonstrate how by this method we can evaluate morphological generalisation in machine translation without manual test design. And **thirdly**, using our proposed method, we assess the effect of the source language BPE (Sennrich et al., 2016) vocabulary size in Finnish-English NMT performance, and conclude that a smaller vocabulary helps the NMT models in morphological generalisation.

2 Background

In the broader field of machine learning, CG has been analysed in various domains besides that of natural language, such as visual question answering (Bahdanau et al., 2018), visual reasoning (Zerroug et al., 2022) and mathematics (Saxton et al., 2019), but in this work we focus on natural language tasks. Two reviews have recently been published about CG in NLP, of which Donatelli and Koller (2023) focus on semantic parsing and the aforementioned Hupkes et al. (2022) (henceforth *Hupkes*) take a broader view, reviewing generalisation in general, not only the compositional type.

Hupkes categorised NLP generalisation experiments along five dimensions, of which we discuss two here to motivate our work. The first is the *type of generalisation* along which the *compositional* type is distinguished from the *morphological* type. *Hupkes* define compositionality as "the ability to systematically recombine previously learned elements to map new inputs made up from these elements to their correct output. In language, the inputs are 'forms' (e.g. phrases, sentences, larger pieces of discourse), and the output that they need to be mapped to is their meaning ...". In NMT, the translation works as a proxy to meaning, so that CG can be evaluated by evaluating the translation (Dankers et al., 2022) (other works that assess CG in NMT include (Li et al., 2021; Raunak et al., 2019)).

Hupkes contrast compositional with structural, including morphological, generalisation where an output space is not required but which focuses on generation of the correct forms. These definitions suggest a clear divide between the categories, which is understandable when analysing the literature: morphological generalisation, specifically inflection generation, has for decades been studied in psycholinguistics (Berko, 1958; Marcus et al., 1992) and computational linguistics (Rumelhart and McClelland, 1986; Corkery et al., 2019; Kodner et al., 2022). These studies do not address the question of how the different inflections are mapped to different meanings, hence they do not address *compositional* generalisation. However, inflections do bear meaning, of course, and so *compositional morphological* generalisation is an ability that humans possess, and NLP systems ought to be tested on.

Although *Hupkes* do not categorise any experiments as assessing *compositional morphological* generalisation, there has been at least one that we think could be so categorised: Burlot and Yvon (2017) designed an NMT test suite in which a single morphological feature is modified in a source language sentence, creating a contrastive pair, and the translations of the contrastive sentences are inspected for a corresponding change in the target language.

The other dimension of *Hupkes* relevant to the motivation of our experiments is that of *shift source*: the shift between train and test sets could occur naturally (as in two natural corpora in different domains), it can be created by generating synthetic data, or an artificial partition of natural data can be obtained. Most of the previous methods to assess compositional generalisation in NMT (Bur-

lot and Yvon, 2017; Li et al., 2021; Dankers et al., 2022) have synthetised data for the test sets. Generating synthetic data has its benefits: any morphological form can occur in the data when it is generated, and a single morphological feature can be easily focused on and evaluated qualitatively as well as quantitatively.

However, synthetic data has at least practical disadvantages, leaving aside the more theoretical question of how well the synthetic language approximates natural language, assuming the ultimate goal is systems that process natural language. In practice, synthetic test sets require manual design, which means it is difficult to come by a method to generate an unlimited number of synthetic sentences, or a method that could work in arbitrary languages. Furthermore, when manually designing test suites to evaluate morphological generalisation, as Burlot and Yvon (2017) designed, the requirement for manual work restricts the number of morphological phenomena we have resources to test.

The other option is to create artificial data splits of natural data. While natural data may be noisier and it might be more difficult to focus on a specific phenomenon of the language by this method, this method is easier to automate completely. Furthermore, the method of automatically generating data splits that we present in the next section is also generalisable to other tasks (e.g. paraphrase detection) and any corpus of sentences. Generating artificial data splits of natural data has previously been used to test CG in translation (Raunak et al., 2019), but not for assessing *morphological* generalisation, as far as we are aware. (For a more general discussion of splitting data into non-random testing and training sets, see Søgaard et al. (2021).)

The method we describe in this paper is an application of the DBCA method developed by *Keysers*. Since this method is generic and task-agnostic, it can be applied to any dataset for which it is possible to define atom and compound distributions. Although it is easier to define these distributions for synthetic data, as in the CFQ dataset described by *Keysers*, it can also be applied to natural data, for example in semantic parsing (Shaw et al., 2021). The next section describes how DBCA can be used to assess morphological generalisation in any task where the training and testing corpora consist of natural language sentences.

3 Applying DBCA to assess morphological generalisation in NLP

DBCA is a method to evaluate CG by splitting a dataset into train/test sets with differing distributions, requiring some capacity to generalise from the training distribution to the test distribution. Specifically, the distributions of *atoms* (known primitives) and *compounds* (combinations of atoms) are controlled to get similar atom distributions but contrasting compound distributions in the training and test sets. In our application of DBCA to a corpus of natural language sentences, the atom distribution $\mathcal{F}_A$ of the corpus is the distribution of the lemmas and morphological features and the compound distribution $\mathcal{F}_C$ is the distribution of their combinations. Table 1 presents examples of atoms and compounds in this work.

To determine the atom and compound distributions, we first need to obtain the lemmas and morphological tags of all words in the corpus, which we accomplish for Finnish corpora using the Turku Neural Parser Pipeline (Kanerva et al., 2018). For the experiments presented in Section 4, we use a corpus of 1M sentences. In practice, we do not have resources to control the distribution of all lemmas even in this relatively small corpus, so we need to select some subset of the lemmas that we include in our analysis.

