Polarity inversion operators in PLM

Kletz David[1,2], Amsili Pascal[1], Candito Marie[2]
[1]Université Sorbonne Nouvelle & Lattice (CNRS/ENS-PSL/USN),
[2]Université Paris Cité & LLF (CNRS/UPC),
david.kletz@sorbonne-nouvelle.fr, marie.candito@u-paris.fr, pascal.amsili@ens.fr

Abstract

From a linguistic perspective, negation is a unique and inherently compositional operator. In this study, we investigate whether the `bert-large-cased` Pretrained Language Model (PLM) properly encodes this compositional aspect of negation when embedding a token that falls within the scope of negation. To explore this, we train two external Multi-Layer Perceptrons to modify contextual embeddings in a controlled manner. The goal is to reverse the polarity information encoded in the embedding while preserving all other token-related information. The first MLP, called the Negator, transforms a negative polarity into a positive one, while the second, the Affirmator, performs the reverse transformation. We then conduct a series of evaluations to assess the effectiveness of these operators. Our results indicate that while the Negator/Affirmator is functional, it only partially simulates the negation operator. Specifically, applying it recursively does not allow us to recover the original polarity, suggesting an incomplete representation of negation within the PLM's embeddings. In addition, a downstream evaluation on the Negated LAMA dataset reveals that the modifications introduced by the Negator/Affirmator lead to a slight improvement in the model's ability to account for negation in its predictions. However, applying the Negator/Affirmator recursively results in degraded representations, further reinforcing the idea that negation is not fully compositional within PLM embeddings.

1 Introduction

In this work, we aim to investigate how well Pretrained Language Models (PLMs) handle compositionality, by focusing on the possibility of defining a "negation operator."

From a logical and linguistic perspective, negation provides a typical example of semantic compositionalism: its effect is systematic and independent of the specific meaning of the clause to which it applies: negation simply reverses the truth value of a statement.

To put it differently, the meaning of a negation word (such as *not*) in a sentence does not depend on the particular verb used in the sentence, nor on the original polarity (i.e., whether the sentence was initially affirmative or negative). Instead, it follows a general rule: it systematically flips the sentence's polarity.

PLMs, however, do not construct the meaning of a sentence by recursively decomposing it into meaningful constituents. Instead, they generate contextual embeddings, so that the representation of a word depends on the surrounding words in the sentence. Given this, our goal is to identify a transformation (a function) that acts as a "negation operator" on embeddings. In other words, we want to find a way to manipulate the numerical representation of a word such that, after applying this transformation, we obtain an embedding that closely resembles what the model would have produced if the same word had occurred in a sentence with the opposite polarity.

For example, given an affirmative sentence like (1-a), we want to define an operation such that, when applied to the embedding that a PLM associates with the main verb *buy* in an affirmative context (noted Vp), it yields an embedding (noted $Vp-$) that is as close as possible to the embedding that the same PLM would assign to the token *buy* in a negative context (1-b) (noted Vn).

(1) a. Sam will buy a new car.
b. Sam will not buy a new car.

In the remainder of this paper, we will refer to a pair of sentences such as (1) as a **minimal pair** (keeping implicit the fact that the difference between the two sentences necessarily involves negation).

Our reasoning is as follows: if a PLM contains

Proceedings of the 29th Conference on Computational Natural Language Learning, pages 312–322
July 31 - August 1, 2025 ©2025 Association for Computational Linguistics

a compositional negation operator, then the difference between the embeddings Vp and Vn should be learnable, regardless of the lexical properties of the verb and the polarity of its context.

We therefore try in this work to learn a polarity inversion function that can operate on verb embeddings and that is sufficiently general to work on verbs whose lemmas were not seen at training, and on verbs occurring in affirmative or negative contexts.

We show that it is indeed possible to learn an operator (a *Negator*) that produces from Vp embeddings new embeddings $Vp-$ close enough to their corresponding Vn, and such that it generalizes correctly to lemmas not seen during training. This suggests that it is possible to locate in the embeddings distinct encodings for lexical representation and polarity. It is also possible to learn an *Affirmator* that produces an appropriate embedding $Vn+$ even for lemmas not seen at training (section 3). However, it turns out that these two operators cannot be used one for another: a Negator (trained only with embeddings occurring in affirmative contexts) does not succeed at "inverting" the embedding of a verb occurring in a positive context (similarly for the Affirmator). This indicates that they do not generalize to a true polarity inversion operation independent of the direction of the inversion, which is contrary to the classical logical and linguistic interpretation of negation (they are not involutions, i.e., they are not their own inverse: $\text{Negator}(\text{Negator}(x)) \neq x$ and $\text{Affirmator}(\text{Affirmator}(x)) \neq x$). On the contrary, we show (in section 4) that they are indeed reciprocal functions of each other: $\text{Affirmator}(\text{Negator}(x)) \approx x$.

