A BILINGUAL GENERATIVE TRANSFORMER FOR SEMANTIC SENTENCE EMBEDDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Semantic sentence embedding models take natural language sentences and turn them into vectors, such that similar vectors indicate similarity in the semantics between the sentences. Bilingual data offers a useful signal for learning such embeddings: properties shared by both sentences in a translation pair are likely semantic, while divergent properties are likely stylistic or language-specific. We propose a deep latent variable model that attempts to perform source separation on parallel sentences, isolating what they have in common in a latent semantic vector, and explaining what is left over with language-specific latent vectors. Our proposed approach differs from past work on semantic sentence encoding in two ways. First, by using a variational probabilistic framework, we introduce priors that encourage source separation, and can use our model's posterior to predict sentence embeddings for monolingual data at test time. Second, we use high-capacity transformers as both data generating distributions and inference networks – contrasting with most past work on sentence embeddings. In experiments, our approach substantially outperforms the state-of-the-art on a standard suite of semantic similarity evaluations. Further, we demonstrate that our approach yields the largest gains on more difficult subsets of test where simple word overlap is not a good indicator of similarity.

1 INTRODUCTION

Learning useful representations of language has been a source of recent success in natural language processing (NLP). Much work has been done on learning representations for words (Mikolov et al., 2013; Pennington et al., 2014) and sentences (Kiros et al., 2015; Conneau et al., 2017). More recently, deep neural architectures have been used to learn contextualized word embeddings (Peters et al., 2018; Devlin et al., 2018) which have enabled state-of-the-art results on many tasks. We focus on learning semantic *sentence* embeddings in this paper, which play an important role many downstream applications. Since they do not require any labelled data for fine-tuning, sentence embeddings are useful for a variety of problems right out of the box. These include Semantic Textual Similarity (STS; Agirre et al. (2012)), mining bitext (Zweigenbaum et al., 2018), and paraphrase identification (Dolan et al., 2004). These tasks also have downstream uses like in fine-tuning machine translation systems (Wieting et al., 2019a) and data augmentation .

Prior work on sentence embedding has largely used deep architectures. Some recent approaches include Infersent (Conneau et al., 2017), the Universal Sentence Encoder (USE) (Cer et al., 2018), and Gensen (Subramanian et al., 2018). Surprisingly, however, the best performing models for many sentence embedding tasks relating to semantic similarity, in contrast, used simple architectures that are agnostic to word order. For instance, some of the top performing techniques use word embedding averaging (Wieting et al., 2016a), character n-gram averaging (Wieting et al., 2016b), and subword embedding averaging (Wieting et al., 2019b) to create representations. These simple approaches generalize well to unseen domains, but are fundamentally limited by their inability to capture word order. Training for such approaches relies on discriminative objectives defined on paraphrase data from bilingual pivoting (Ganitkevitch et al., 2013), back-translation of bilingual text (Wieting & Gimpel, 2018), or, recently, bilingual text directly (Wieting et al., 2019b). Approaches using parallel text and latent variable models have also been explored (Chen et al., 2019).

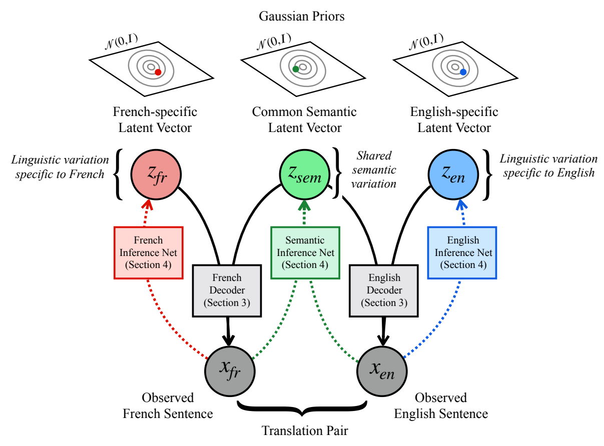
Intuitively, bilingual data offers a useful signal for learning semantic embeddings of sentences. Within a translation pair, properties shared by both sentences are likely semantic, while those that are divergent are likely stylistic or language-specific. In contrast with previous work, we leverage bilingual data directly to learn deep architectures for sentence embedding that are sensitive to word order in a probabilistic framework. Specifically, we propose a deep generative model that is encouraged to perform source separation on parallel sentences – isolating what they have in common in a latent semantic embedding, and explaining what is left over with language-specific latent vectors – through a variational objective. As a result, our approach is able to balance the high-capacity of deep transformer architectures (Vaswani et al., 2017) with the goal of generalizing to new domains. Further, by using amortized inference (Kingma & Welling, 2013) and introducing factored transformer-based inference networks for approximating the model's posterior on source separation, our trained system can effectively encode monolingual sentences at test time via posterior inference. Finally, since our model and training objective are probabilistic, in contrast with discriminative systems, our approach does not require knowledge of the distance metrics to be used during evaluation.