Selecting the lemma subset could be done in many ways, but the following is a way we deemed reasonable. To limit the number of lemmas, we first filter out lemmas that do not appear in the list of 94110 Finnish lemmas[2] or, since this list does not include proper names, in lists[3] of names for places, or lists of Finnish and English given names. This way, the lemmas that are filtered out include most of the typos and other non-words. Then we rank the remaining lemmas by frequency in our corpus, and sample a fixed number of lemma occurrences from constant intervals in the ranked list of lemmas. Specifically, we take 40000 lemma occurrences at intervals of 1000 lemma types in the list of lemmas. For our corpus of 1M sentences, this method subsamples

[2]Available at `https://kaino.kotus.fi/sanat/nykysuomi/`

[3]List of names of places: `https://kaino.kotus.fi/eksonyymit/?a=aineisto`
English given names: `https://en.wiktionary.org/wiki/Appendix:English_given_names`
and Finnish: `https://tinyurl.com/3mn52ms6`
`https://tinyurl.com/mwjvaxkk`

	Atoms	Compounds
Description	lemmas and morphological tags	combinations of atoms
Examples	`tunturi`, `Case=Gen`, `Case=Ade`, `Number=Sing`, `Number=Plur`	`tunturi|Case=Gen|Number=Plur` ("tunturien"), `tunturi|Case=Ade|Number=Sing` ("tunturilla")

Table 1: Description and examples of what we call "atoms" and "compounds". The compounds are the unique word forms, determined by the lemma and the morphological tags. The word form is written inside the brackets.

the lemmas with frequency ranks of 1000-1033, 2000-2083, 3000-3174, and so on, so that there are fewer frequent lemma types than rare lemma types, but the total number of occurrences of each bucket is around 40k. Lemmas that occur fewer than 10 times in the corpus are excluded. After the filtering, we have 8720 lemma types that occur about 390k times in total in our corpus of 1M sentences. We append the list of 48 morphological tags[4] (after filtering some that indicate uninteresting words such as 'Typo' and 'Abbr') that these lemmas appear with to the lemma list to complete our list of atoms.

Keysers weighted the compounds to "avoid double-counting compounds that are highly correlated with some of their super-compounds". The idea is to lessen the weight of those compounds that only or often occur as a part of one certain super-compound. We weight the compounds analogously, but use only two levels in our weighting, which makes the weighting simpler than in *Keysers*: we consider the combinations of morphological tags as the lower level of compounds, and these combined with lemmas as the higher level. Thus the motivation for weighting in our case is not to use those morphological tag combinations that only occur with some specific lemma. Therefore, we look for the lemma with which each morphological tag combination occurs most often, and give the tag combination a weight that is the complement of the empirical probability that the tag combination occurs with this lemma. For example, we found that the rare morph tag combination `Case=Ade | Degree=Pos | Number=Plur | PartForm=Pres | VerbForm=Part | Voice=Pass` occurs 84% of the time with the lemma `saada` forming the word "saatavilla", so it gets a weight of 0.16. After weighting the tag combinations, we exclude those that have a weight of 0.33 or less.

[4]See `https://universaldependencies.org/docs/fi/feat/` for the list of Finnish morphological tags.

After the described filtering steps, we have 8322 atoms, which includes the lemmas and morphological tags. The atoms occur about 1.3M times in 273k sentences in our corpus of 1M sentences. There are 335 morphological tag combinations that these lemmas appear with, which create about 69k unique word forms with the lemmas; i.e. there are 69k compounds that we use in our analysis. These compounds occur 352k times in the corpus, in 273k sentences.

Calculating atom and compound divergences is done the same way as in *Keysers*. Namely, divergence $\mathcal{F}$ between distributions P and Q is calculated using the Chernoff coefficient $C_\alpha(P\|Q) = \sum_k p_k^\alpha \, q_k^{1-\alpha} \in [0,1]$ (Chung et al., 1989), with $\alpha = 0.5$ for the atom divergence and $\alpha = 0.1$ for the compound divergence. As described by *Keysers*, $\alpha = 0.5$ for the atom divergence "reflects the desire of making the atom distributions in train and test as similar as possible", and $\alpha = 0.1$ for the compound divergence "reflects the intuition that it is more important whether a certain compound occurs in P (train) than whether the probabilities in P (train) and Q (test) match exactly". Since the Chernoff coefficient is a similarity metric, the atom and compound divergences of a train set V and a test set W are:

$$\mathcal{D}_A(V\|W) = 1 - C_{0.5}(\mathcal{F}_A(V) \,\|\, \mathcal{F}_A(W))$$
$$\mathcal{D}_C(V\|W) = 1 - C_{0.1}(\mathcal{F}_C(V) \,\|\, \mathcal{F}_C(W)).$$

Once the divergences are defined, we can split a corpus of natural language sentences into training and testing sets with an arbitrary compound and atom divergence values. For this, we use a simple greedy algorithm, sketched in Algorithm 1. For a maximum compound divergence split, the score is calculated as

$$\text{score}(Q,P) = \mathcal{D}_C(Q\|P) - \mathcal{D}_A(Q\|P),$$

and in general, for any desired compound divergence value c:

$$\text{score}(Q,P) = -|c - \mathcal{D}_C(Q\|P)| - \mathcal{D}_A(Q\|P).$$

Procedure 1 Data division algorithm.