It is worth studying further the properties of these two operators, even though they don't behave exactly as is expected from a logical perspective. Since they are not involutions, we study in section 5 the effect of their multiple application, and in particular a possible (non-linguistic) effect of "polarity reinforcement", usable to improve the processing of negation by a PLM.

Finally, in the last section of this paper (§ 6), we study the impact of the integration of our Negator into the processing pipeline of the negated LAMA task.

Our experiments show that the integration of the Negator leads to a slight improvement in the model's predictions. This suggests that modifying embeddings with the Negator allows the language modeling head to differentiate a little bit better between positive and negative embeddings, enabling it to adjust its predictions more accurately. However, the operator is applied several times ("recursively"), the predictions of the model become very unnatural, which is another way to show that our operators do not restrain their action to the strict encoding of polarity in the embeddings.

2 Related works

Negation in PLMs The presence of contextual polarity information in contextual embeddings generated by PLMs has been investigated by Celikkanat et al. (2020), who specifically looked for "traces" of negation. By analyzing contextual embeddings produced by a PLM, they showed that it is possible to predict whether the main verb of a sentence is negated or not. Building on this, Kletz et al. (2023b) showed that the encoding of such information is itself dependent on the syntactic position of the token used as input, in particular whether it falls or not within the scope of a negation.

Beyond encoding, the ability of models to consider negation in their predictions within a Masked Language Model (MLM) setup has also been explored. Kassner and Schütze (2020) and Ettinger (2020) examined how negating the main verb of a clause affects its truth value. Specifically, they investigated the capability of masked language models to adjust their predictions for a masked position when confronted with factual world knowledge (Li et al., 2016).

Kassner and Schütze (2020) constructed the negated LAMA dataset by negating sentences from the original LAMA dataset (Petroni et al., 2019). They then analyzed the behavior of masked PLMs when processing negated cloze-style sentences. Their findings revealed a similarity between model predictions in affirmative and negative contexts, leading them to conclude that "PLMs do not distinguish positive and negative sentences."

Similarly, Ettinger (2020) used sentences originally designed by Fischler et al. (1983) to observe how human expectations about sentence continuation shift when negation is introduced. The lack of corresponding adjustments in PLM predictions led her to a similar conclusion that PLMs exhibit insensitivity to negation.

However, other approaches (Gubelmann and Handschuh (2022) and Kletz et al. (2023a)), decided to avoid factual statements. They constructed examples with two sentences, where a particular word was either highly plausible (in positive cases) or semantically ruled out (in negative cases) at a masked position in the second sentence, given the context provided by the first. The fact that larger PLMs adjusted their predictions based on sentence polarity led these authors to a different conclusion that certain PLMs are indeed capable of considering negation.

Hosseini et al. (2021) proposed improving the predictions of `bert-base-cased` in negative contexts by fine-tuning it into a new model called `BERTNOT`. They created a dataset of 40,000 concatenated sentence pairs, each consisting of a premise (sourced from Wikipedia) and a hypothesis where a noun dependent on the main verb was selected and masked. Half of these pairs retained the hypothesis unchanged, while the other half contained a negated version of the premise, created by negating the main verb. The fine-tuning process involved two key objectives: one function aimed to prevent the model from predicting the selected token in sequences where the second sentence was negated, while another function ensured that the masked token distribution remained unchanged for the other 20,000 sequences. `BERTNOT` was subsequently evaluated using NLI datasets and Negated LAMA. The evaluation results indicated that `BERTNOT` made far fewer factually incorrect predictions than `bert -base-cased`.

Compositionality in PLMs In general, the evaluation of compositionality in language models focuses on compositional behaviors (McCurdy et al., 2024) and the ability of PLMs to generalize. Research in this area typically tests models through external tasks, where successful resolution implies the ability to generalize compositionally—either lexically (as in COGS (Kim and Linzen, 2020) and SCAN (Lake and Baroni, 2018)) or structurally (as in SLOG (Li et al., 2023)).

Kim and Linzen (2020) reported disappointing performance from tested models on generalization sets, concluding that these models struggle with both lexical and structural compositional generalization. However, more recent studies have shown that using models with pretraining strategies focused on meta-learning (Conklin et al., 2021) or employing newer transformer-based architectures (Sun et al., 2023; Tay et al., 2021; Raffel et al., 2020) significantly improves compositional generalization, surpassing the capabilities of smaller transformer models.

3 Inverting polarity : training a Negator and Affirmator

In this section, we learn mathematical functions (MLPs) to modify contextualized embeddings so as to mimic the difference between embeddings originating from the two clauses of a minimal pair. We will talk of the "**polarity of an embedding**" for short. Hence for instance, in *I wish war didn't exist*, the polarity of the embedding of *exist* or *war* is negative, whereas the polarity of the embedding of *I* or *wish* is positive.

