Grammatical information in BERT sentence embeddings as two-dimensional arrays

Vivi Nastase and **Paola Merlo**
Department of Linguistics
University of Geneva
Paola.Merlo@unige.ch, vivi.a.nastase@gmail.com

Abstract

Sentence embeddings induced with various transformer architectures encode much semantic and syntactic information in a distributed manner in a one-dimensional array. We investigate whether specific grammatical information can be accessed in these distributed representations. Using data from a task developed to test rule-like generalizations, our experiments on detecting subject-verb agreement yield several promising results. First, we show that while the usual sentence representations encoded as one-dimensional arrays do not easily support extraction of rule-like regularities, a two-dimensional reshaping of these vectors allows various learning architectures to access such information. Next, we show that various architectures can detect patterns in these two-dimensional reshaped sentence embeddings and successfully learn a model based on smaller amounts of simpler training data, which performs well on more complex test data. This indicates that current sentence embeddings contain information that is regularly distributed, and which can be captured when the embeddings are reshaped into higher dimensional arrays. Our results cast light on representations produced by language models and help move towards developing few-shot learning approaches.

1 Introduction

Transformer-based models have taken the NLP world, and not only, by storm in recent years. They have even reached super-human performance on standard benchmarks such as SuperGLUE (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2018), and the output of GPT-* and chatGPT are often worryingly difficult to distinguish from human-produced data (Marcus, 2022; Susnjak, 2022). Considering such performance, the expectations are high that the word and sentence representations produced by transformer-based architectures can also be useful for finer-grained tasks, such as those that target specific grammatical phenomena.

Long-distance agreement, a specific and simple grammatical phenomenon, is often used to test the syntactic abilities of deep neural networks (Linzen et al., 2016; Gulordava et al., 2018; Goldberg, 2019; Linzen and Baroni, 2021). Long-distance agreement tests are usually framed as a prediction task: whether the model gives higher probability to the correct form of the verb, despite intervening *attractors* – nouns appearing between the subject and the verb. This decision is somehow implicit: the language model gives a prediction based on the data it has been built on.

In this paper, we investigate what happens when we target subject-verb agreement explicitly in BERT sentence embeddings for detecting subject-verb agreement in French sentences. We work with raw BERT sentence embeddings – not fine-tuned for specific tasks – and investigate how this specific information is encoded in the distributed representations. More specifically, we ask: Can we detect subject-verb agreement in sentence embeddings? And can we manipulate the raw sentence embeddings to make this phenomenon more easy to detect? To address these questions, we adopt the framework and task definition for rule-like learning described in Merlo et al. (2021). In this framework, learning a given linguistic property or rule is formulated as a task of implicit rule detection in a multiple-choice setup, based on input sequences that share the target property. For the problem under investigation here, each sequence consists of sentences that share subject-verb agreement, but have different distances between the subject and the verb, different clause structures, and different agreement patterns.

We show that BERT sentence embeddings encoded as a one-dimensional array are only successful at detecting subject-verb agreement when provided with large amounts of data. Reshaping the embeddings to two-dimensional arrays, and combining these with VAE-based architectures, allows

a system to detect better the shared patterns in the input sequences, while relying on much smaller amounts of simpler data. These results open up new avenues of exploration for few-shot learning (Fei-Fei et al., 2006; Brown et al., 2020; Schick and Schütze, 2021). They also support further analyses of more disentangled representations, those representations that encode underlying rule-like generalisations, typical of human knowledge representation, but not of neural networks (Sablé-Meyer et al., 2021). The contributions of this paper are:

1. We show that that there are higher-dimension patterns that encode syntactic phenomena in BERT sentence embeddings, beyond the one-dimensional array representation that is readily available from the transformer output.

2. We show that, used together with VAE-based architectures, two-dimensional reshapings of these representations facilitate the discovery of patterns that encode specific targeted grammatical phenomena.

3. We show that, through the 2D-ed representations, we get better access to encoded patterns, and can detect better the targeted grammatical phenomenon when training with a smaller amount of simpler data.

The code and the data are available here: `https://github.com/CLCL-Geneva/BLM-SNFDisentangling`.

Terminology Sentence embeddings can be read from the output of BERT systems as a $1 \times N$ vector (N usually 768 or 1024). This can be viewed as the projection of the sentence into an N-dimensional space. In this paper, we use the word *dimensions* to refer to the shape of the data structure used to represent the sentence embeddings. In particular, we use *one-dimensional array* to refer to the $1 \times N$ vector sentence representation obtained directly from BERT, and *2D representations* to refer to the 2D reshaped ($Rows \times Columns$) array.

2 Related work

Producing sentence representations is a non-trivial issue, mainly because of the structural grammatical and semantic relations they express and their varying complexity and length (Stevenson and Merlo, 2022). The deep learning framework has allowed for a variety of elegant solutions to explicitly learn sentence representations or to induce them as a side-effect or modeling of a more complex problem (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017; Peters et al., 2018). Transformer architectures, such as BERT (Devlin et al., 2019), have provided one such solution, where the representation of an input text as a one-dimensional array (usually 1x768 or 1x1024 for the large versions of the model) can be readily obtained from the output of the model. Depending how the system is trained, the sentence embedding can be obtained from the encoding of the [CLS] token, or as a combination of the embeddings of the tokens in the sentence.

Sentence transformers (Reimers and Gurevych, 2019) implement architecture and training regime changes on BERT to optimize sentence embeddings for downstream tasks. Nikolaev and Padó (2023) analyze the relation between specific sentence properties (e.g. the contribution of different POS) and the geometry of the embedding space of sentence transformers.

Whether obtained from sentence transformers or directly from the output of a BERT-based system, sentence embeddings have been shown to capture information about syntactic and semantic properties. For example, Manning et al. (2020) show that attention heads capture information about dependency relations in transformer models, and Thrush et al. (2020) show the BERT representations contain important information about argument structure and the meaning of verbs.

Subject-verb agreement is one of the phenomena used to probe a deep-learning system's syntactic abilities. While it is a simple word-level phenomenon, it encodes long-distance relations between words and requires knowledge of structural sentence properties to be correctly learned. Goldberg (2019) shows that sentence embeddings capture this property, by testing the language model learned by BERT in predicting the contextually appropriate form of the verb. Linzen and Baroni (2021) include an overview of work that analyzes deep-learning models on this task. While the models tested show high performance in predicting the contextually correct form of a verb, they are guided – and misled – by biases within the corpus on which they were trained, e.g. they pay undue attention to the first noun in the sentence. Linzen and Baroni also include a survey of work that probe deep learning models to understand how grammatical

information is encoded. Giulianelli et al. (2018); Conneau et al. (2018); McCoy et al. (2018) show that specific grammatical information – such as the plurality of the subject, the maximal depth of the parse tree of the sentence, the verb auxiliaries – can be decoded from the sentence encodings (or the hidden state) of the respective systems. Lakretz et al. (2021) analyze the actual architecture of an LSTM language model (Gulordava et al., 2018), and track the impact of each unit on the long-distance agreement performance. They uncover a combination of a sparse mechanism – two units – and a larger distributed circuit that together keep track of number and syntactic structure.