Input: G ▷ Corpus of sentences
Input: N ▷ Use N sentences from G
Input: a ▷ Lower bound for $|V|/|W|$
Input: b ▷ Upper bound for $|V|/|W|$
Output: V, W ▷ Train set, test set
  $V \leftarrow \{x \in_R G\}$ ▷ A random sentence
  $W \leftarrow \emptyset$
  $G \leftarrow G \backslash V$
  for $i \leftarrow 1$ **to** N **do**
    $r \leftarrow |V|/|W|$
    $s_V \leftarrow \max_{x \in G} \text{score}(V \cup \{x\}, W)$
    $i_V \leftarrow \text{argmax}_{x \in G} \text{score}(V \cup \{x\}, W)$
    $s_W \leftarrow \max_{x \in G} \text{score}(V, W \cup \{x\})$
    $i_W \leftarrow \text{argmax}_{x \in G} \text{score}(V, W \cup \{x\})$
    if $(s_V > s_W \wedge r < b) \vee r < a$ **then**
      $V \leftarrow V \cup \{i_V\}$
      $G \leftarrow G \backslash \{i_V\}$
    else
      $W \leftarrow W \cup \{i_W\}$
      $G \leftarrow G \backslash \{i_W\}$
    end if
  end for

In practice, we do not have resources to calculate the $\max_{x \in G} \text{score}$. Instead, at each iteration we take a subset $G' \subset G$, say 1000 sentences, and calculate $\max_{x \in G'} \text{score}$.

As mentioned above, this method can be used for any corpus that consists of natural language sentences for which the morphological tags can be obtained. In the next section we use this method to assess morphological generalisation in machine translation.

4 Experiments and results

4.1 NMT model training setup and data

We chose Finnish as the language we analyse because of its rich morphology and because there is a good morphological tagger available for Finnish. We use the English-Finnish parallel corpus from the Tatoeba challenge data release (Tiedemann, 2020). We first apply some heuristics provided by Aulamo et al. (2020) to remove noisy data, and restrict the maximum sentence length to 100 words, after which we take a random sample of 1 million sentence pairs.

We use the OpenNMT-py (Klein et al., 2017) library to train Finnish-English Transformer NMT models using the hyperparameters provided in the example config file[5], which includes the standard 6 transformer layers with 8 heads and a hidden dimension of 512, as in (Vaswani et al., 2017). We train the models until convergence or until a maximum of 33000 steps with 2000 warm-up steps and a batch size of 4096 tokens.

For more details about the setup, see the Github repository linked on the first page.

4.2 The effect of compound divergence on translation performance

The basic experiment we propose is to make at least two different train/test splits of a corpus, using $\mathcal{D}_C$ values of 0 and 1, respectively, (keeping $\mathcal{D}_A = 0$) and assess the change in translation performance (for which we use BLEU (Papineni et al., 2002) and chrF2++ (Popović, 2017) as metrics). Since with $\mathcal{D}_C = 1$ there are more unseen word forms in the test set, we expect a decrease in translation performance from $\mathcal{D}_C = 0$ to $\mathcal{D}_C = 1$ that is caused by the $\mathcal{D}_C = 1$ test set requiring more morphological generalisation capacity.

We show empirically the decrease in performance in Section 4.3, but the cause of this decrease is of course more difficult to verify exactly. The atom and compound distributions are the only things we explicitly control when splitting the corpus, and we only require the compound divergence to differ between different data splits. Therefore, we assume the differing compound divergence to be the cause of this effect, but to be more certain, we conduct two simple checks to look for confounding factors.

Firstly, an increase in the average sentence length could be another factor that makes one test set more difficult than another. Increasing the sequence length from training to test set is actually a method that has been proposed to test a certain type of compositional generalisation, sometimes called *productivity* (Hupkes et al., 2020; Raunak et al., 2019). We calculated the average sentence lengths of the train and test sets of the 8 different data splits that we obtained using 8 different random seeds for the data split algorithm. What we found is that for $\mathcal{D}_C = 1$ the average lengths in test sets are actually shorter (ranging from 11.35 to 11.66 words) than those for $\mathcal{D}_C = 0$ (ranging from 12.27 to 13.72 words). The average training set sentence lengths are similar for both $\mathcal{D}_C$ values,

[5] https://github.com/OpenNMT/OpenNMT-py/blob/master/config/config-transformer-base-1GPU.yml

ranging from 8.66 to 8.79 for $\mathcal{D}_C = 0$ and from 8.65 to 8.73 for $\mathcal{D}_C = 1$. Thus we know that an increased difference between train and test set sentence lengths cannot explain the decrease in NMT performance from $\mathcal{D}_C = 0$ to $\mathcal{D}_C = 1$ since the difference is actually larger for $\mathcal{D}_C = 0$. The fact that the average sentence length in training sets is always significantly shorter than in test sets is an interesting unintended artefact of the data division algorithm that deserves further investigation in the future, but it does not confound our analysis.

As the second sanity check, we evaluated the NMT models on a neutral test set to see if, for any reason, the training set would be in general worse with $\mathcal{D}_C = 1$ than with $\mathcal{D}_C = 0$, instead of only being worse for the specific test set that we have created. For this we used the Tatoeba challenge test set, which we did not use to train or tune the hyperparameters of any models. The results for the vocabulary size 1000 are presented in Figure 1. We used the models trained on the training sets from the data splits with compound divergences 0.0, 0.5 and 1.0. The compound divergences between these training sets and the Tatoeba challenge test set do correlate with the target $\mathcal{D}_C$ of the data split, but they range only from about 0.4 to 0.6.

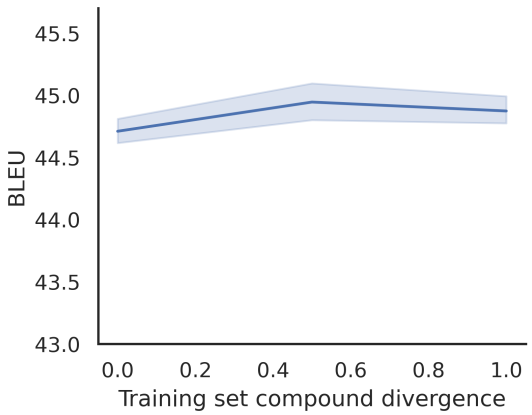

Figure 1: Results on the Tatoeba challenge test set. The x-axis labels denote the compound divergences between the training sets and the test sets analysed later in Figure 2. That is, the divergence is not between the training sets and the Tatoeba challenge test set.