More precisely, we consider embeddings of target verbs. The basic principle for our Negator function (resp. Affirmator) is to take as input the contextualized embedding of an affirmative verb, noted Vp (Vn for a negative verb) and output the corresponding embedding as if the verb was in a negative (resp. affirmative) context ($Vp-$, resp. $Vn+$)[1].

The Negator (resp. Affirmator) consists in a MLP trained on (Vp,Vn) pairs (resp. (Vn,Vp) pairs). The evaluation consists in comparing $Vp-$ to the original Vn, and $Vn+$ to the original Vp. For short we will talk of the **original** embeddings (Vp or Vn) and their corresponding **reversed** embeddings ($Vn+$ and $Vp-$).

Data We took as a starting point a set of 20,000 minimal pairs provided by Hosseini et al. (2021), formed with 20,000 sentences from Wikipedia, where the direct object of a target verb has been masked, along with a version where the target verb is negated.

We have deduplicated the 20,000 pairs, and removed pairs containing either zero or more than one masked position (resulting from errors in the masking process), and those where the target verb is tokenized into several subwords when encoded by the PLM we test (namely `bert-large-cased`). This brought the dataset down to $\sim$15,000 pairs. For our purposes, we restored the masked object, and identified the target verb[2], left unmasked.

We then split this data into 11,708 training pairs and 2,927 test pairs, each set corresponding to disjoint sets of target verb lemmas.

[1] We did try to obtain the Affirmator by defining the reciprocal function of the Negator. However, the learned parameter square matrices turned out to be non-invertible (details in Appendix A).

[2] To this end, we parsed the sentences using stanza (Qi et al., 2020), and took the closest verbal ancestor node of the direct object, in the dependency tree.

Architecture and training The MLPs for Negator and Affirmator have the same architecture: 4 hidden layers of same size as input contextualized embeddings, namely 1024, with *LeakyRELU* activation (with a negative slope of 10^{-2}) for the first 3 layers and ELU for the last layer ($\alpha = 1$).

We train the Negator on the training (Vp,Vn) pairs using the MSE loss, and simply switched to (Vn,Vp) pairs to train the Affirmator[3].

Evaluation metrics We measure the quality of the Negator (resp. Affirmator) using two direct metrics and two indirect metrics, each comparing the original embeddings to their corresponding reversed embeddings (hence comparing $Vp-$ to Vn, and $Vn+$ to the original Vp). The two direct metrics are simply **cosine** similarity and mean square error (**MSE**). The two indirect metrics compare the probability distributions output by the language modeling head of the PLM, when fed with an original embedding vs. when fed with the corresponding reversed one. More precisely, if we note Pn the distribution obtained with the original Vn embedding, and $Pp-$ that obtained from $Vp-$, we use the **KL-divergence** $D_{KL}(Pp - ||Pn)$ averaged over each evaluated pair, and the proportion of evaluated pairs for which the top-1 prediction is the same in Pn and $Pp-$ (and accordingly for the Affirmator case), hereafter **same-top-1**. Among these four metrics, higher cosines and same-top-1 will mean better quality, while it is the opposite for MSE and KL divergences.

Moreover, while same-top-1 can be interpreted in isolation, for the other three metrics, we need reference values for comparison. To this effect, we compute cosine, MSE and KL-divergence for sets of various pairs of embeddings, obtained by encoding sentence pairs from our dataset, with `bert-large-cased`. These pairs of embeddings either concern the same token from a pair of sentences varying in polarity (Vp, Vn), or different tokens from the same sentence, or two tokens in two different sentences but corresponding to the same word form, and finally two embeddings from two random tokens taken from two random sentences from the affirmative sentences of our dataset.

The reference values are provided in Table 1. The first row concerns Vp and Vn pairs, and provide the reference values for embeddings differing only in polarity, and we will refer to these values to evaluate our Negator and Affirmator. As all the metrics show, all other tested pairs of embeddings show a much higher divergence. Note that two distinct tokens of the same sentence (second row) have much closer embeddings than the embeddings of the same word in two different sentences (third row).

sent.	pol.	token	MSE	Cosine	KL-div
=	≠	=	0.02	0.96	0.05
=	=	≠	0.30	0.50	8.17
≠	?	=	0.46	0.23	9.21
≠	+	?	0.57	0.14	20.74

Table 1: Calibration of metrics: reference values for MSE, cosine and KL-divergence metrics, when using various kinds of pairs of embeddings. The pairs are either embeddings from the same sentence (when ignoring polarity) (first column), from sentences with equal, different, irrelevant (?) or positive polarity (pol. column), and from the same word or not (token column). The first row compares (Vp, Vn).