In experiments, our probabilistic source separation approach substantially outperforms the state-of-the-art on a standard suite of STS evaluations, generalizing more effectively than both order-agnostic baselines and previous deep architectures. Further, we conduct a thorough analysis by identifying subsets of the STS evaluation where simple word overlap is not able to accurately assess semantic similarity. On these most difficult instances we find that our approach yields the largest gains, indicating that our system is modeling word order to good effect.

Lastly, we analyze our model to uncover what information was captured by the source separation into semantic and language-specific variables. We find that the language-specific variables tend to explain more superficial properties of observations like overall sentence length and amount and location of punctuation, but not semantic information, matching our intuition.

2 Model

The generative process of our model, Bilingual Generative Tranformer (BGT), is depicted in Figure 2. Given parallel text in two languages (English (`en`) and French (`fr`)) that form a pair (x_{en}, x_{fr}), we hypothesis that we can learn separate representations for shared semantic content and language-specific variation. Further, the by encouraging the model to perform this source separation, the learned semantic encoders will more crisply represent the underlying semantics, increasing performance on downstream semantic tasks.



Gaussian Priors
$\mathcal{N}(0,I)$
$\mathcal{N}(0,I)$
$\mathcal{N}(0,I)$
French-specific Latent Vector
Common Semantic Latent Vector
English-specific Latent Vector
Linguistic variation specific to French
z_{fr}
z_{sem}
Shared semantic variation
z_{en}
Linguistic variation specific to English
French Inference Net (Section 4)
Semantic Inference Net (Section 4)
English Inference Net (Section 4)
French Decoder (Section 3)
English Decoder (Section 3)
x_{fr}
x_{en}
Observed French Sentence
Observed English Sentence
Translation Pair

Figure 1: The generative process of our model. Latent variables modeling the linguistic variation in French and English, z_{fr} and z_{en}, as well as a latent variable modeling the common semantics, z_{sem}, are drawn from a Gaussian distribution. The observed text in each language is then conditioned its language variable and z_{sem}.

More specifically, for each translation pair consisting of English sentence x_{en} and a French sentence x_{fr}, we sample language-specific variation variables (language variables) z_{fr} and z_{en} respectively for each side of the translation, as well as a shared semantic variation variable (semantic variable) z_{sem}. These latent variables $z_i \in \mathbb{R}^k$ are each sampled from a multivariate Gaussian prior $\mathcal{N}(0, I_k)$. In our decoder, x_{en} will be sampled conditioned on z_{sem} and z_{en}, while x_{fr} will be sampled conditioned on z_{sem} and z_{fr} Thus, our model is encouraged to explain variation that is shared by translations with the shared semantic variable, but must explain all language-specific variation with the corresponding language-specific variables. Our generative model is a type of source separation model.

We define X as our training set of parallel text consisting of N examples, $X = \{\langle x_{en}^1, x_{fr}^1\rangle, \ldots, \langle x_{en}^N, x_{fr}^N\rangle\}$ and Z as our collection of latent variables $Z =$

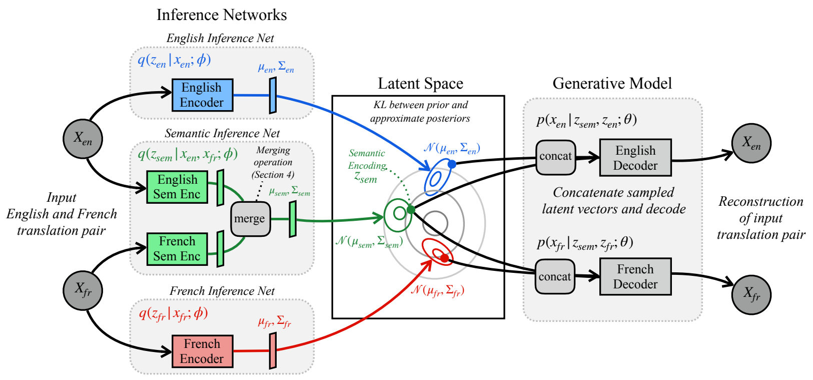

Inference Networks
English Inference Net
$q(z_{en}|x_{en};\phi)$
English Encoder
μ_{en}, Σ_{en}
Semantic Inference Net
$q(z_{sem}|x_{en}, x_{fr};\phi)$
Merging operation (Section 4)
English Sem Enc
French Sem Enc
merge
μ_{sem}, Σ_{sem}
French Inference Net
$q(z_{fr}|x_{fr};\phi)$
French Encoder
μ_{fr}, Σ_{fr}
X_{en}
X_{fr}
Input English and French translation pair
Latent Space
KL between prior and approximate posteriors
Semantic Encoding, z_{sem}
$\mathcal{N}(\mu_{en}, \Sigma_{en})$
$\mathcal{N}(\mu_{sem}, \Sigma_{sem})$
$\mathcal{N}(\mu_{fr}, \Sigma_{fr})$
Generative Model
$p(x_{en}|z_{sem}, z_{en};\theta)$
concat
English Decoder
Concatenate sampled latent vectors and decode
$p(x_{fr}|z_{sem}, z_{fr};\theta)$
concat
French Decoder
Reconstruction of input translation pair