Lasri et al. (2022) focus on how BERT encodes grammatical number in English and how this information is used for performing number agreement. The focus is on word embeddings and quantifying how much number information they encode at various layers of the BERT architecture. Using a combination of probing approaches, they discover that subjects and predicates embeddings do encode number information, but at different layers. Further investigations into where and how the number information is shared reveals that number information is not directly shared, but rather passed through intermediate tokens.

We also target BERT embeddings to investigate the subject-verb agreement property. Rather than looking at properties of word/token embeddings, we analyze sentence embeddings as the embeddings of the special [CLS] token. We investigate how accessible the number agreement is in raw BERT sentence embeddings in several steps:

- test whether the subject-verb agreement rule can be recovered through the sentence representation
- test whether different shapes of the sentence embedding – 1D and various 2D forms – make the targeted rule more easy to find
- test these different shapes of sentence embeddings with several encode-decoder architectures, based on variational autoencoders (Kingma and Welling, 2013).

3 Sentence representations for detecting subject-verb agreement

3.1 Data

Specific grammatical phenomena are often studied on specifically designed or selected datasets (e.g.

	Contexts Template			
1	NP-sg	PP1-sg		VP-sg
2	NP-pl	PP1-sg		VP-pl
3	NP-sg	PP1-pl		VP-sg
4	NP-pl	PP1-pl		VP-pl
5	NP-sg	PP1-sg	PP2-sg	VP-sg
6	NP-pl	PP1-sg	PP2-sg	VP-pl
7	NP-sg	PP1-pl	PP2-sg	VP-sg
8	NP-pl	PP1-pl	PP2-sg	VP-pl

	Answer Set	
1	NP-sg PP1-sg et NP2 VP-sg	Coord
2	**NP-pl PP1-pl NP2-sg VP-pl**	correct
3	NP-sg PP1-sg VP-sg	WNA
4	NP-sg PP1-sg PP2-sg VP-pl	AE
5	NP-pl PP1-sg PP1-sg VP-pl	WN1
6	NP-pl PP1-pl PP2-pl VP-pl	WN2

Figure 1: BLM instances for verb-subject agreement, with two attractors. WNA=wrong nr. of attractors; AE=agreement error; WN1=wrong nr. for 1^{st} attractor (N1); WN2=wrong nr. for 2^{nd} attractor (N2).

	Example of contexts			
1	The vase	with the flower		leaks.
2	The vases	with the flower		leak.
3	The vase	with the flowers		leaks.
4	The vases	with the flowers		leak.
5	The vase	with the flower	from the garden	leaks.
6	The vases	with the flower	from the garden	leak.
7	The vase	with the flowers	from the garden	leaks.
8	???			

Example of answers	
The vase with the flower and the garden leaks.	Coord
The vases with the flowers from the garden leak.	Correct
The vase with the flower leaks.	WNA
The vase with the flower from the garden leak.	AE
The vases with the flower from the garden leak.	WN1
The vases with the flowers from the gardens leak.	WN2

Figure 2: Examples of actual sentences of type I data (original in French).

(Nikolaev and Padó, 2023; Linzen et al., 2016)). We use BLM-AgrF (An et al., 2023). The structure of each problem in this task and dataset is inspired from RPM visual pattern tests – Raven Progressive Matrices – where one problem consists of overlapping rules the solver must detect (Raven, 1938; Zhang et al., 2019). A Blackbird Language Matrix (BLM) problem (Merlo et al., 2021) for subject-verb agreement consists of a context set of seven sentences that share the subject-verb agreement phenomenon, but differ in other aspects – e.g. number of intervening attractors between the subject and the verb, different grammatical numbers for these attractors, and different clause structures. An example template is illustrated in Figure 1, and an actual example in Figure 2.

The dataset comprises three subsets, of increasing lexical complexity. Type I data is generated

based on manually provided seeds, and a template that captures the rules mentioned above. Type II data is generated based on Type I data, by introducing lexical variation with the aid of a transformer, by generating alternatives for masked nouns. Type III data is generated by combining sentences from different instances from the Type II data. This will allow us to investigate the impact of lexical variation on the ability of a system to detect grammatical patterns.

Each subset contains an equal number of instances comprising three clause structures (we include complete instances – in French – in Appendix A.1). These structural variations alter the distance and relative depth of the subject and verb to produce a variety of conditions, to allow us to investigate how the subject-verb agreement information is encoded in BERT sentence embeddings.

Each problem is paired with a set of candidate answers. To allow for probing the learned model, apart from the correct answer, the answer sets contain negative examples built by corrupting some of the generating rules. This helps investigate the kind of information and structure learned, and the type of mistakes a system is prone to.

Table 1 shows the data statistics. Each of the three subsets of datasets is split 90:10 into train and test subsets, which are provided with the data. We use 20% of the train data for development.

dataset	number of problems	train:test split
Type I	2304	90:10
Type II	38400	90:10
Type III	38400	90:10

Table 1: Data statistics. The different types of data reflect different amounts of lexical variation within a problem instance.

3.2 Sentence representations

We investigate BERT sentence representations in a series of architectures designed to test whether we can access the relevant information for subject-verb agreement detection. We obtain the sentence embedding from the last layer of BERT, as the embedding of the [CLS] special token.

Figure 3 shows the summary of the architectures explored. Details of the architecture parameters are in Appendix A.2.

To investigate the impact of 2D-ing the sentence embeddings, the input sequences are given as a

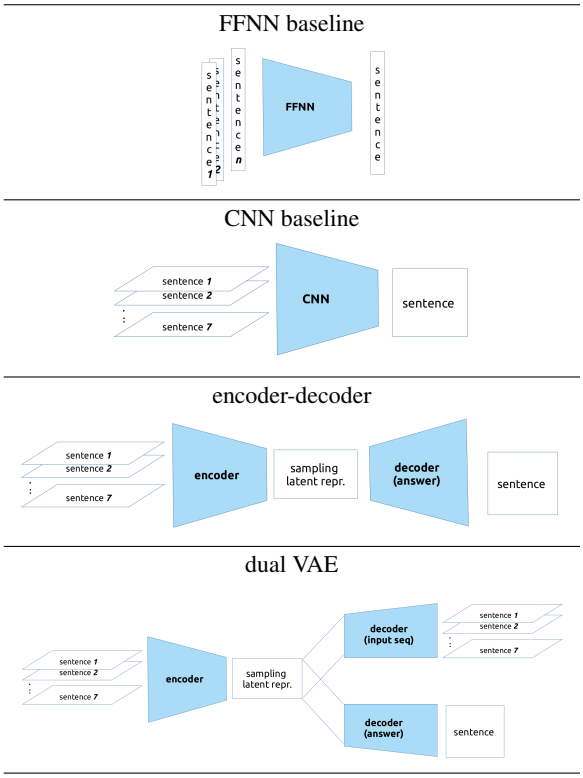

Figure 3: Architecture variations for exploring sentence embeddings

stack of 1D or 2D-ed sentence embeddings. This sequence of architectures also allows us to test the impact of additional abstracting steps – through compression into a latent VAE layer – for accessing patterns that encode the desired information.[1]

FFNN baseline The FFNN baseline is a three-layer feed-forward neural network. It transforms the sequence of the seven context sentence embeddings into a 1D-tensor, which is passed through three fully-connected layers, and outputs a vector that we take to represent the embedding of the answer sentence. This architecture allows the system to find patterns within and across sentences through the nodes in the successive layers.