From Figure 1 we can see that the NMT models trained with different data sets, from data splits with different $\mathcal{D}_C$ values, do not show similar decrease in performance on the neutral-ish Tatoeba challenge test set as on the test sets obtained from the data split algorithm. We take this to mean that the models trained on $\mathcal{D}_C = 1$ data splits are not in general worse than those trained with $\mathcal{D}_C = 0$ data splits, but only worse on the high-divergence test set.

4.3 The effect of BPE vocabulary size on morphological generalisation in NMT

Next, we make the assumption, based on the analysis in Section 4.2, that we can measure morphological generalisation by measuring the decrease of NMT performance between train/test splits of $\mathcal{D}_C = 0$ and $\mathcal{D}_C = 1$. Previous studies have suggested the hypothesis that NMT models with smaller BPE vocabularies are more capable of modelling morphological phenomena than those with larger vocabularies (for example Libovický and Fraser (2020)). In this section, we compare the morphological generalisation capacities of NMT models with different source-side (Finnish) vocabulary sizes, using the method we have proposed.

As a preliminary experiment, we tuned the BPE vocabulary size for our setup (see Section 4.1) on the Tatoeba challenge development set, and found the optimal size to be around 3000 BPE tokens for both the source and target languages. Since we are interested in the Finnish morphology, next we kept the target (English) vocabulary size constant and varied only the source-side vocabulary size.

We chose 7 different vocabulary sizes, 3 larger and 3 smaller than the optimal 3000, and evaluated them with target compound divergence values of 0.0, 0.25, 0.5, 0.75 and 1.0. The sizes of the test sets are in the order of a few tens of thousands, or a little over a hundred thousand, sentences. The relatively large test set size leads to statistical significance even for small BLEU differences (see Table 3 for details).

From the BLEU results for $\mathcal{D}_C = 0$ and $\mathcal{D}_C = 1$ in Figure 2 we can see that as we either increase or decrease the vocab size from 3000, the performance drops, but it drops slightly differently w.r.t $\mathcal{D}_C$. This effect is most conspicuous for the pair of sizes 500 and 18000. The larger vocabulary performs slightly better when there is less need for morphological generalisation, but the small vocabulary performs better when it is needed more. In general, from this figure we can see that the vocabulary size roughly correlates with the angle of

the downward slope, suggesting that the larger the vocabulary, the poorer the capacity for morphological generalisation.

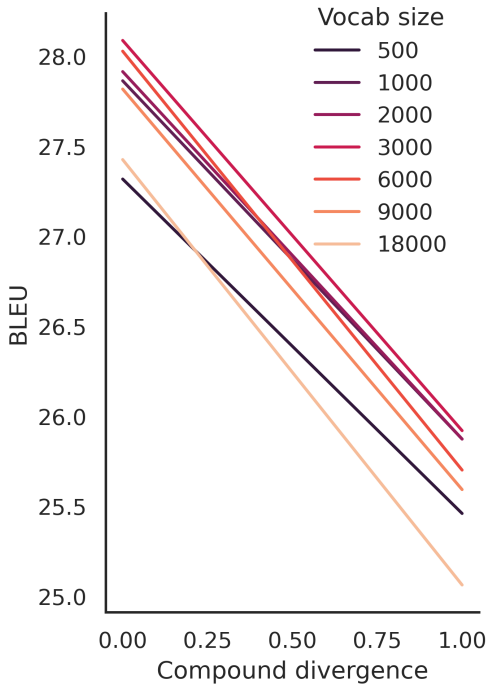

Figure 2: Different source vocabulary sizes evaluated with minimum and maximum (0 and 1) compound divergence data splits. Compound divergence value 1 requires more morphological generalisation. The larger the vocabulary the steeper the slope, suggesting poorer ability to generalise. For more details, see Table 3 in Appendix A.

To investigate the effect of the initialisation of the data split algorithm on the results, we split the same corpus starting from 8 different random initialisations, and trained NMT models for each data split. For this, we chose two pairs of vocabulary sizes that showed most clearly contrasting performance w.r.t $\mathcal{D}_C$: 500&18000 and 1000&6000. The main results are presented in Table 2. For these results, the test sets of the 8 random seeds are concatenated together to create exceptionally large test sets of around 400k-500k sentences. The results for the individual data splits are presented in Appendix A in Table 4.

From these results we can see the same contrasting performance of the small and large vocabularies w.r.t the different compound divergence values. The difference is small but statistically significant and consistent. The models with small vocabularies show better performance than those with large ones when morphological generalisation is needed, and vice versa when morphological generalisation is not needed as much.

5 Discussion and future work

In Section 3, we proposed an application of DBCA to divide any corpus of sentences, for which morphological tags are available, into training and test sets with similar distributions of lemmas and morphological tags but contrasting distributions of word forms, in order to assess morphological generalisation. By this method, we can take a large proportion of the morphological phenomena of a selected language into consideration, in our experiments 335 different morphological categories that together with about 8k lemmas create 69k unique Finnish word forms, and evaluate the effects of the contrasting train/test distributions of the word forms in machine translation. This enables a different, complementing type of assessment of morphological generalisation than previous synthetic benchmarks (mainly Burlot and Yvon (2017)) that focus on a smaller number of morphological phenomena. The benefit of our method is the comprehensiveness, focusing on the corpus-wide distributions of word forms.