Figure 2: The computation graph for the variational lower bound used during training. The English and French text are fed into their respective inference networks and the semantic inference network to ultimately produce the language variables z_{fr} and z_{en} and semantic variable z_{sem}. Each language variable is then concatenated to z_{sem} and used by the decoder to reconstruct the input sentence pair.

$(\langle z_{en}^1, z_{fr}^1, z_{sem}^1\rangle, \ldots, \langle z_{en}^N, z_{fr}^N, z_{sem}^N\rangle)$. For simplicity of notation, we often describe a single translation pair and corresponding latent variables with $\langle x_{en}, x_{fr}\rangle$ and $\langle z_{en}, z_{fr}, z_{sem}\rangle$, respectively.

Decoder Architecture. Most latent variable models for text use LSTMs (Hochreiter & Schmidhuber, 1997) as their decoders. However, state-of-the-art models in neural machine translation have seen increased performance and speed using deep Transformer architectures. We also found in our experiments (see Section C for details) that these led to increased performance in our setting. We use two decoders in our model, one for modelling $p(x_{fr}|z_{sem}, z_{fr};\theta)$ and one for modeling $p(x_{en}|z_{sem}, z_{en};\theta)$.

Each decoder takes in two latent variables, a language variable and a semantic variable. These variables are concatenated together prior to being used by the decoder for reconstruction. We explore 4 ways of using this latent vector: (1) Concatenate it to the word embeddings (Word) (2) Use it as the initial hidden state (Hidden, LSTM only) (3) Use it as you would the *attention context vector* in the traditional sequence-to-sequence framework (Attention) and (4) Concatenate it to the hidden state immediately prior to computing the logits (Logit). We also experimented with combinations of these four approaches. This analysis is in Appendix A. From these experiments, we see that the closer the sentence embedding is to the softmax, the better the performance on downstream tasks evaluating its semantic content. We hypothesise that this is due to better gradient propagation because the sentence embedding is now closer to the error signal. This is similar to the mechanism used by residual connections (He et al., 2016) to stabilize learning and improve performance. Since Attention and Logit performed best, we used these in our Transformer experiments.

3 Learning and Inference

We wish to maximize the marginal probability of the observed X given the latent variables Z and the parameters of the two decoders, θ. There is one decoder for each language, and each decoder takes in its corresponding latent variable and z_{sem}.

$$p(X;\theta) = \int_Z p(X, Z;\theta) dZ$$

Unfortunately, this integral is intractable due to the complex relationship between X and Z. However, related latent variable models like variational autoencoders (VAEs (Kingma & Welling, 2013)) learn by optimizing a variational lower bound on the log marginal likelihood. This surrogate objective is called the evidence lower bound (ELBO) and introduces a variational approximation, q to the true posterior of the model p. The q distribution is parameterized by a neural network with parameters ϕ. ELBO can be written for our model as follows:

$$\begin{aligned}\text{ELBO} =& \mathbb{E}_{q(Z|X;\phi)}[\log p(X|Z;\theta)] - \\ & \text{KL}(q(Z|X;\phi)||p(Z;\theta))\end{aligned}$$

This lower bound on the marginal can be optimized by gradient ascent by using the reparameterization trick (Kingma & Welling, 2013). This trick allows for the expectation under q to be approximated through sampling in a way that preserves backpropagation.

We make several independence assumptions for $q(z_{sem}, z_{en}, z_{fr}|x_{en}, x_{fr}; \phi)$. Specifically, to match our goal of source separation, we factor q as $q(z_{sem}, z_{en}, z_{fr}|x_{en}, x_{fr}; \phi) = q(z_{sem}|x_{en}, x_{fr}; \phi)q(z_{en}|x_{en})q(z_{fr}|x_{fr}; \phi)$, with ϕ being the parameters of the encoders that make up the inference networks, defined in the next paragraph.