3.1 Results

	MSE	cosine	KL-div	same-top-1
Vp vs. $Vn+$	0.12	0.80	0.66	83.9
Vn vs. $Vp-$	0.13	0.79	0.80	81.5

Table 2: Evaluation of the Affirmator (first row) and Negator (second row) on the test set: comparison metrics for pairs of original vs reversed embeddings.

We provide the evaluation results of the Negator and Affirmator, computed on the test set, in Table 2. The same-top-1 results are above 80%. Interpreting the three other metrics requires to compare them to the reference values in Table 1. The same trend is observed for MSE, cosine and KL-divergence: although the results comparing original and reversed embeddings are less good than when comparing the original (Vp, Vn) pair (first row of Table 1), they are a lot better than when comparing other kinds of pairs of embeddings (last 3 rows of Table 1). These observations tend to show that our trained polarity inversion operations lead to embeddings that are (i) close to the corresponding original embedding ($Vn+$ close to Vp, $Vp-$ close to Vn); and (ii) close enough to appropriately feed the original language modeling head, resulting in a probability distribution over the vocabulary that is close to the original one.

[3] We use the Adam optimizer. We tuned the learning rate (10^3) and the number of epochs (4) using cross-validation on the training set.

Generalization across verbs Since the training and test set contain disjoint sets of verb lemmas, the previous observations tend to show a good generalization to verbs unseen during training. To further check this generalization, we would also like to verify that the averaging applied in the metrics does not hide a disparity in performance, and in particular that errors are not concentrated on a specific set of verbs. To this end we calculate the same-top-1 proportion per lemma, and count the number of verbal lemmas for which the proportion is very low (top-1 accuracy below 20%), indicating a total failure of inverting the polarity of embeddings for these verbs. We restrict ourselves to lemmas with at least 5 occurrences in the test set.

The results are provided in table 3. We observe only 3 lemmas with a same-top-1 proportion of less than 20% for Affirmator, and none for Negator. We can thus conclude that there are practically no lemmas for which polarity re-encoding systematically fails.

This further confirms that it is indeed possible to learn a polarity transformation of a verbal embedding, *independently of the corresponding verb*, a first step towards a compositional polarity inversion operator (cf. section 1).

Model	# tested lemmas	Cases w/ rate <20%
Affirmator	277	3
Negator	271	0

Table 3: Total number of unique lemmas tested, and number with same-top-1 proportion below 20%.

Generalization across polarities The second necessary condition was that the learnt polarity inversion operations should generalize across polarity. In our case, it means firstly that the Negator and Affirmator should actually correspond to the same (or a close) mathematical function, performing a polarity inversion independently of the polarity of its input. Secondly, given the logical interpretation of negation, both the Negator and Affirmator should be an involution, namely their own reciprocal function, hence Negator(Negator(V)) should be close to V. We report on this investigation in section 5.

4 Evaluation via a polarity probe

In order to further assess the effectiveness of the Negator/Affirmator, we employ a MLP probe trained to predict the polarity of verbal embeddings. Importantly, the probe is trained exclusively on Vp and Vn, without exposure to reversed embeddings ($Vp-$ and $Vn+$).

Training of the probe The trained probe is an MLP consisting of a hidden layer of the same size as the input (1024), with sigmoid activation. It is trained for 5 epochs with a learning rate of 0.3.

As training data we reuse the dataset used to train our Affirmators/Negators: we keep at random one sentence from each pair, which yields ≃14,000 sentences balanced with respect of their polarity. We split them into 11708/2927 for training and testing, keeping a balanced polarity in each set.

Evaluation on original embeddings The accuracy of the probe on the test set is provided in the "Original" columns of Table 4.

Embedding								
Original			Reversed			Reinforced		
inp.	exp.	acc.	inp.	exp.	acc.	inp.	exp.	acc.
Vn	n	95.9	$Vn+$	p	99.9	$Vn-$	n	99.9
Vp	p	96.6	$Vp-$	n	99.9	$Vp+$	p	99.8

Table 4: Accuracies of the polarity-predicting probe, on the verbal embeddings of the test set, using either the **Original** embeddings (Vn or Vp), the **Reversed** ones ($Vn+$ or $Vp-$), and the **Reinforced** ones ($Vp+$ or $Vn-$). Columns inp.: type of input embedding; Columns exp.: expected polarity label; Columns acc.: probe accuracy

We observe that the probe has a very high accuracy to predict the polarity of original embeddings (first three columns, above 95%), although not perfect.

Evaluation on reversed embeddings We now check how the probe behave when fed with reversed embeddings. Results of applying the probe on these are provided in the "Reversed" columns of Table 4. We observe almost perfect accuracy for both the Negator and the Affirmator. This constitutes a further evaluation of the quality of the Negator/Affirmator, since they allow to better predict the polarity of an embedding.