The learning objective is to maximize the probability of the correct answer from the candidate answer set. Because the incorrect answers in the answer set are specifically designed to be minimally different from the correct answer, we implement the objective through the max-margin loss function. This function combines the scores of the correct and erroneous ones relative to the sentence embedding predicted by the system. We first compute a

[1]The code is available here: https://github.com/CLCL-Geneva/BLM-SNFDisentangling

score for the embedding e_i of each candidate answer a_i in the answer set $\mathcal{A}$ with respect to the predicted sentence embedding e_{pred} as the cosine of the angle between the respective vectors:

$$score(e_i, e_{pred}) = cos(e_i, e_{pred})$$

The loss uses the max-margin between the score for the correct answer e_c and for each of the incorrect answers e_i:

$$\mathcal{L}_a = \sum_{e_i} [1 - score(e_c, e_{pred}) + score(e_i, e_{pred})]^+$$

For prediction, the answer with the highest $score$ from a candidate set is taken as the correct answer.

CNN baseline The CNN baseline consists of N convolutional steps, followed by a linear layer to compress the output to the desired dimensions. The input consists of a stack of sentence representations. We use two variations of this architecture: (i) Baseline_CNN_1DxSeq: for a stack of 1D sentence representations, there are three 2D convolutional steps, which use kernels of size 3x3; (ii) Baseline_CNN_{NxM}: for a stack of (NxM) 2D-ed sentence representations, there is one 3D convolutional layer, with a kernel size 3x15x15. The size of the kernels was set after preliminary experiments. For both variations, the kernels allow the system to detect patterns within a sentence representation and across the sequence.

This system uses the same max-margin loss function as the FFNN baseline system.

Encoder-decoder This system is essentially a variational autoencoder (VAE) (Kingma and Welling, 2013; Kingma et al., 2015), but it does not reconstruct the input, rather it constructs an answer. For each of the input variations, the encoder consists of a similar architecture as the corresponding CNN baseline[2], but the output of the linear layer is the size L of the latent layer ($2 \times L$ to represent the mean and standard deviation for a vector of length L). A new vector (of size L) is sampled using the output of the encoder as the means and standard deviation of L normal distributions, using the reparametrization trick (Kingma et al., 2015). This vector is then unpacked through a decoder to produce a sentence representation. The architecture of the decoder mirrors as close as possible the architecture of the encoder. It differs in that it outputs a single sentence representation, and not the representation of a sequence of sentences. The two variations are named VAE_1DxSeq and VAE_{MxN} in the upcoming results tables.

The training objective is to maximize the probability of the correct answer, while improving the approximation of the posterior distribution on the latent layer. This is implemented through a loss function that combines the max-margin loss on the constructed answer (as for previous architectures), with an additional factor – the regularization factor on the latent layer, typical of VAE models:

$$\mathcal{L}_l = KL(q_{enc}(z|x) \parallel p(z))$$

where q_{enc} is the approximate posterior distribution of p, $p(z) = \mathcal{N}(0, 1)$ and $q_{enc}(z|x) = \mathcal{N}(\mu_x, \sigma_x)$, where the $[\mu_x; \sigma_x]$ is the latent vector output by the encoder for the input vector x.

The final loss function is:

$$\mathcal{L}_{enc-dec} = \mathcal{L}_a + \beta * \mathcal{L}_l$$

The β coefficient is used to push for disentanglement on the latent layer of a VAE (Higgins et al., 2016). For the reported experiments $\beta = 1$.

Dual VAE The dual VAE adds a decoder to reconstruct the input to the encoder-decoder system, which mirrors the encoder in architecture and parameters. The two variations are Dual_VAE_1DxSeq and Dual_VAE_{NxM}.

The training objective is to maximize both the probability of the answer and the reconstructed input, and improve the approximation of the posterior distribution on the latent layer. Essentially, we add a factor to the loss function of the encoder-decoder architecture, reflecting the reconstruction loss:

$$\mathcal{L}_{dVAE} = \mathcal{L}_a + \alpha * \mathcal{L}_{recon} + \beta * \mathcal{L}_l$$

where α is a coefficient that can control the contribution of the input reconstruction signal. For the experiments reported, $\alpha = 0.01$, and serves as a scaling factor, to bring the value of the reconstruction loss within the same magnitude as the answer loss $\mathcal{L}_a$. The reconstruction loss is computed as the mean-square error between the representation of the input sequence (x) and the reconstructed one (x'): $\mathcal{L}_{recon} = MSE(x, x')$

[2]Because the FFNN baseline performed very well, and this set-up provides a full receptive field, we also had an encoder-decoder variation using FFNN but it performed much worse than the other variations, and we do not report on it here.