Encoder Architecture. We use three inference networks as shown in Figure 2. An English inference network to produce the English language variable, a French inference network to produce the French language variable, and a semantic inference network to produce the semantic variable. Our model uses 4 encoders in total, with the semantic inference network having 2 that work together, one used to encode each language. Just as in the decoder architecture, we also use a Transformer for the encoders. We experiment with four ways of merging the two semantic variables in the semantic inference network: (1) Use just the English semantic variable, (2) Alternate between the English and French semantic variables (3) Alternate between the English and French semantic variables and include a regularization term to push these variables towards each other, and (4) Pool their representations together, dropping out one of them occasionally to avoid a training-testing discrepancy. We experiment with mean, max, and min pooling .

The motivation for (3) and (4) was that since both semantic encoders are supposed to be capturing the core semantic information of the sentence, their outputs should be similar since this information should be language agnostic. Interestingly, we found that (3) had the best performance in our setting, followed by (1) and (2) and (4) indicating that this regularization is helpful for learning these semantic representations.

Finally, to obtain the sentence embeddings that we'll use in downstream tasks, we use the mean of the semantic inference network, where only the English semantic encoder contributes to the merge operation.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

The training data for BGT and our our non-probabilistic model ablation, BGT W/O PRIOR, is a mixture of OpenSubtitles[1] `en-fr` data and `en-fr` Gigaword[2] data. To create our dataset, we combined the complete corpora of each dataset and then randomly selected 1,000,000 sentence pairs to be used for training with 10,000 used for validation. We use `sentencepiece` (Kudo & Richardson, 2018) with a vocabulary size of 20,000 to segment the sentences, and we chose sentence pairs whose sentences are at most 100 tokens each.

In designing the model architectures for the encoders and decoders, we experimented with Transformers and LSTMs. Due to better performance, we use a 5 layer Transformer for each of the encoders and a single layer decoder for each of the decoders. This design decision was empirically motivated as we found using a larger decoder was slower and worsened performance, but conversely, adding more encoder layers improved performance. More discussion of these trade-offs along with ablations and comparisons to LSTMs are included in Appendix C.

For both models, we set the dimension of the embeddings and hidden states for the encoders and decoders to 1024. Since we experiment with two different architectures,[3] we follow two different optimization strategies. For training models with Transformers, we use Adam (Kingma & Ba, 2014)

[1]`http://opus.nlpl.eu/OpenSubtitles.php`

[2]`https://www.statmt.org/wmt10/training-giga-fren.tar`

[3]We use LSTMS in our ablations.

with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-8}$. We use the same learning rate schedule as (Vaswani et al., 2017), i.e., the learning rate increases linearly for 4,000 steps to 5×10^{-4}, after which it is decayed proportionally to the inverse square root of the number of steps. For training the LSTM models, we use Adam with a fixed learning rate of 0.001. We train the models for 20 epochs.

We used label smoothed cross entropy with $\epsilon = 0.1$ as our objective function for machine translation. For BGT, we pretrain the semantic encoders (for both French and English) on translation, and then we freeze them for 10 epochs while updating all other parameters. We anneal the KL term so that it increased linearly for 50,000 updates. We also found that including the translation objective during training of BGT increased performance slightly, and so this loss was included in our experiments. We regularize the semantic encoders for `en` and `fr` using an L2 loss with a weight of 2.

Lastly, in Section B, we illustrate how crucial it is to train the Transformers with large batch sizes. Without this, the model can learn the goal task (such as translation) with reasonable accuracy, but the learned semantic embeddings are of poor quality until batch sizes approximately reach 25,000 tokens. Therefore, we use a maximum batch size of 50,000 tokens in our experiments.

4.2 Baseline Models

We experiment with four baseline models, covering the most effective approaches for learning paraphrastic embeddings from the literature.

The first baseline is the sentence piece averaging model, SP, from (Wieting et al., 2019b), which is the best of the averaging models (i.e. compared to averaging only words or character n-grams). This model uses a contrastive loss with a margin. Following their setting, we fix the margin to 0.4 and tune the number of batches to pool for selecting negative examples from $\{80, 120\}$. We set the dimension of the embeddings to 1024.

The second baseline is the LSTM model, BiLSTM, from (Wieting & Gimpel, 2017). We apply it to parallel text as was done in (Wieting et al., 2019b). We train a 1024 dimensional single layer LSTM, where sentence embeddings are obtained by mean-pooling the hidden states. Following (Wieting & Gimpel, 2017), we then shuffle the inputs with probability p, setting p to 0.3. BiLSTM uses the same contrastive loss function as SP, so we again fix the margin to 0.4 and tune the number of batches to combine for selecting negative examples from $\{80, 120\}$.

We also compare to well known sentence embedding models (Conneau et al., 2017; Subramanian et al., 2018; Cer et al., 2018), as well as BERT (Devlin et al., 2018). We used the pretrained BERT model in two ways to create a sentence embedding. The first is to concatenate the hidden states for the CLS token in the last four layers. The second way is to concatenate the hidden states of all word tokens in the last four layers and mean pool these representations. Both methods result in a 4096 dimension embedding.