5 Polarity inversion or reinforcement?

In this section, we examine the effects of applying the Negator to a verbal embedding originating from a negated verb (which, following our notation, results in $Vn-$). Similarly, we analyze $Vp+$ cases, where the Affirmator is applied to a verbal embedding originally not negated.

In this case, if the Negator/Affirmator is the same transformation, applying a real inversion of polarity independently of the polarity of their argument, then $Vn-$ should be close to Vp and have positive polarity. $Vp+$ should be close to Vn and have negative polarity.

If on the contrary the Negator/Affirmator are distinct, each "moving" the polarity of their argument on a "polarity scale" in opposite directions, then we anticipate a reinforcement of the encoding of the polarity (and we will use the term **reinforced** embedding for $Vn-$ and $Vp+$ types of embeddings).

The results are provided in the "Reinforced" columns of Table 4, the accuracy being calculated when expecting a reinforcement rather than an inversion. We can see that the accuracies are almost perfect for both the Negator and Affirmator. So as the name "reinforced" hinted, we observe a reinforcement of the polarity instead of an inversion independent of input polarity.

Note though that the accuracy on the Reinforced cases is similar to that of the Reversed cases. So while it shows that the Negator/Affirmator does strengthen the polarity encoding, it is surprising it cannot surpass the reversed cases.

Error analysis We further study the counts of well-classified/misclassified cases, and whether the polarity inversion or reinforcement introduces new errors. Table 4 provides the exact counts of correct/incorrect polarity prediction by the probe, when fed by original, reversed and reinforced embeddings. After polarity inversion, we count 179 corrected errors and only 3 introduced errors (resp. 177 and 3 after reinforcement).

The very low number of new errors introduced by the Negator/Affirmator further assesses their ability to inverse/reinforce polarity encoding in embeddings, without altering it.

Orig.	Count	After modif.	Rev.	Reinf.
✓	re	↪✓ ↪ x	4589 3	4589 3
x	180	↪ ✓ ↪ x	179 1	177 3

Table 5: Counts of correct/incorrect labels after applying the polarity probe on original, reversed and reinforced embeddings.

6 Using the Negator to enhance `bert-large-cased`'s predictions

We now propose to use the Negator for a different objective: rather than studying the possibility of learning a compositional negation operator, we investigate whether the negator can help to improve the negation "understanding" of a `bert-large-cased` model, in a downstream task. We choose the negatedLAMA task, which Kassner and Schütze (2020) designed to assess the ability of `bert` to adapt its language modeling predictions to the presence of negation (cf. section 2).

6.1 The negated LAMA data and task

The negated LAMA dataset (Kassner and Schütze, 2020) is a negated version of LAMA (Petroni et al., 2019), itself developed to assess the factual knowledge stored in PLMs. It consists of factual statements derived from various encyclopedic sources[4], in which a token is masked (e.g. *dog* (2)), hereafter the **original affirmative token**).

The negated LAMA dataset is constructed by associating each affirmative factual statement (p) from LAMA with their negated counterpart (n).

(2) **Op** (Original): A beagle is a type of **dog**.
Mp A beagle is a type of [MASK].
Mn A beagle is not a type of [MASK].

The original affirmative token should be the top-1 prediction for the affirmative sentences, but this token becomes factually wrong in the negative counterparts, hence these pairs provide a way to assess a model's sensitivity to polarity changes.

Since the negated LAMA data is not explicitly available, we reconstructed the dataset, and the details of this process can be found in Appendix B. Consequently, although we made every effort to ensure accuracy, the version of the dataset we use differs from the ones employed by Kassner and Schütze (2020) and Hosseini et al. (2021).

To measure performances of the model, we use the **stability rate** of Kassner and Schütze (2020), which measures the percentage of identical top-1 predictions for (Mp,Mn) pairs. The lower the stability rate is, the more the model is sensitive to negation. Note that this measure does not take into consideration the cases where the top-1 prediction for Mp is not identical to the original affirmative

[4] Google-RE (Google, 2013), T-REX (Elsahar et al., 2018), ConceptNet (Speer and Havasi, 2012), and SQuAD (Rajpurkar et al., 2016).

token: if, for example in (2), a model has *mammal* as its top-1 prediction, what matters for the stability rate is whether this token is still the top-1 prediction in the negative case Mn. Hence, we also introduce a metric to quantify the average rate of factually incorrect predictions, referred to as the ***fipa* rate**[5]. The *fipa* rate measures the proportion of top-1 predictions for negated sentences that still match the original affirmative token. A lower *fipa* rate suggests that the model is better at generating factually correct predictions under negation[6].