Using only corpus-wide metrics such as BLEU, as we used, does not allow for the qualitative evaluation that the synthetic benchmarks offer. In the terminology of Burlot and Yvon (2017), this *holistic*, document-level evaluation can be contrasted with *analytic* evaluation that focuses more specifically on difficulties in morphology. A trick that could enable a more analytic assessment of the translations of the unseen word forms would be to align the words in the source sentences with the words in the reference translations and the words in the predicted translations, and evaluate only the translations of the parts of the sentences that correspond to the unseen word forms. Similar method has been used previously for example by Bau et al. (2019); Stanovsky et al. (2019), and we aim to experiment with this method in future work.

Especially combined with this word-alignment trick, we could also make our evaluation more *fine-grained* (this concept also from (Burlot and Yvon, 2017)), that is, our evaluation could differentiate between different types of mistakes. Since we have the morphological tags, we could sort the words by morphological category and compare the translation accuracies to look for any especially

	chrF2++		BLEU	
Vocab	$\mathcal{D}_C = 0$	$\mathcal{D}_C = 1$	$\mathcal{D}_C = 0$	$\mathcal{D}_C = 1$
500	51.20 (51.20 ± 0.05)	**49.33** (49.33 ± 0.05)	27.50 (27.50 ± 0.07)	**25.4** (25.40 ± 0.07)
18000	**51.29** (51.29 ± 0.05)	49.04 (49.05 ± 0.05)	**27.69** (27.69 ± 0.07)	25.18 (25.18 ± 0.07)
	$p = 0.0003$	$p = 0.0003$	$p = 0.0003$	$p = 0.0003$
1000	51.78 (51.78 ± 0.05)	**49.79** (49.79 ± 0.05)	28.17 (28.17 ± 0.07)	**25.89** (25.89 ± 0.07)
6000	**51.83** (51.83 ± 0.05)	49.67 (49.67 ± 0.05)	**28.24** (28.24 ± 0.07)	25.80 (25.80 ± 0.07)
	$p = 0.0003$	$p = 0.0003$	$p = 0.0003$	$p = 0.0003$

Table 2: Pairwise comparisons of the source vocabulary sizes 500 and 18000; 1000 and 6000. The results are calculated for the concatenated test sets generated with 8 random seeds. Inside brackets is the true mean estimated from bootstrap resampling and the 95% confidence interval. The results for the individual seeds are presented in Appendix A in Table 4 and Figure 3.

difficult categories for the translation models.

To demonstrate the use of our proposed method, we compared NMT models with different BPE vocabulary sizes, since vocabulary size had been hypothesised to affect the capacity to model morphology in translation. Besides vocabulary size, there are many other model design choices that have been proposed to help either in generalisation or in capturing morphological phenomena. For example, factored NMT systems (García-Martínez et al., 2016) can cover more of the target side vocabulary than subword-based NMT systems, which can help in modelling the morphology of the target language. It would be interesting to assess different model types such as factored NMT systems or LSTM-based systems to see how they compare with Transformers on our evaluation method.

The DBCA method is general, and could be applied to a wide variety of tasks and datasets. Our application of DBCA is more specific, but it still inherits some of the generality of the original method. Our method is directly applicable to any machine learning task where the data consists of sentences for which the morphological tags are available. In the future, we intend to extend our assessment of morphological generalisation to other languages, as well as to other NLP tasks, such as paraphrase detection.

6 Conclusion

We proposed a method to assess morphological generalisation by distribution-based compositionality assessment. Because this method is fully automated, it enables more comprehensive assessment of morphological generalisation than previously proposed synthetic benchmarks, in terms of the number of inflection types we can evaluate. We used our method to assess NMT models with different BPE vocabulary sizes and found that models with smaller vocabularies are better at morphological generalisation than those with larger vocabularies. Lastly, we discussed the varied future directions that our generalisable method offers, such as assessing morphological generalisation in other NLP tasks besides NMT.

References

Mikko Aulamo, Sami Virpioja, and Jörg Tiedemann. 2020. OpusFilter: A configurable parallel corpus filtering toolbox. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 150–156. Association for Computational Linguistics.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2018. Systematic generalization: What is required and can it be learned? In *International Conference on Learning Representations*.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jean Berko. 1958. The child's learning of english morphology. *Word*, 14(2-3):150–177.

Franck Burlot and François Yvon. 2017. Evaluating the morphological competence of machine translation systems. In *Proceedings of the Second Conference on Machine Translation*, pages 43–55.

JK Chung, PL Kannappan, CT Ng, and PK Sahoo. 1989. Measures of distance between probability distributions. *Journal of mathematical analysis and applications*, 138(1):280–292.

Maria Corkery, Yevgen Matusevych, and Sharon Goldwater. 2019. Are we there yet? encoder-decoder neural networks as cognitive models of english past tense inflection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3868–3877.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175.

Lucia Donatelli and Alexander Koller. 2023. Compositionality in computational linguistics. *Annual Review of Linguistics*, 9.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation architectures. In *Proceedings of the 13th International Conference on Spoken Language Translation*.

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.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2022. State-of-the-art generalisation research in NLP: a taxonomy and review. *CoRR*.

Jenna Kanerva, Filip Ginter, Niko Miekka, Akseli Leino, and Tapio Salakoski. 2018. Turku neural parser pipeline: An end-to-end system for the conll 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Jordan Kodner, Salam Khalifa, Khuyagbaatar Batsuren, Hossep Dolatian, Ryan Cotterell, Faruk Akkus, Antonios Anastasopoulos, Taras Andrushko, Aryaman Arora, Nona Atanalov, et al. 2022. Sigmorphon–unimorph 2022 shared task 0: Generalization and typologically diverse morphological inflection. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 176–203.