We also implicitly compare to previous machine translation approaches like (Espana-Bonet et al., 2017; Schwenk & Douze, 2017; Artetxe & Schwenk, 2018) in Section A where we explore different variations of training LSTM sequence-to-sequence models. We find our translation baseline (both LSTM and Transformer) significantly outperforms the architectures from these works.

4.3 Evaluation

Our primary evaluation is the 2012-2016 SemEval Semantic Textual Similarity (STS) shared tasks (Agirre et al., 2012; 2013; 2014; 2015; 2016), which predict the degree to which sentences have the same meaning as measured by human judges. The evaluation metric is Pearson's r with the gold labels.

Secondly, we evaluate on *Hard STS*, where we combine and filter the STS datasets in order to make a more difficult evaluation. We hypothesize that these datasets contain many examples where their gold scores are easy to predict by either having similar structure and word choice and a high score or dissimilar structure and word choice and a low score. Therefore, we evaluate on these splits of the data using symmetric word error rate (SWER),[4] finding sentence pairs with low SWER and low

[4]We define symmetric word error rate for sentences s_1 and s_2 as $\frac{1}{2}WER(s_1, s_2)$ + $\frac{1}{2}WER(s_2, s_2)$, since word error rate (WER) is an asymmetric measure.

Data	Sentence 1	Sentence 2	Gold Score
Hard+	Other ways are needed.	It is necessary to find other means.	4.5
Hard-	How long can you keep chocolate in the freezer?	How long can I keep bread dough in the refrigerator?	1.0
Negation	It's not a good idea.	It's a good idea to do both.	1.0

Table 1: Examples from our *Hard STS* dataset and our negation split. The sentence pair in the first row has dissimilar structure and vocabulary yet a high gold score. The second sentence pair has similar structure and vocabulary and a low gold score. The last sentence pair contains a negation, *not* that causes otherwise similar sentences to have low semantic similarity.

	Semantic Textual Similarity (STS)							Machine Trans.	
Model	2012	2013	2014	2015	2016	Hard+	Hard-	MT15	MT16
BERT (CLS)	46.0	-	49.8	55.1	61.2	1.4	22.4	**62.0**	**57.1**
BERT (Mean)	48.8	-	54.0	59.2	63.3	3.1	24.1	58.3	54.4
Infersent	59.2	-	69.6	71.3	71.4	3.8	27.3	49.1	46.8
GenSen	60.6	-	65.8	74.2	66.4	6.4	24.7	59.9	53.7
USE	61.4	-	70.6	74.3	73.9	16.4	28.1	61.3	55.1
SP	67.6	59.6	75.1	78.2	76.6	18.6	25.5	53.0	49.4
BiLSTM	63.3	53.9	71.4	74.0	75.3	15.4	23.1	56.4	52.6
BGT w/o Prior	66.4	59.7	73.5	77.9	77.5	20.9	40.7	56.9	54.5
BGT	**68.2**	**61.4**	**75.3**	**78.8**	**77.6**	**23.2**	**42.8**	58.7	55.4

Table 2: Results of our models and prior work. The first two rows are models from this paper, the next 3 rows are models we trained on our training data, and the last 3 rows utilize pretrained models from the literature. We show results, measured in Pearson's $r \times 100$, for each year of the STS tasks 2012-2016, results in Pearson's $r \times 100$ on our two *Hard STS* datasets, and results in Spearman's $\rho \times 100$ on the two MT datasets.

gold scores as well as sentence pairs with high SWER and very high gold scores. This results in two datasets, Hard+ which have SWERs in the bottom 20% of all STS pairs and whose gold label is between 0 and 1,[5] and Hard- where the SWERs are in the top 20% ad the gold scores are between 4 and 5. We also evaluate on a split focused on negation. Example are shown in Table 1.

Lastly, we evaluate on machine translation evaluation metric tasks.[6] The purpose of these tasks is to produce scores for a reference and a candidate translation that correlate well with human labels. While similar to STS in that sentences with the same semantics will have higher scores, other variables are also accounted for like fluency. Recent work has shown that sentence similarity models can improve the quality of translation (Wieting et al., 2019a), and so improvements on these datasets might carry over to improvements in machine translation as well.

4.4 Results

The results on the STS, *Hard STS*, and the MT metric evaluations are shown in Table 2.[7] From the results, we see that BGT has the highest performance for every year of the STS task. It does especially well compared to prior work on the two *Hard STS* datasets. For the MT metric datasets, BERT (CLS) has the best performance on both, but BGT outperforms the other 3 models with the same experimental setup.[8]

[5]STS scores are between 0 and 5.