6.2 Setup

We propose to integrate the Negator into the language modeling prediction pipeline of a `bert-large-cased` model, by applying the Negator to a token's representation at last layer, before feeding the language modeling head (see Figure 1).

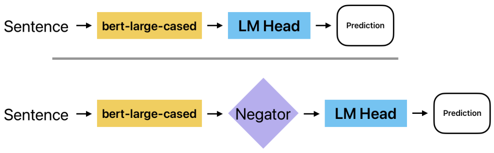

Figure 1: Inserting the Negator in the language modeling task.

Training We retrain a Negator using the encodings of the original masked tokens of Hosseini et al. (2021)'s dataset, not being unmasked (using our notation scheme, we consider Mp, $Mp-$, Mn, $Mn-$, under the same conditions as described in Section 3).

Application to negated LAMA The use of the Negator allows us to make two comparisons: between original, reversed and reinforced embeddings (Mn, Mp-, Mn-).

Furthermore, inspired by Ravfogel et al. (2021), we apply the Negator recursively multiple times, progressively even more reinforcing the encoding of negation polarity (e.g., $Mp \rightarrow Mp- \rightarrow Mp-- \rightarrow Mp---$ etc.). We will call these **super-reinforced embeddings**, and note $Mnk-$ the result of applying k times the Negator to Mn.

6.3 Quantitative analysis

The results for the *fipa* rate and stability rate are presented in Table 6.

The prediction shifts of the PLM are highly dependent on the dataset subset used, with no subset enabling the PLM to achieve a stability rate below 30%.

The insertion of the Negator into the processing pipeline under a polarity inversion configuration (line 2) alters the model's predictions. However, the stability and *fipa* rates do not show any improvement—often remaining similar or even worse—compared to directly negating the input sentence. The insertion of the Negator into the processing pipeline under a polarity reinforcement configuration (line 3) is the first combination to yield an improvement, reducing *fipa* rate by 5% to 20% and increasing the stability rate from 14% to 26%.

The use of super-reinforced embeddings leads to the most significant improvements in evaluation quality, both in polarity reinforcement and inversion configurations. The *fipa* rate decreases, ranging from 65% to 79%, and the stability rate improves between 71% and 84%.

Compared to other models, the model incorporating super-reinforced embeddings surpasses the performance of bert-large-cased, as tested by Kassner and Schütze (2020). Additionally, it outperforms BERTNOT (Hosseini et al., 2021), [7] achieving lower *fipa* rates than those reported by Hosseini et al. (2021). Furthermore, a comparison between line 1 and line 4 reveals that the reduction in fipa rate is even more significant than the improvement achieved by BERTNOT.

6.4 Qualitative analysis

To ensure that the representations are not degraded by the application of the Negator and that only the encoding of polarity is affected, we now conduct a complementary qualitative analysis.

We analyze the top-1 predictions of our architecture. For comparison, we revisit the four examples highlighted in Hosseini et al. (2021) and the eight examples from Kassner and Schütze (2020). The results are presented in Table 7.

Semantic and syntactic constraints are preserved in the $Mn-$ configuration.[8] However, the model's

[5] This metric may correspond to the average top-1 error rate used by Hosseini et al. (2021). However, since they do not explicitly define it, we cannot confirm this equivalence.

[6] Note each above cited work use only one of these two metrics, which clearly gives an incomplete evaluation.

[7] For this comparison, we refer to Table 12 in the Appendix of their paper, as the results presented in the main text—while higher—were obtained using a BERT-base-cased model.

[8] For instance, in the sentence "Charles Nodier did not die

	fipa rate				stability rate			
subset	SQUAD	conceptnet	Google-re	T-rex	SQUAD	conceptnet	Google-re	T-rex
Mn	11.2	2.7	22.2	57.7	43.4	31.6	60.3	90.0
Mp-	15.5	3.3	22.5	58.3	59.5	59.0	61.1	84.8
Mn-	8.9	2.4	20.2	54.5	32.6	23.5	44.1	77.4
Mn5-	**2.3**	**0.8**	**6.4**	**19.8**	**9.5**	**7.3**	**11.6**	**25.3**
Mp5-	5.3	1.4	11.9	23.1	18.4	14.6	25.3	31.4

Table 6: Percentage of cases where the top-1 prediction when feeding the LM head with embedding in column 1 is (left) identical to the expected factual answer for the Vp case, and (right) identical to the top-1 prediction for the Vp case; each broken down for each LAMA subset.