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.

Yafu Li, Yongjing Yin, Yulong Chen, and Yue Zhang. 2021. On compositional generalization of neural machine translation. 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 4767–4780.

Jindřich Libovickỳ and Alexander Fraser. 2020. Towards reasonably-sized character-level transformer nmt by finetuning subword systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2572–2579.

Gary F Marcus. 1998. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282.

Gary F. Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T. John Rosen, Fei Xu, and Harald Clahsen. 1992. Overregularization in language acquisition. *Monographs of the Society for Research in Child Development*, 57(4):i–178.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Barbara Partee et al. 1995. Lexical semantics and compositionality. *An invitation to cognitive science*, 1:311–360.

Maja Popović. 2017. chrf++: words helping character n-grams. In *Proceedings of the second conference on machine translation*, pages 612–618.

Vikas Raunak, Vaibhav Kumar, and Florian Metze. 2019. On compositionality in neural machine translation. *arXiv preprint arXiv:1911.01497*.

David E Rumelhart and James L McClelland. 1986. On learning the past tenses of english verbs.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *ArXiv*, abs/1904.01557.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

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.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1823–1832.

Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. 2019. Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684, Florence, Italy. Association for Computational Linguistics.

Jörg Tiedemann. 2020. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Aimen Zerroug, Mohit Vaishnav, Julien Colin, Sebastian Musslick, and Thomas Serre. 2022. A benchmark for compositional visual reasoning. *arXiv preprint arXiv:2206.05379*.

A Detailed results

Vocab size	BLEU per Compound divergence				
	0.00	0.25	0.50	0.75	1.00
500	27.3182 (27.3207 ± 0.1694)	26.5671 (26.5661 ± 0.1970)	25.3562 (25.3538 ± 0.1735)	24.7654 (24.7629 ± 0.1839)	25.4613 (25.4627 ± 0.1744)
1000	27.8639 (27.8654 ± 0.1826)	27.3292 (27.3268 ± 0.2033)	25.8704 (25.8681 ± 0.1773)	25.5556 (25.5516 ± 0.1824)	25.8745 (25.8747 ± 0.1751)
2000	27.9148 (27.9170 ± 0.1798)	27.5814 (27.5795 ± 0.2033)	26.0737 (26.0715 ± 0.1756)	25.5343 (25.5310 ± 0.1846)	25.8743 (25.8753 ± 0.1746)
3000	28.0879 (28.0894 ± 0.1825)	27.5439 (27.5422 ± 0.2038)	25.9751 (25.9734 ± 0.1739)	25.6909 (25.6888 ± 0.1842)	25.9213 (25.9236 ± 0.1793)
6000	28.0282 (28.0299 ± 0.1828)	27.3659 (27.3625 ± 0.2013)	25.9772 (25.9752 ± 0.1801)	25.4389 (25.4360 ± 0.1858)	25.7023 (25.7032 ± 0.1722)
9000	27.8178 (27.8196 ± 0.1850)	27.2639 (27.2612 ± 0.2072)	25.7347 (25.7331 ± 0.1708)	25.3648 (25.3612 ± 0.1857)	25.5939 (25.5947 ± 0.1790)
18000	27.4262 (27.4281 ± 0.1829)	26.8142 (26.8115 ± 0.2053)	25.3575 (25.3540 ± 0.1744)	24.7417 (24.7397 ± 0.1873)	25.0643 (25.0649 ± 0.1742)
	chrF2++ per Compound divergence				
500	51.0092 (51.0110 ± 0.1375)	50.5787 (50.5782 ± 0.1611)	49.7497 (49.7488 ± 0.1425)	49.2403 (49.2380 ± 0.1564)	49.1858 (49.1856 ± 0.1385)
1000	51.5254 (51.5268 ± 0.1395)	51.3332 (51.3315 ± 0.1625)	50.3049 (50.3037 ± 0.1403)	49.9848 (49.9819 ± 0.1507)	49.5935 (49.5925 ± 0.1429)
2000	51.5363 (51.5383 ± 0.1392)	51.5222 (51.5204 ± 0.1635)	50.3994 (50.3982 ± 0.1448)	49.9124 (49.9093 ± 0.1536)	49.6822 (49.6818 ± 0.1421)
3000	51.6843 (51.6853 ± 0.1378)	51.4699 (51.4687 ± 0.1622)	50.4017 (50.4011 ± 0.1409)	50.0425 (50.0401 ± 0.1538)	49.6165 (49.6170 ± 0.1411)
6000	51.6609 (51.6628 ± 0.1375)	51.3303 (51.3282 ± 0.1614)	50.3249 (50.3242 ± 0.1413)	49.792 (49.7897 ± 0.1551)	49.4806 (49.4801 ± 0.1417)
9000	51.3687 (51.3704 ± 0.1383)	51.09 (51.0879 ± 0.1625)	50.0669 (50.0659 ± 0.1414)	49.7751 (49.7725 ± 0.1557)	49.3615 (49.3612 ± 0.1396)
18000	51.0249 (51.0266 ± 0.1391)	50.7775 (50.7761 ± 0.1618)	49.7367 (49.7352 ± 0.1428)	49.2308 (49.2288 ± 0.1518)	48.7807 (48.7799 ± 0.1448)

Table 3: The BLEU and chrF2++ results for the different source-side (Finnish) BPE vocabulary sizes and different compound divergence values. Inside brackets is the true mean estimated from bootstrap resampling and the 95% confidence interval.