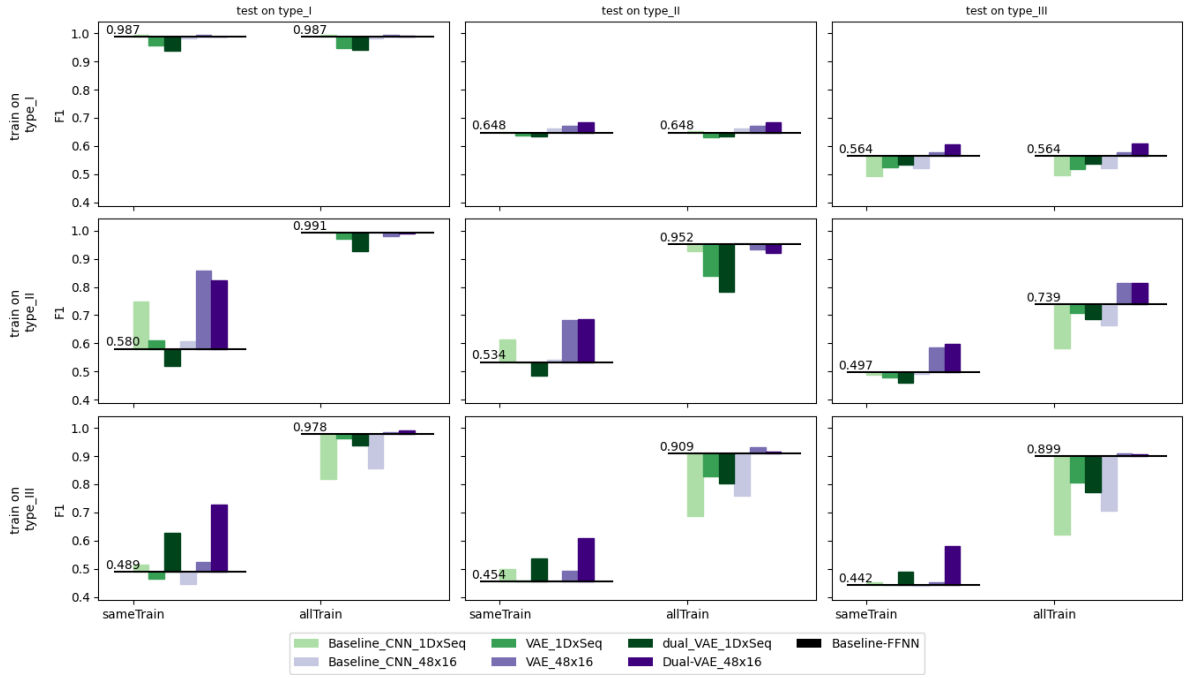

Figure 4: (best seen in color) F1 scores (averages over five runs) on the FFNN baseline and CNN, enc-dec and dual VAE systems with 48x16 2D-ed sentence embeddings. The graphs shows the difference in F1 between the systems relative to the reference baseline (FFNN). Each panel includes results on training on the same amount of data (left) and training on full data (right).

4 Experiments

We report experiments on seven systems – the baseline FFNN, and two variations (the input as a stack of 1D or 2D sentence representations) for each of the other three architectures.

Our aim was to explore raw BERT sentence embeddings, and not variations fine-tuned for specific tasks. The results reported here are based on sentence embeddings obtained using BERTTokenizer and BERTModel from the transformers Python library, using the pretrained "BERT-base-multilingual-cased" model[3].[4]

Preliminary experiments have been used to determine the kernel size for processing the 2D and 3D tensors with the stack of 1D and 2D-ed sentence embeddings respectively. The optimal kernel for 2D tensors was 3x3, and for the 3D tensors was 3x15x15. This large kernel size imposes specific restrictions on the dimensions of the 2D-ed sentence embeddings. Because a sentence embedding is a one-dimensional array of length 768, there are only 4 possible 2D values, where both dimensions are greater than 15 (16x48, 24x32, 32x24, 48x16).

We have also explored the size of the latent layer (5:25, step 5), and chosen the latent size 5. All systems used a learning rate of 0.001 and Adam optimizer, and batch size 100. For the baselines and experiments on the full training set for type II and type III data the training was done for 50 epochs, and for type I data and the sameTrain set-up, the training was done for 120 epochs.

The experiments were run on an HP PAIR Workstation Z4 G4 MT, with an Intel Xeon W-2255 processor, 64G RAM, and a MSI GeForce RTX 3090 VENTUS 3X OC 24G GDDR6X GPU.

4.1 Results

Figures 4 and 5 show the main experimental results. All results represent F1 averages over five runs. More details are provided in Appendix A.3.

Figure 4 shows the difference in F1 between the FFNN baseline (the black lines) and the increasingly complex architectures. Each row corresponds to the data type used for training, and the columns to the data type used for testing. Each panel shows the test results of models trained on the same amount of data, on the left, and training on full data, on the right.

[3] https://huggingface.co/bert-base-multilingual-cased

[4] We have run preliminary experiments with French-specific sentence embeddings using FlauBERT (Le et al., 2020). The results were lower than when using a multilingual cased BERT language model.

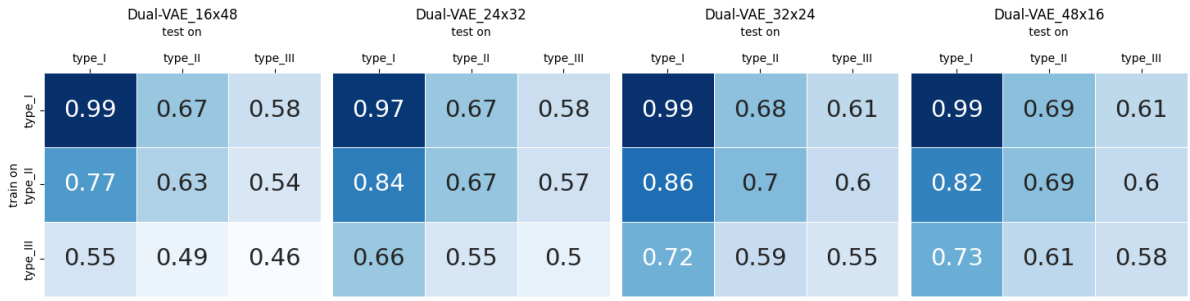

Figure 5: Impact of different reshapings: F1 results (averages over five runs) on (16x48), (24x32), (32x24), (48x16) reshaping, using the dual VAE architecture, and trained on similar amounts of training data.

Figure 5 shows the results obtained using different 2D transformations of the one-dimensional tensor BERT sentence embeddings with the dual VAE system, when training on the same amount of data (2073 instances – the amount available for type I data). Overall, the best-performing embedding is the one reshaped in a 48x16 matrix, the setting then used for the results reported in the system study, shown in Figure 4.

Figure 7 shows the impact of the amount of training data on the performance of the models. The results reported are average F1 scores over 5 runs, using the dual-VAE architecture with 48x16 sentence embedding.

4.2 Discussion

Impact of 2D-ed representations and VAE-based architectures In Figure 4, the horizontal black lines represent the performance of the Baseline FFNN system, and the bars show the relative performance of the system variations with 1D and 2D-ed sentence representations.

When using the full training data the results on type II and type III subsets are very high. This is in line with ML theory, as input with more variety leads to better-performing models, *when given enough training data*. The low results of the baselines and the systems using the 1D representations on the restricted training set-up (2073 instances – the available amount of training+validation data for type I – for all subsets) shows that these systems do not access the most relevant information from the sentence embedding for our targeted phenomenon.

Pairwise comparisons of similar architectures with different types of input show that 2D-ed representations lead to better results in almost all settings, particularly in the harsher training scenario with limited data (the left bar group in each plot).

The progression of architectures – from the CNN to the dual VAE – also show an increase in results, for both types of input representations. The phenomenon is more evident – and more useful – particularly for the restricted training scenario. It shows that forcing the representations to more compressed and abstract forms is useful for distilling the information useful to detect our targeted grammatical phenomenon.