Paper	Sentence	Representation received by the LM head		
		Mn	Mn-	Mn5-
H	iOS is not developed by [MASK].	Apple (0.22)	Apple (0.19)	it (0.05)
H	The majority of the amazon forest is not in [MASK].	cultivation (0.43)	cultivation (0.13)	forest (0.04)
H	Charles Nodier did not die in [MASK].	battle (0.29)	battle (0.14)	prison (0.13)
H	Mac OS is not developed by [MASK].	Apple (0.73)	Apple (0.64)	Apple (0.19)
K&S	Marcel Oopa did not die in the city of [MASK].	Paris (0.09)	Paris (0.08)	residence (0.04)
K&S	Anatoly Alexine was not born in the city of [MASK].	Moscow (0.31)	Moscow (0.28)	town (0.05)
K&S	Platonism is not named after [MASK].	Plato (0.78)	Plato (0.35)	himself (0.48)
K&S	Lexus is not owned by [MASK].	Toyota (0.18)	Google (0.07)	it (0.03)
K&S	Birds cannot [MASK].	fly (0.76)	fly (0.33)	property (0.01)
K&S	A beagle is not a type of [MASK].	dog (0.83)	dog (0.72)	person (0.53)
K&S	Quran is not a [MASK] text.	religious (0.32)	religious (0.23)	valid (0.13)
K&S	Isaac's chains are not made out of [MASK].	iron (0.22)	iron(0.16)	stone (0.08)

Table 7: Qualitative analysis of predictions on embeddings modified by Negator. Column 'Paper': 'H' refers to sentences from Hosseini et al. (2021) 'K&S' refers to sentences from Kassner and Schütze (2020). Each cell indicates the prediction. The associated probability is given in parentheses.

predictions frequently remain unchanged from the original, which are often factually incorrect.

Incorporating super-reinforced embeddings does lead to modifications in predictions. However, with the super-reinforced embeddings, the generated sentences often appear unnatural, ultimately compromising the quality of the predictions.

These observations suggest that this method cannot serve to enhance negation interpretation of `bert-large-cased`.

7 Conclusions

In this paper, we explored the compositionality of negation within PLMs by investigating whether a transformation, which we call the "Negator/Affirmator," could reverse the polarity of a verb's embedding. Our results show that it is possible to learn such a function and that it can generalize to unseen lemmas. However, a complementary study reveals that the simple application of the Negator is not sufficient to significantly improve the predictions of `bert-large-cased` in the presence of negation, while multiple applications of the Negator improve the treatment of negation at the expense of a degradation of the embeddings. Even though a negation operation seems therefore learnable, its use for improving the predictions of a PLM still remains problematic.

Looking ahead, it would be interesting to extend this work by isolating operations that encode other compositional operators. This could help determine whether the handling of negation by PLMs is specific or if it is part of a broader pattern in the treatment of compositional operations.

in [MASK].", the masked position is syntactically constrained to be filled by a noun, noun phrase, or temporal expression, while semantically, it must refer to a place or time of death.

References

Hande Celikkanat, Sami Virpioja, Jörg Tiedemann, and Marianna Apidianaki. 2020. Controlling the Imprint of Passivization and Negation in Contextualized Representations. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 136–148, Online. Association for Computational Linguistics.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. In *Proceedings of the 59th Annual Meet-*

ing of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3322–3335, Online. Association for Computational Linguistics.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Allyson Ettinger. 2020. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Ira Fischler, Paul A Bloom, Donald G Childers, Salim E Roucos, and Nathan W Perry Jr. 1983. Brain potentials related to stages of sentence verification. *Psychophysiology*, 20(4):400–409.

Google. 2013. Google relation extraction corpus. `https://code.google.com/archive/p/relation-extraction-corpus/`. Corpus of judged relation triples.

Reto Gubelmann and Siegfried Handschuh. 2022. Context matters: A pragmatic study of PLMs' negation understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4602–4621, Dublin, Ireland. Association for Computational Linguistics.

Arian Hosseini, Siva Reddy, Dzmitry Bahdanau, R Devon Hjelm, Alessandro Sordoni, and Aaron Courville. 2021. Understanding by understanding not: Modeling negation in language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1301–1312, Online. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. Negated and Misprimed Probes for Pretrained Language Models: Birds Can Talk, But Cannot Fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

David Kletz, Pascal Amsili, and Marie Candito. 2023a. The self-contained negation test set. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 212–221, Singapore. Association for Computational Linguistics.

David Kletz, Marie Candito, and Pascal Amsili. 2023b. Probing structural constraints of negation in pretrained language models. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 541–554, Tórshavn, Faroe Islands. University of Tartu Library.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Bingzhi Li, Lucia Donatelli, Alexander Koller, Tal Linzen, Yuekun Yao, and Najoung Kim. 2023. SLOG: A structural generalization benchmark for semantic parsing. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3213–3232, Singapore. Association for Computational Linguistics.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Kate McCurdy, Paul Soulos, Paul Smolensky, Roland Fernandez, and Jianfeng Gao. 2024. Toward compositional behavior in neural models: A survey of current views. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9323–9339, Miami, Florida, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shauli Ravfogel, Grusha Prasad, Tal Linzen, and Yoav Goldberg. 2021. Counterfactual Interventions Reveal the Causal Effect of Relative Clause Representations on Agreement Prediction. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 194–209, Online. Association for Computational Linguistics.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey. European Language Resources Association (ELRA).