		chrF2++		BLEU	
Seed	Vocab	$\mathcal{D}_C = 0$	$\mathcal{D}_C = 1$	$\mathcal{D}_C = 0$	$\mathcal{D}_C = 1$
11	500	51.01 (51.01 ± 0.14)	49.19 (49.19 ± 0.14)	27.32 (27.32 ± 0.17)	25.46 (25.46 ± 0.17)
	18000	51.02 (51.03 ± 0.14)	48.78 (48.78 ± 0.14)	27.43 (27.43 ± 0.18)	25.06 (25.06 ± 0.17)
		p = 0.2439	p = 0.0003	p = 0.0243	p = 0.0003
22	500	51.01 (51.01 ± 0.14)	49.08 (49.08 ± 0.15)	27.3 (27.3 ± 0.18)	25.2 (25.2 ± 0.18)
	18000	50.85 (50.85 ± 0.14)	49.05 (49.05 ± 0.15)	27.17 (27.17 ± 0.18)	25.1 (25.1 ± 0.18)
		p = 0.0003	p = 0.1913	p = 0.0107	p = 0.053
33	500	51.07 (51.07 ± 0.14)	49.37 (49.37 ± 0.17)	27.37 (27.37 ± 0.18)	25.09 (25.09 ± 0.2)
	18000	50.97 (50.97 ± 0.14)	49.04 (49.04 ± 0.17)	27.3 (27.3 ± 0.18)	24.83 (24.83 ± 0.2)
		p = 0.0047	p = 0.0003	p = 0.092	p = 0.0003
44	500	52.02 (52.02 ± 0.17)	49.7 (49.7 ± 0.18)	28.3 (28.3 ± 0.21)	25.8 (25.8 ± 0.22)
	18000	52.44 (52.44 ± 0.17)	49.43 (49.43 ± 0.17)	28.72 (28.72 ± 0.21)	25.63 (25.63 ± 0.22)
		p = 0.0003	p = 0.0003	p = 0.0003	p = 0.0077
55	500	52.33 (52.34 ± 0.18)	49.34 (49.34 ± 0.16)	29.04 (29.04 ± 0.23)	25.29 (25.29 ± 0.2)
	18000	52.76 (52.76 ± 0.18)	49.04 (49.04 ± 0.16)	29.58 (29.58 ± 0.24)	25.08 (25.08 ± 0.2)
		p = 0.0003	p = 0.0003	p = 0.0003	p = 0.001
66	500	50.98 (50.98 ± 0.14)	49.24 (49.24 ± 0.14)	27.12 (27.12 ± 0.18)	25.31 (25.31 ± 0.18)
	18000	51.06 (51.06 ± 0.14)	48.87 (48.87 ± 0.14)	27.4 (27.4 ± 0.18)	25.04 (25.04 ± 0.17)
		p = 0.0183	p = 0.0003	p = 0.0003	p = 0.0003
77	500	50.84 (50.83 ± 0.14)	49.46 (49.46 ± 0.14)	27.12 (27.12 ± 0.18)	25.41 (25.4 ± 0.16)
	18000	50.68 (50.68 ± 0.14)	49.22 (49.22 ± 0.14)	27.06 (27.06 ± 0.18)	25.25 (25.25 ± 0.17)
		p = 0.0007	p = 0.0003	p = 0.1186	p = 0.0023
88	500	50.97 (50.97 ± 0.14)	49.38 (49.38 ± 0.14)	27.22 (27.22 ± 0.18)	25.61 (25.61 ± 0.17)
	18000	51.37 (51.37 ± 0.14)	49.05 (49.05 ± 0.14)	27.81 (27.81 ± 0.18)	25.43 (25.43 ± 0.18)
		p = 0.0003	p = 0.0003	p = 0.0003	p = 0.0003
11	1000	51.53 (51.53 ± 0.14)	49.59 (49.59 ± 0.14)	27.86 (27.87 ± 0.18)	25.87 (25.87 ± 0.18)
	6000	51.66 (51.66 ± 0.14)	49.48 (49.48 ± 0.14)	28.03 (28.03 ± 0.18)	25.7 (25.7 ± 0.17)
		p = 0.0003	p = 0.0017	p = 0.0013	p = 0.001
22	1000	51.46 (51.46 ± 0.14)	49.64 (49.64 ± 0.15)	27.9 (27.9 ± 0.18)	25.69 (25.69 ± 0.18)
	6000	51.47 (51.47 ± 0.14)	49.61 (49.61 ± 0.15)	27.94 (27.94 ± 0.19)	25.64 (25.64 ± 0.18)
		p = 0.3059	p = 0.1786	p = 0.1519	p = 0.1383
33	1000	51.59 (51.59 ± 0.14)	49.7 (49.7 ± 0.17)	27.89 (27.88 ± 0.18)	25.45 (25.45 ± 0.2)
	6000	51.63 (51.63 ± 0.14)	49.67 (49.68 ± 0.17)	28.02 (28.02 ± 0.18)	25.51 (25.51 ± 0.21)
		p = 0.117	p = 0.2073	p = 0.0047	p = 0.1276
44	1000	52.67 (52.67 ± 0.16)	50.32 (50.32 ± 0.17)	29.01 (29.01 ± 0.21)	26.53 (26.53 ± 0.22)
	6000	52.68 (52.68 ± 0.16)	50.06 (50.06 ± 0.18)	29.01 (29.01 ± 0.22)	26.33 (26.33 ± 0.22)
		p = 0.2809	p = 0.0003	p = 0.3949	p = 0.0037
55	1000	52.8 (52.8 ± 0.18)	49.92 (49.92 ± 0.16)	29.66 (29.66 ± 0.24)	25.92 (25.92 ± 0.2)
	6000	53.02 (53.03 ± 0.18)	49.72 (49.73 ± 0.16)	29.84 (29.85 ± 0.24)	25.73 (25.73 ± 0.2)
		p = 0.0003	p = 0.0003	p = 0.0017	p = 0.0003
66	1000	51.39 (51.39 ± 0.14)	49.57 (49.57 ± 0.14)	27.64 (27.64 ± 0.18)	25.71 (25.71 ± 0.18)
	6000	51.5 (51.49 ± 0.14)	49.37 (49.37 ± 0.14)	27.79 (27.79 ± 0.19)	25.54 (25.54 ± 0.18)
		p = 0.0013	p = 0.0003	p = 0.0017	p = 0.0017
77	1000	51.51 (51.51 ± 0.15)	49.8 (49.8 ± 0.13)	27.86 (27.86 ± 0.18)	25.84 (25.84 ± 0.17)
	6000	51.76 (51.76 ± 0.14)	49.74 (49.74 ± 0.14)	28.09 (28.09 ± 0.19)	25.74 (25.74 ± 0.17)
		p = 0.0003	p = 0.0453	p = 0.0003	p = 0.022
88	1000	51.9 (51.9 ± 0.14)	49.95 (49.95 ± 0.14)	28.29 (28.29 ± 0.18)	26.2 (26.2 ± 0.18)
	6000	51.6 (51.6 ± 0.14)	49.84 (49.84 ± 0.14)	28.01 (28.01 ± 0.18)	26.23 (26.23 ± 0.18)
		p = 0.0003	p = 0.0007	p = 0.0003	p = 0.2209