The most interesting result is a combination of the impact of the 2D-ed representations and the various architectures: as the panels corresponding to training on type I data (first row in Figure 4) show, the combination of the dual VAE architecture with 2D-ed sentence embedding leads to the best results when testing on type II and type III data, which are lexically more complex than type I data. This shows that with a good combination of input representation and system, a model can find robust patterns even in a smaller amount of simple data.

Impact of 2D-ing the sentence representation The results presented in Figure 4 show that the 48x16 2D version of the sentence representation leads to better results than the 1x768 version. Figure 5 shows the impact of various 2D variations, when training the systems on the same amount of training+validation data. The results are obtained with the dual VAE architecture. The (overall) best performing representation from the 4 variations is the 48x16 version. In fact, for the $N \times M$ 2D-ing, the results are better the smaller M becomes. This indicates that information is somehow uniformly distributed in a BERT sentence embedding, within subsequences of length close to 16 – at least the information relevant to our particular subject-verb agreement task.

Impact of training data Figure 4 shows that the 2D-ed sentence representation combined with the dual VAE architecture leads to the best results, par-

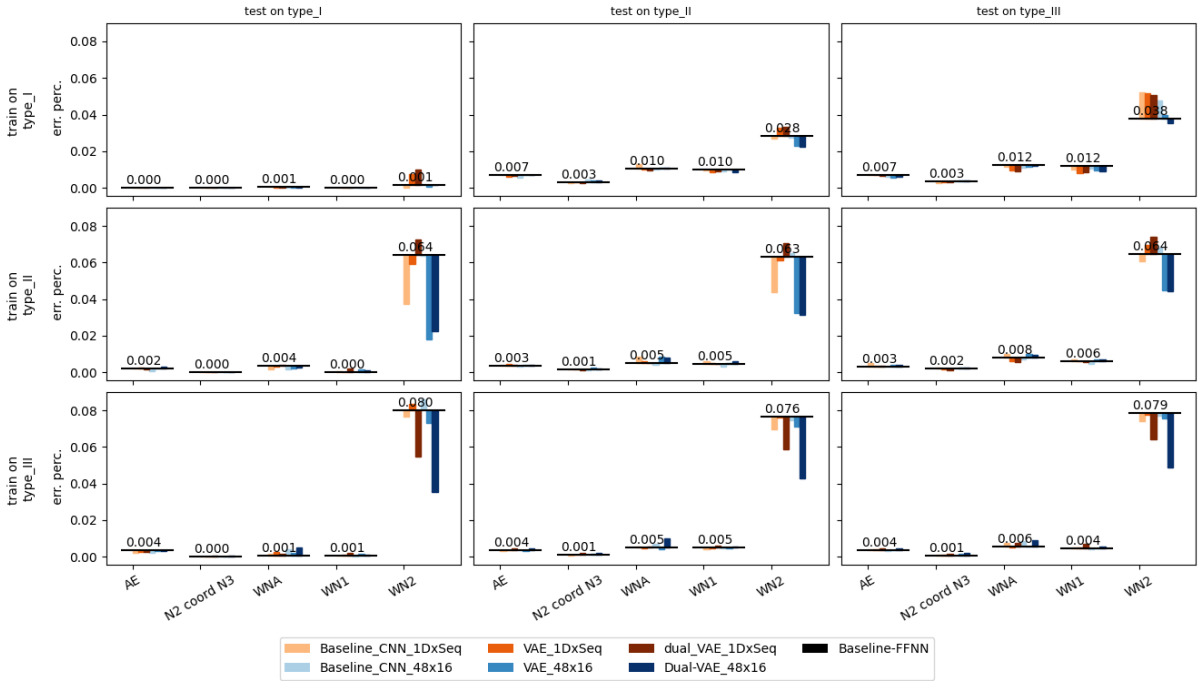

Figure 6: (best seen in color) Error analysis on the four systems trained with the same amount of data. The y-axis is the percentage of the error relative to the size of the test set (i.e. downward bars indicate an improvement.)

ticularly when training the systems on the same amount of training+validation data. We further analyze the learning curves when varying the amount of training+validation data from 50 to 2073 (split 80:20 into training and validation data). Figure 7 shows these results.

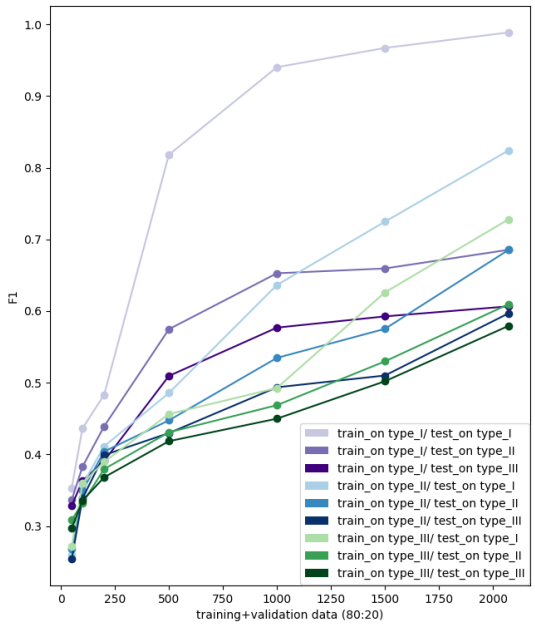

Figure 7: (best seen in color) Training data analysis using the the Dual_VAE_48x16 system

The curves corresponding to training on type I data – in shades of purple – approach the higher performance fastest, showing that they are able to exploit smaller amounts of data better. The lexical variations in type II and type III data seem to obfuscate the targeted patterns, as they require more training data.

4.3 Error Analyses

The error analysis shown in Figure 6 clearly shows that errors are almost always of the same kind — the assignment of the wrong grammatical number to the second attractor, closer to the verb— thus performing a local kind of agreement instead of the correct longer-distance structural agreement. These types of errors are frequent in humans too (Linzen and Leonard, 2018). It can be seen that the dual-VAE corrects these errors to a great degree (and greater than the other models), thus showing that it can detect the non-local patterns better than the other architectures. The combination of 2D-ing the sentence embedding and the large size of the kernel allows the system to find more distant patterns in the sentence embedding, thus connecting more distant tokens.

5 Conclusion

We have proposed and investigated variations of BERT sentence representations for the task of iden-

tifying the rule of subject-verb agreement and some of its properties in a dataset consisting of sequences of sentences that instantiate these underlying rules. BERT sentence embeddings, as 1D arrays, have been successfully used for a variety of tasks, but the information they encode about specific phenomena is distributed over the vector. Reshaping the sentence representation into two-dimensional inputs leads to improved results, and an additional step of abstracting these 2D-ed sentence embeddings leads to further improvements for the task. These more abstracted 2D-ed embeddings also learn a robust model based on a smaller amount of simpler training data, while showing good performance on the more lexically-complex data.