[6]We used the segment level data, where English is the target language, from newstest2015 and newstest2016 available at `http://statmt.org/wmt18/metrics-task.html`. The former contains 6 language pairs and the latter 4.

[7]We obtained values for STS 2012-2016 from (Subramanian et al., 2018). We left off the 2013 results as we include all 4 datasets from this year in our experiments but some of the prior work only used 3. Other results were obtained using the SentEval (Conneau & Kiela, 2018) toolkit.

[8]We note that it is very difficult to compare sentence embeddings as there are numerous factors that make comparisons uneven. These include different embeddings sizes, different sizes of architecture, different training data, using pretrained word embeddings, etc.

Data Split	n	BGT	SP
Bottom 20% WER, label $\in [0, 2]$	421	60.6	53.6
Bottom 10% WER, label $\in [0, 1]$	72	34.0	17.1
Top 20% WER, label $\in [3, 5]$	942	21.1	15.4
Top 10% WER, label $\in [4, 5]$	165	16.8	9.7
Negation	780	70.9	67.4
Top 20% WER, label $\in [0, 2]$	1380	51.5	49.9
Bottom 20% WER, label $\in [3, 5]$	2079	43.0	42.2

Table 3: Results, measured in Pearson's $r \times 100$, for different data splits of the STS data. The first four rows show difficult examples filtered symmetric word error rate (WER), the next row is a negation split, and The last 4 rows show relatively easy examples according to WER.

In Table 3, we show further difficult splits beyond those used in *Hard STS* and compare the top two performing models in the STS task from Table 2. We also show easier splits in the bottom of the table and examples where negation was likely present in the example.[9]

From these results, we see that both positive examples that have little shared vocabulary and structure and negative examples with significant shared vocabulary and structure benefit significantly from using a deeper architecture. Similarly, examples where negation occurs also benefit from our deeper model. These examples are difficult because more than just the identity of the words is needed to determine the relationship of the two sentences, and this is something that SP is not equipped to do since it is unable to model word order.

Lastly, the bottom four rows show *easier* examples where positive examples have high overlap and low WER and vice versa for negative examples. Both models perform similarly on this data, with the BGT model having a small edge consistent with the overall gap between these two models.

5 Analysis

We next analyze BGT by examining the language and semantic variables to learn what elements of syntax and semantics they capture relative to both each other and the sentence embeddings from BGT w/o Prior.

5.1 Language Variables

The language-specific variables enable the semantic variables of the model to focus on capturing the information that is shared by the translation pair.

We first show that the language variables are capturing little semantic information. We evaluate the learned English language variable from our BGT model on our suite of semantic tasks. The results in Table 4 show that these encoders perform closer to a random encoder than the semantic encoder from BGT. This is consistent with what we would expect to see if they are capturing extraneous language-specific information.

	Semantic Textual Similarity (STS)				
Model	2012	2013	2014	2015	2016
Random Encoder	47.5	34.0	51.1	52.4	43.8
English Language Encoder	49.8	45.3	58.3	65.2	64.5
English Semantic Encoder	68.2	61.4	75.3	78.8	77.6

Table 4: STS performance on the 2012-2016 datasets for a randomly initialized Transformer, the trained English language encoder from BGT, and the trained English semantic encoder from BGT. Performance is measured in Pearson's $r \times 100$.

Secondly, we use BGT to reconstruct sentences and analyze these reconstructions to determine what the language and semantic encoders are learning. We also reconstruct outputs by adding Gaussian

[9]We selected examples for the negation split where one sentence contained *not* or *n't* and the other did not.

Model/Source	Sentence
Source	uh , do n't worry , he 's not gonna find out.
Language-Noise	uh ... maybe she can n't talk about it , there 's not like this is out .
Semantic-Noise	sem noise: no , does n't n't we never know ,
Recon.	uh , do n't worry , he 's not gonna find out .
Source	this places agents in a unique position to maintain strong , long-term relationships with their customers .
Language-Noise	this smart position to develop long term .
Semantic-Noise	these sites in this body applies to they , they treat with no evidence .
Recon.	this unique role in a long term , they operate to maintain strong customers with their customers .

Table 5: Two reconstruction examples from our validation set. Notice that Language-Noise outputs tend to be more topically related, while Semantic-Noise outputs tend to have similar structure.

noise, with a mean of 0 and a covariance of the identity, to the semantic encoders or the English and French language variables to see what impact this has on the generated outputs. Our model with no noise added is Recon., with noise on the language variables is Language-Noise, and with noise on the semantic encoders is Semantic-Noise. We used 2,000 examples from our validation set as inputs, filtered to be between 15 and 30 tokens in length. Examples of the outputs are shown in Table 5 that show that the Language-Noise outputs tend to be more topically related, while Semantic-Noise outputs tend to have similar structure.