Kaiser Sun, Adina Williams, and Dieuwke Hupkes. 2023. A replication study of compositional generalization works on semantic parsing. In *ML Reproducibility Challenge 2022*.

Yi Tay, Mostafa Dehghani, Jai Prakash Gupta, Vamsi Aribandi, Dara Bahri, Zhen Qin, and Donald Metzler. 2021. Are pretrained convolutions better than pretrained transformers? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4349–4359, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Limitations

Our study focuses solely on the `bert-large-cased` model, meaning our findings cannot be directly generalized to other Pretrained Language Models (PLMs). Future work should extend this analysis to a broader range of architectures to assess whether our observations hold across different models.

Additionally, both our dataset and model were in English, limiting our conclusions to this linguistic context. Since negation varies across languages in both syntax and semantics, evaluating models trained on other languages would be necessary to determine the broader applicability of our approach.

A Attemps to define the reciprocal of the Negator

Instead of learning both a Negator and Affirmator, we also tried to learn a Negator, and then define its reciprocal, to serve as Affirmator. This supposes to define the reciprocal of activation functions and of linear combinations.

To this end, we used bijective activation functions, whose reciprocal functions are:

$$LeakyRELU^{-1}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \frac{x}{\alpha}, & \text{otherwise} \end{cases}$$

$$ELU^{-1}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ log(\frac{x}{\alpha+1}), & \text{otherwise} \end{cases}$$

The reciprocal of the linear combination with parameters W and b, requires W to be invertible (which is why we chose square parameter matrices), and is written as:

$$x = W^{-1}(y - b) \tag{1}$$

Unfortunately, we empirically observed across various runs that the resulting Negator contained at least one non-invertible matrix (namely with a rank lower than the shape of the matrix).

We also tried to use Moore-Penrose pseudoinverse parameter matrices[9]. In such a case, the definition of the reciprocal is as below (with W^{PI} the pseudoinverse of W):

$$x = W^{PI}(y - b) \tag{2}$$

So for a given linear layer, if $y = Wx + b$ then we compute a x' such that $x' = W^{PI}(y - b) \simeq x$, namely there exists a matrix M such that $x' = x + M$.

[9] Using pytorch, https://pytorch.org/docs/stable/generated/torch.linalg.pinv.html, Paszke et al. (2019))

This empirically failed in the sense that when applying the reciprocal functions in sequence, we noted the M matrices kept growing exponentially. The approximation made using pseudoinverses led to growing errors.

We conclude to the impossibility of inversing the Negator to obtain an Affirmator.

B Reconstruction and Preprocessing of Negated LAMA

The original LAMA dataset is available both in the repository of Petroni et al. (2019) [10] and on the Hugging Face (Wolf et al., 2020) platform [11]. However, with the exception of the SQUAD subset, the number of entries differs between these two sources for every subset. A comparison of the dataset sizes from these two sources can be found in Table 8, under the columns "LAMA" (repository from the original paper) and "LAMA HF" (Hugging Face platform).

The inputs of negated LAMA are either explicitly provided, or through the introduction of a negation pattern.

Upon examining the data, we found that not all entries could be used to reconstruct the negated LAMA dataset. We applied filtering criteria to exclude entries with the following issues:

- Presence of multiple masked tokens
- Absence of a corresponding negated sentence
- Lack of alignment with a recognizable negation pattern

These inconsistencies accounted for nearly two-thirds of the data in the Google-RE and T-REx subsets, and we were unable to fully resolve all of them.

The final sizes of the subsets used to evaluate our models are listed in the "Retained Examples" column of Table 8.

Consequently, we use a version of negated LAMA that is different from the one used by Kassner and Schütze (2020) and Hosseini et al. (2021).

Dataset	**LAMA Subsets**		
	LAMA	LAMA HF	#retained examples
SQUAD	305	305	301
conceptnet	2996	29774	8296
Google-re	5527	6106	2926
T-rex	34039	1304391	16991

Table 8: Subset sizes of LAMA from different sources. **Col. “LAMA”**: Number of entries in Petroni et al. (2019) repository. **Col. “LAMA HF”**: Number of entries in the Hugging Face version. **Col. “Retained Examples”**: Final number of entries used in our negated LAMA version.

[10] https://dl.fbaipublicfiles.com/LAMA/negated_data.tar.gz

[11] https://huggingface.co/datasets/facebook/lama