We plan to explore whether BERT sentence embedding may have even higher-dimension patterns, and whether such nD-ed BERT sentence representation can be used to detect other grammatical or semantic phenomena. We also plan to directly distill 2D sentence representation in more compact and disentangled representation, to encode more explicitly some of the distributed information in these embeddings.

Limitations

The experiments reported were performed on a dataset of French sentences, with a particular organization: sequences of sentences as input, each with a slightly different structure but sharing the subject-verb agreement rule. All sentences in the input sequence are processed together. In future work we plan to separate the distillation of rules from a sentence representation from the processing of the sequence.

We have investigated only part of the parameters in the proposed architectures. In particular, the β coefficient in the encoder-decoder and the dual VAE architectures was set to 1. Higher values may lead to more disentangled representations on the latent layer.

Ethics Statement

To the best of our knowledge, there are no ethics concerns with this paper.

Acknowledgments

We gratefully acknowledge the partial support of this work by the Swiss National Science Foundation, through grants #51NF40_180888 (NCCR Evolving Language) and SNF Advanced grant TMAG-1_209426 to PM.

References

Aixiu An, Chunyang Jiang, Maria A. Rodriguez, Vivi Nastase, and Paola Merlo. 2023. BLM-AgrF: A new French benchmark to investigate generalization of agreement in neural networks. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1363–1374, Dubrovnik, Croatia. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Fei-Fei, R. Fergus, and P. Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting*

Neural Networks for NLP, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1195–1205. Association for Computational Linguistics.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Diederik P Kingma, Tim Salimans, and Max Welling. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Yair Lakretz, Dieuwke Hupkes, Alessandra Vergallito, Marco Marelli, Marco Baroni, and Stanislas Dehaene. 2021. Mechanisms for handling nested dependencies in neural-network language models and humans. *Cognition*.

Karim Lasri, Tiago Pimentel, Alessandro Lenci, Thierry Poibeau, and Ryan Cotterell. 2022. Probing for the usage of grammatical number. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8818–8831, Dublin, Ireland. Association for Computational Linguistics.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. 2020. FlauBERT : des modèles de langue contextualisés pré-entraînés pour le français (FlauBERT : Unsupervised language model pre-training for French). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 2 : Traitement Automatique des Langues Naturelles*, pages 268–278, Nancy, France. ATALA et AFCP.

Tal Linzen and Marco Baroni. 2021. Syntactic structure from deep learning. *Annual Review of Linguistics*, 7(1):195–212.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association of Computational Linguistics*, 4(1):521–535.

Tal Linzen and Brian Leonard. 2018. Distinct patterns of syntactic agreement errors in recurrent networks and humans. In *Proceedings of the 40th Cognitive Science Society*.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117:30046 – 30054.

Gary Marcus. 2022. The dark risk of large language models. *WIRED*.

R Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: Hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society*.

Paola Merlo, Aixiu An, and Maria A. Rodriguez. 2021. Blackbird's language matrices (BLM): a new benchmark to investigate disentangled generalisation in neural networks.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Dmitry Nikolaev and Sebastian Padó. 2023. Representation biases in sentence transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3701–3716, Dubrovnik, Croatia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

John C. Raven. 1938. Standardization of progressive matrices. *British Journal of Medical Psychology*, 19:137–150.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Mathias Sablé-Meyer, Joël Fagot, Serge Caparos, Timo van Kerkoerle, Marie Amalric, and Stanislas Dehaene. 2021. Sensitivity to geometric shape regularity in humans and baboons: A putative signature of human singularity. *Proceedings of the National Academy of Sciences*, 118(16).

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Suzanne Stevenson and Paola Merlo. 2022. Beyond the benchmarks: Toward human-like lexical representations. *Frontiers of Artificial Intelligence*, 5:796741.

Teo Susnjak. 2022. ChatGPT: The end of online exam integrity?

Tristan Thrush, Ethan Wilcox, and Roger Levy. 2020. Investigating novel verb learning in BERT: Selectional preference classes and alternation-based syntactic generalization. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 265–275, Online. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. 2019. Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A.1 Type I instance examples

A Supplementary Materials

Main clause						
1		L'ordinateur	avec le programme			est en panne.
2		Les ordinateurs	avec le programme			sont en panne.
3		L'ordinateur	avec les programmes			est en panne.
4		Les ordinateurs	avec les programmes			sont en panne.
5		L'ordinateur	avec le programme	de l'expérience		est en panne.
6		Les ordinateurs	avec le programme	de l'expérience		sont en panne.
7		L'ordinateur	avec les programmes	de l'expérience		est en panne.
8		Les ordinateurs	avec les programmes	de l'expérience		sont en panne.
Completive clause						
1	Jean suppose que	l'ordinateur	avec le programme			est en panne.
2	Jean suppose que	les ordinateurs	avec le programme			sont en panne.
3	Jean suppose que	l'ordinateur	avec les programmes			est en panne.
4	Jean suppose que	les ordinateurs	avec les programmes			sont en panne.
5	Jean suppose que	l'ordinateur	avec le programme	de l'expérience		est en panne.
6	Jean suppose que	les ordinateurs	avec le programme	de l'expérience		sont en panne.
7	Jean suppose que	l'ordinateur	avec les programmes	de l'expérience		est en panne.
8	Jean suppose que	les ordinateurs	avec les programmes	de l'expérience		sont en panne.
Relative clause						
1		L'ordinateur	avec le programme		dont Jean se servait	est en panne.
2		Les ordinateurs	avec le programme		dont Jean se servait	sont en panne.
3		L'ordinateur	avec les programmes		dont Jean se servait	est en panne.
4		Les ordinateurs	avec les programmes		dont Jean se servait	sont en panne.
5		L'ordinateur	avec le programme	de l'expérience	dont Jean se servait	est en panne.
6		Les ordinateurs	avec le programme	de l'expérience	dont Jean se servait	sont en panne.
7		L'ordinateur	avec les programmes	de l'expérience	dont Jean se servait	est en panne.
8		Les ordinateurs	avec les programmes	de l'expérience	dont Jean se servait	sont en panne.

Answer set for problem constructed from lines 1-7 of the main clause sequence		
1	L'ordinateur avec le programme et l'experiénce est en panne.	N2 coord N3
2	**Les ordinateurs avec les programmes de l'experiénce sont en panne.**	correct
3	L'ordinateur avec le programme est en panne.	wrong number of attractors
4	L'ordinateur avec les programmes de l'experiénce sont en panne.	agreement error
5	Les ordinateurs avec le programme de l'experiénce sont en panne.	wrong nr. for 1^{st} attractor noun (N1)
6	Les ordinateurs avec les programmes des experiénces sont en panne.	wrong nr. for 2^{nd} attractor noun (N2)

Figure 8: BLM-AgrF instances for verb-subject agreement, with two attractors (programme, experiénce), and three clause structures. And candidate answer set for a problem constructed from lines 1-7 of the main clause sequence.