Length We analyzed how the length of sentences from Recon., Language-Noise, and Semantic-Noise compare to the source sentences. To do so, we computed the mean and median of the absolute difference of the lengths of the respective outputs of each model compared to the input. These results are shown in Table 7.

From the table, we see that adding noise to the language variables causes the reconstructions to deviate much more in length than when noise is added to the semantic encoder. This suggests that length is mostly encoded in these language variables.

Model	`en`	`fr`
Recon.	12.0	17.4
Language-Noise	53.9	68.0
Semantic-Noise	25.6	22.5

Table 6: Word error rate on the generated sentences from Recon., Language-Noise, and Semantic-Noise where all words that are not punctuation marks are masked.

Model	Mean		Median	
	`en`	`fr`	`en`	`fr`
Recon.	1.5	1.7	1.0	1.0
Language-Noise	9.6	68.0	7.0	5.0
Semantic-Noise	3.5	22.5	3.0	2.0

Table 7: Mean and median absolute length difference between source sentences and Recon., Language-Noise, and Semantic-Noise.

Punctuation We also analyze the outputs to see if the language encoders better encode information about punctuation. To do this, we mask out all words that aren't punctuation[10] and compute the WER with the source sentences. The results are shown in Table 6. Since the WERs are much lower when the language-specific latent variables are intact, we can surmise that this information is also mostly captured in these variables.

Semantics Lastly, we hypothesize that the core meaning of the sentence pair are mostly encoded in the semantic variables. To test this, we computed sentence embeddings by mean pooling the GloVe (Pennington et al., 2014) vectors for the nouns in the source and from the outputs of Language-Noise and Semantic-Noise. We found the average cosine similarity to be higher for Language-Noise at 58.0 than Semantic-Noise at 49.8, which suggests that even though the semantic variables seem to be lacking structure, they do carry semantic information about the sentence content.

[10]Punctuation were taken from the set $\{$ ' ! " # \$ % & \' ( ) $* + , - . / : ; < = > ?$ @ [ ] ^_ ' $\{ — \}$ ~ . $\}$.

6 CONCLUSION

We propose BGT, that attempts to perform source separation on parallel data to separate out the common semantic information between the two languages from the language-specific information which is stored in separate language variables. We find that our model bests all baselines on semantic similarity tasks, with the largest gains coming from a new challenge we propose as *Hard STS*, designed to foil methods approximating semantic similarity as word overlap. We also show that the language-specific variables are capturing more superfluous information like punctuation, where the common semantic variable contains most of the semantic information.

REFERENCES

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2012.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. * sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pp. 32–43, 2013.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval*, pp. 497–511, 2016.

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*, 2018.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. A multi-task approach for disentangling syntax and semantics in sentence representations. *arXiv preprint arXiv:1904.01173*, 2019.

Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Copenhagen, Denmark, September 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of COLING*, 2004.

Cristina Espana-Bonet, Adám Csaba Varga, Alberto Barrón-Cedeño, and Josef van Genabith. An empirical analysis of nmt-derived interlingual embeddings and their use in parallel sentence identification. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1340–1350, 2017.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The Paraphrase Database. In *Proceedings of HLT-NAACL*, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pp. 3294–3302, 2015.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pp. 2227–2237, 2018.

Holger Schwenk and Matthijs Douze. Learning joint multilingual sentence representations with neural machine translation. *arXiv preprint arXiv:1704.04154*, 2017.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

John Wieting and Kevin Gimpel. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2078–2088, Vancouver, Canada, July 2017.

John Wieting and Kevin Gimpel. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 451–462. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/P18-1042`.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In *Proceedings of the International Conference on Learning Representations*, 2016a.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1504–1515, 2016b.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. Beyond bleu: Training neural machine translation with semantic similarity. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4344–4355, 2019a.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. Simple and effective paraphrastic similarity from parallel translations. *Proceedings of the ACL*, 2019b.

Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. Overview of the third bucc shared task: Spotting parallel sentences in comparable corpora. In *Proceedings of 11th Workshop on Building and Using Comparable Corpora*, pp. 39–42, 2018.

A Location of Sentence Embedding in Decoder for Learning Representations

As mentioned in Section 2, we experimented with 4 ways to incorporate the sentence embedding into the decoder: Word, Hidden, Attention, and Logit. We also experimented with combinations of these 4 approaches. We evaluate these embeddings on the STS tasks and show the results, along with the time to train the models 1 epoch in Table 9.

For these experiments, we train a single layer bidirectional LSTM (BiLSTM) translation model with embedding size set to 1024 and hidden states set to 512 dimensions (in order to be roughly equivalent to our Transformer models). To form the sentence embedding in this variant, we mean pool the hidden states for each time step. The cell states of the decoder are initialized to the zero vector.