Table 4: Pairwise comparisons of the source vocabulary sizes 500 and 18000; 1000 and 6000 on the minimum and maximum compound divergence data splits. For 8 data split algorithm random seeds. Inside brackets is the true mean estimated from bootstrap resampling and the 95% confidence interval.

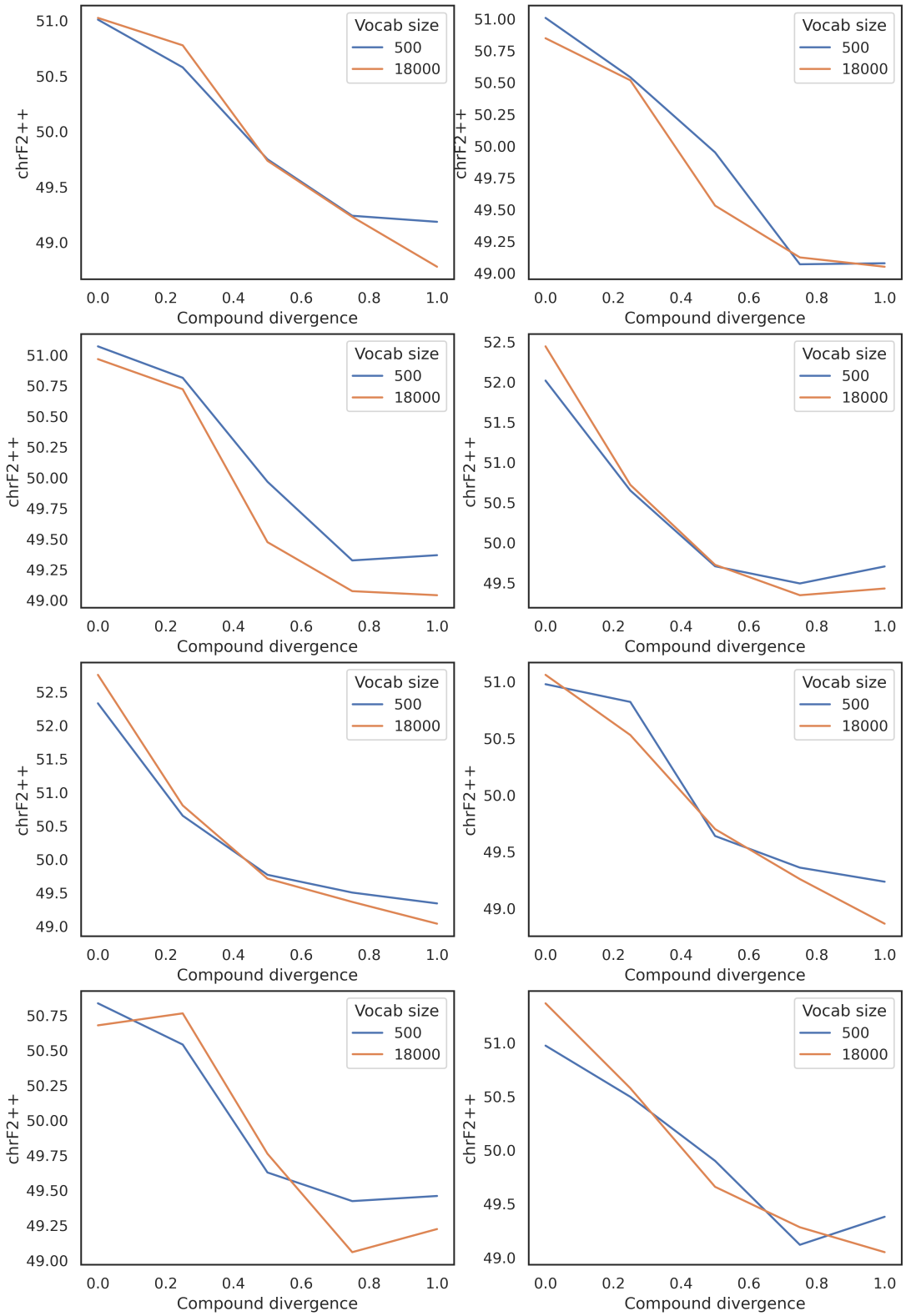

Figure 3: Comparison of vocabulary sizes 500 and 18000 with compound divergence values 0.0, 0.25, 0.5, 0.75 and 1.0. The same results are partly in Table 4.