A.2 System architecture details

Baseline_FFNN

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
BaselineFFNN                             [100, 768]                --
--Linear: 1-1                            [100, 1536]               8,259,072
--Linear: 1-2                            [100, 1536]               2,360,832
--Linear: 1-3                            [100, 768]                1,180,416
==========================================================================================
Total params: 11,800,320
Trainable params: 11,800,320
Non-trainable params: 0
Total mult-adds (G): 1.18
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 3.07
Params size (MB): 47.20
Estimated Total Size (MB): 52.42
==========================================================================================
```

Baseline_CNN_1DxSeq: CNN with stack of 1D sentence embeddings

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
BaselineCNN_1DxSeq                       [100, 768]                --
--Conv2d: 1-1                            [100, 4, 5, 766]          40
--Conv2d: 1-2                            [100, 8, 3, 764]          296
--Conv2d: 1-3                            [100, 16, 1, 762]         1,168
--Linear: 1-4                            [100, 768]                9,364,224
==========================================================================================
Total params: 9,365,728
Trainable params: 9,365,728
Non-trainable params: 0
Total mult-adds (G): 1.11
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 37.29
Params size (MB): 37.46
Estimated Total Size (MB): 76.91
==========================================================================================
```

Baseline_CNN_48x16: CNN with 48x16 2D-ed sentence embedding

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
BaselineCNN                              [100, 7, 768]                --
--Conv3d: 1-1                            [100, 32, 5, 34, 2]       21,632
--Linear: 1-2                            [100, 768]                8,356,608
==========================================================================================
Total params: 8,378,240
Trainable params: 8,378,240
Non-trainable params: 0
Total mult-adds (G): 1.57
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 9.32
Params size (MB): 33.51
Estimated Total Size (MB): 44.98
==========================================================================================
```

VAE_1DxSeq: encoder-decoder with stack of 1D sentence embeddings

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
VariationalAutoencoder                   [100, 5]                  --
--Encoder: 1-1                           [100, 5]                  --
     --Conv2d: 2-1                       [100, 4, 5, 766]          40
     --Conv2d: 2-2                       [100, 8, 3, 764]          296
     --Conv2d: 2-3                       [100, 16, 1, 762]         1,168
     --Linear: 2-4                       [100, 10]                 121,930
--simpleSampling: 1-2                    [100, 5]                  --
--Decoder_answer: 1-3                    [100, 1, 1, 768]          --
     --Linear: 2-5                       [100, 12192]              73,152
     --ConvTranspose2d: 2-6              [100, 8, 1, 764]          392
     --ConvTranspose2d: 2-7              [100, 4, 1, 766]          100
     --ConvTranspose2d: 2-8              [100, 1, 1, 768]          13
==========================================================================================
Total params: 197,091
Trainable params: 197,091
Non-trainable params: 0
Total mult-adds (M): 230.28
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 54.40
Params size (MB): 0.79
Estimated Total Size (MB): 57.33
==========================================================================================
```

VAE_48x16: encoder-decoder with 48x16 2D-ed sentence embeddings

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
VariationalAutoencoder                   [100, 5]                  --
--Encoder: 1-1                           [100, 5]                  --
     --Conv3d: 2-1                       [100, 32, 5, 34, 2]       21,632
     --Linear: 2-2                       [100, 10]                 108,810
--simpleSampling: 1-2                    [100, 5]                  --
--Decoder_answer: 1-3                    [100, 1, 1, 48, 16]       --
     --Linear: 2-3                       [100, 2176]               13,056
     --ConvTranspose3d: 2-4              [100, 1, 1, 48, 16]       7,201
==========================================================================================
Total params: 150,699
Trainable params: 150,699
Non-trainable params: 0
Total mult-adds (G): 1.30
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 11.07
Params size (MB): 0.60
Estimated Total Size (MB): 13.82
==========================================================================================
```

Dual_VAE_1DxSeq: dual VAE with stack of 1D sentence embeddings

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
VariationalAutoencoder                   [100, 5]                  --
--Encoder: 1-1                           [100, 5]                  --
     --Conv2d: 2-1                       [100, 4, 5, 766]          40
     --Conv2d: 2-2                       [100, 8, 3, 764]          296
     --Conv2d: 2-3                       [100, 16, 1, 762]         1,168
     --Linear: 2-4                       [100, 10]                 121,930
--simpleSampling: 1-2                    [100, 5]                  --
--Decoder_mirror: 1-3                    [100, 1, 7, 768]          --
     --Linear: 2-5                       [100, 12192]              73,152
     --ConvTranspose2d: 2-6              [100, 8, 3, 764]          1,160
     --ConvTranspose2d: 2-7              [100, 4, 5, 766]          292
     --ConvTranspose2d: 2-8              [100, 1, 7, 768]          37
--Decoder_answer: 1-4                    [100, 1, 1, 768]          --
     --Linear: 2-9                       [100, 12192]              73,152
     --ConvTranspose2d: 2-10             [100, 8, 1, 764]          392
     --ConvTranspose2d: 2-11             [100, 4, 1, 766]          100
     --ConvTranspose2d: 2-12             [100, 1, 1, 768]          13
==========================================================================================
Total params: 271,732
Trainable params: 271,732
Non-trainable params: 0
Total mult-adds (M): 635.19
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 95.37
Params size (MB): 1.09
Estimated Total Size (MB): 98.61
==========================================================================================
```