Architecture	STS	Time (s)
BiLSTM (Hidden)	54.3	1226
BiLSTM (Word)	67.2	1341
BiLSTM (Attention)	68.8	1481
BiLSTM (Logit)	69.4	1603
BiLSTM (Word + Hidden)	67.3	1377
BiLSTM (Word + Hidden + Attention)	68.3	1669
BiLSTM (Word + Hidden + Logit)	69.1	1655
BiLSTM (Word + Hidden + Attention + Logit)	68.9	1856

Table 8: Results for different ways of incorporating the sentence embedding in the decoder for a BiLSTM on the Semantic Textual Similarity (STS) datasets, along with the time taken to train the model for 1 epoch. Performance is measured in Pearson's $r \times 100$.

From this analysis, we see that the best performance is achieved with Logit, when the sentence embedding is place just prior to the softmax. The performance is much better than Hidden or Hidden+Word used in prior work. For instance, recently Artetxe & Schwenk (2018) used the Hidden+Word strategy in learning multilingual sentence embeddings.

A.1 VAE Training

We also found that incorporating the latent code of a VAE into the decoder using the Logit strategy increases the mutual information while having little affect on the log likelihood. We trained 2 LSTM VAE models where one uses the Hidden strategy and one uses Hidden + Logit. We found that the mutual information increased form 0.89 to 2.46, while the approximate negative log likelihood, estimated by importance weighting, increased slightly form 53.3 to 54.0 when using Logit.

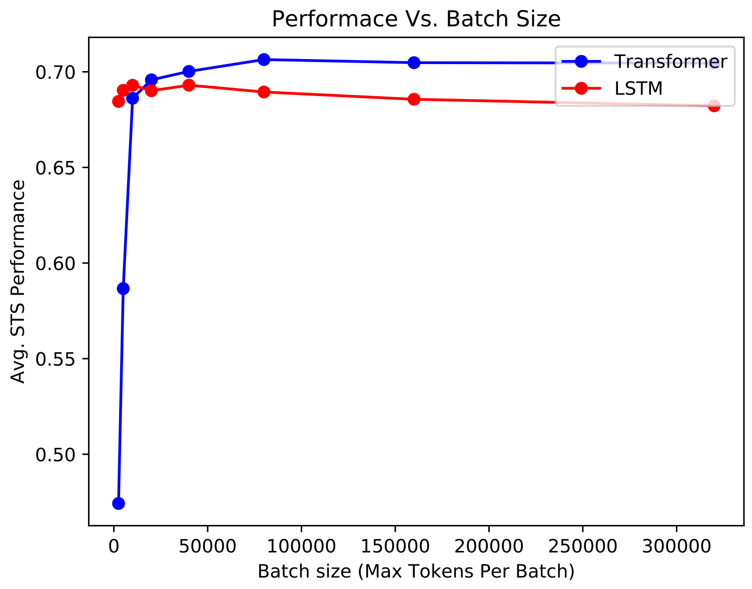


Figure 3: The relationship between average performance for each year of the STS tasks 2012-2016 (Pearson's $r \times 100$) and batch size (maximum number of words per batch).

B Relationship Between Batch Size and Performance for Transformer and LSTM

It has been observed previously that the performance of Transformer models is sensitive to batch size . We found this to be especially true when training sequence-to-sequence models to learn sentence embeddings. Figure 3 shows plots of the average 2012-2016 STS performance of the learned sentence embedding as batch size increases for both the BiLSTM and Transformer. Initially, at a batch size of 2500 tokens, sentence embeddings learned are worse than random, even though validation perplexity does decrease during this time. Performance rises as batch size increases up to around 100,000 tokens. In contrast, the BiLSTM is more robust to batch size, peaking much earlier around 25,000 tokens, and even degrading at higher batch sizes.

C Model Ablations

Architecture	STS	Time (s)
Transformer (5L/1L)	70.3	1767
Transformer (3L/1L)	70.1	1548
Transformer (1L/1L)	70.0	1244
Transformer (5L/5L)	69.8	2799

Table 9: Results on the Semantic Textual Similarity (STS) datasets for different configurations of BGT w/o Prior, along with the time taken to train the model for 1 epoch. (XL/YL) means X layers were used in the encoder and Y layers in the decoder. Performance is measured in Pearson's $r \times 100$.

In this section, we vary the number of layers in the encoder and decoder in BGT w/o Prior. We see that performance increases as the number of encoder layers increases, and also that a large decoder hurts performance, allowing us to save training time by using a single layer. These results can be compared to those in Table 9 showing that Transformers outperform BiLSTMS in these experiments.