Dual_Vae_48x16: dual VAE with 48x16 2D-ed sentence embeddings

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
VariationalAutoencoder                   [100, 5]                  --
--Encoder: 1-1                           [100, 5]                  --
     --Conv3d: 2-1                       [100, 32, 5, 34, 2]       21,632
     --Linear: 2-2                       [100, 10]                 108,810
--simpleSampling: 1-2                    [100, 5]                  --
--Decoder_mirror: 1-3                    [100, 1, 7, 48, 16]       --
     --Linear: 2-3                       [100, 10880]              65,280
     --ConvTranspose3d: 2-4              [100, 1, 7, 48, 16]       21,601
--Decoder_answer: 1-4                     [100, 1, 1, 48, 16]       --
     --Linear: 2-5                       [100, 2176]               13,056
     --ConvTranspose3d: 2-6              [100, 1, 1, 48, 16]       7,201
==========================================================================================
Total params: 237,580
Trainable params: 237,580
Non-trainable params: 0
Total mult-adds (G): 12.92
==========================================================================================
Input size (MB): 2.15
Forward/backward pass size (MB): 24.07
Params size (MB): 0.95
Estimated Total Size (MB): 27.17
==========================================================================================
```

A.3 Detailed experimental results

Results on 2D-ing BERt sentence embeddings The detailed version of the results in Figure 5

TRAIN ON	TEST ON	16X48 F1 (STD)	24X32 F1 (STD)	32X24 F1 (STD)	48X16 F1 (STD)
type I	type I	**0.9905** (0.0069)	0.9740 (0.0061)	0.9879 (0.0064)	0.9887 (0.0080)
type I	type II	0.6682 (0.0035)	0.6683 (0.0041)	0.6815 (0.0021)	**0.6855** (0.0028)
type I	type III	0.5795 (0.0049)	0.5819 (0.0011)	**0.6072** (0.0045)	0.6066 (0.0018)
type II	type I	0.7732 (0.0283)	0.8355 (0.0137)	**0.8632** (0.0089)	0.8242 (0.0065)
type II	type II	0.6333 (0.0132)	0.6725 (0.0077)	**0.6984** (0.0046)	0.6855 (0.0040)
type II	type III	0.5431 (0.0069)	0.5689 (0.0024)	0.5952 (0.0058)	**0.5969** (0.0046)
type III	type I	0.5550 (0.0461)	0.6649 (0.0059)	0.7221 (0.0228)	**0.7281** (0.0259)
type III	type II	0.4947 (0.0218)	0.5474 (0.0076)	0.5884 (0.0053)	**0.6096** (0.0064)
type III	type III	0.4620 (0.0151)	0.5042 (0.0089)	0.5515 (0.0050)	**0.5794** (0.0054)

Table 2: Analysis of 2D-ing the BERT sentence embeddings: F1 (std) scores for the four 2D combinations. The highest value for each train/test combination highlighted in bold.

Results on system analysis The detailed version of the results in Figure 4

TRAIN ON	TEST ON	BASELINE_FFNN	BASELINE_CNN 48X16	VAE_48X16	DUAL_VAE_48X16
TRAIN ON FULL TRAINING DATA					
type I	type I	0.9870 (0)	0.9827 (0)	**0.9957** (0)	0.9905 (0.0042)
type I	type II	0.6482 (0)	0.6612 (0)	0.6729 (0)	**0.6829** (0.0070)
type I	type III	0.5643 (0)	0.5229 (0)	0.5776 (0)	**0.6089** (0.0062)
type II	type I	**0.9913** (0)	**0.9913** (0)	0.9801 (0.0052)	0.9896 (0.0044)
type II	type II	0.9523 (0)	**0.9552** (0)	0.9331 (0.0014)	0.9215 (0.0017)
type II	type III	0.7391 (0)	0.6622 (0)	**0.8156** (0.0019)	0.8152 (0.0037)
type III	type I	0.9784 (0)	0.8571 (0)	0.9853 (0.0035)	**0.9913** (0.0027)
type III	type II	0.9086 (0)	0.7578 (0)	**0.9309** (0.0039)	0.9146 (0.0040)
type III	type III	0.8987 (0)	0.7062 (0)	**0.9089** (0.0016)	0.9047 (0.0046)
TRAIN ON THE SAME AMOUNT OF DATA (2073 INSTANCES: 1658 TRAIN/415 VALIDATION)					
type I	type I	0.9870 (0)	0.9827 (0)	**0.9957** (0)	0.9887 (0.0080)
type I	type II	0.6482 (0)	0.6612 (0)	0.6729 (0)	**0.6855** (0.0028)
type I	type III	0.5643 (0)	0.5229 (0)	0.5776 (0)	**0.6066** (0.0018)
type II	type I	0.5801 (0)	0.6061 (0)	**0.8571** (0)	0.8242 (0.0065)
type II	type II	0.5336 (0)	0.5411 (0)	0.6833 (0)	**0.6855** (0.0040)
type II	type III	0.4974 (0)	0.4901 (0)	0.5846 (0)	**0.5969** (0.0046)
type III	type I	0.4892 (0)	0.4459 (0)	0.5238 (0)	**0.7281** (0.0259)
type III	type II	0.4542 (0)	0.4576 (0)	0.4935 (0)	**0.6096** (0.0064)
type III	type III	0.4419 (0)	0.4380 (0)	0.4529 (0)	**0.5794** (0.0054)

Table 3: Analysis of systems: F1 (std) scores for the FFNN baseline and the 3 2D-ed system architectures. The highest value for each train/test combination is highlighted in bold.

TRAIN ON	TEST ON	BASELINE_CNN_1DXSEQ	VAE_1DXSEQ	DUAL_VAE_1DXSEQ
TRAIN ON FULL TRAINING DATA				
type I	type I	0.9948 (0.0064)	0.9489 (0.0088)	0.9403 (0.0084)
type I	type II	0.6518 (0.0044)	0.6305 (0.0069)	0.6347 (0.0026)
type I	type III	0.4961 (0.0043)	0.5194 (0.0086)	0.5357 (0.0037)
type II	type I	0.9974 (0.0035)	0.9697 (0.0039)	0.9264 (0.0152)
type II	type II	0.9256 (0.0041)	0.8393 (0.0016)	0.7819 (0.0111)
type II	type III	0.5837 (0.0064)	0.7060 (0.0154)	0.6853 (0.0107)
type III	type I	0.8173 (0.0390)	0.9628 (0.0065)	0.9385 (0.0050)
type III	type II	0.6865 (0.0191)	0.8279 (0.0060)	0.8020 (0.0020)
type III	type III	0.6205 (0.0180)	0.8046 (0.0038)	0.7727 (0.0036)
TRAIN ON THE SAME AMOUNT OF DATA (2073 INSTANCES: 1658 TRAIN/415 VALIDATION)				
type I	type I	0.9939 (0.0044)	0.9558 (0.0042)	0.9385 (0.0069)
type I	type II	0.6491 (0.0070)	0.6358 (0.0039)	0.6335 (0.0018)
type I	type III	0.4946 (0.0047)	0.5249 (0.0092)	0.5346 (0.0051)
type II	type I	0.7489 (0.0387)	0.6095 (0.0202)	0.5203 (0.0330)
type II	type II	0.6146 (0.0180)	0.5306 (0.0077)	0.4862 (0.0185)
type II	type III	0.4898 (0.0062)	0.4774 (0.0048)	0.4608 (0.0072)
type III	type I	0.5134 (0.0193)	0.4641 (0.0213)	0.6286 (0.1111)
type III	type II	0.4994 (0.0093)	0.4595 (0.0071)	0.5377 (0.0502)
type III	type III	0.4516 (0.0019)	0.4440 (0.0051)	0.4884 (0.0299)

Table 4: Analysis of systems: F1 (std) scores for the three 1DxSeq system architectures. The highest value for each train/test combination highlighted in bold.