# How Emotionally Stable is ALBERT?
# Testing Robustness with Stochastic Weight Averaging on a Sentiment Analysis Task

**Urja Khurana**[♡] and **Eric Nalisnick**[◇] and **Antske Fokkens**[♡♣]

♡ CLTL, Dept. of Language, Literature & Communication, Vrije Universiteit Amsterdam
◇ Informatics Institute, University of Amsterdam
♣ Dept. of Mathematics and Computerscience, Eindhoven University of Technology

`{u.khurana,antske.fokkens}@vu.nl`
`e.t.nalisnick@uva.nl`

## Abstract

Despite their success, modern language models are fragile. Even small changes in their training pipeline can lead to unexpected results. We study this phenomenon by examining the robustness of ALBERT (Lan et al., 2020) in combination with Stochastic Weight Averaging (SWA)—a cheap way of ensembling—on a sentiment analysis task (SST-2). In particular, we analyze SWA's stability via CheckList criteria (Ribeiro et al., 2020), examining the agreement on errors made by models differing only in their random seed. We hypothesize that SWA is more stable because it ensembles model snapshots taken along the gradient descent trajectory. We quantify stability by comparing the models' mistakes with Fleiss' Kappa (Fleiss, 1971) and overlap ratio scores. We find that SWA reduces error rates in general; yet the models still suffer from their own distinct biases (according to CheckList).

## 1 Introduction

Current language models perform well on data that resemble the distribution they are trained on, but even a slight variation in the model training setup can lead to results that diverge from what is originally reported (Fokkens et al., 2013; Sellam et al., 2021). Furthermore, when the model relies on spurious correlations for decision making, it then contains biases that are not represented by real world data. Ideally a model should be robust to data that has (slightly) different characteristics from the data it was trained on. Accuracy and related metrics, despite their popularity, are usually not sufficient to identify these frailties. This is known as *underspecification* (D'Amour et al., 2020): different predictors can achieve similar results on a specific task, but exhibit diverging performance on other tasks due to different induced biases.

*Stress tests* are an increasingly popular method for exposing biases of a model. To test the linguistic capabilities and robustness of models, Ribeiro et al. (2020) introduce *CheckList*, an evaluation methodology that is comparable to the aforementioned *stress tests* for robustness. CheckList can be used to investigate which linguistic phenomena are fully captured by a model and for which the model is thus expected to be robust across datasets.

Robustness and generalization can be improved by ensembling multiple models. Training different models, however, is expensive. *Stochastic Weight Averaging* (SWA) (Izmailov et al., 2018) is a way of ensembling without the need to train different models. During training, the weights of the model at specific timepoints are averaged, avoiding the need to keep track of several models. The idea is that SWA explores different solutions close to a high performing minimum.

In this paper, we study the effect of SWA on the robustness to both a standard sentiment analysis dataset and different CheckList capabilities. We investigate if models varying only in their random seeds still have different behavior on the same data when trained using SWA. Specifically, we train ALBERT-large (Lan et al., 2020) on SST-2 (Socher et al., 2013), a sentiment analysis dataset, with 10 random seeds. We perform one run with SWA turned off (termed *vanilla models*) and repeat the procedure with SWA turned on (termed *SWA models*). We explore the robustness of the trained models using the CheckList methodology by looking at the stability of mistakes. We quantify this stability to measure the agreement in mistakes between the different models and compare the resulting values between the vanilla and SWA models.

Our main hypothesis is that **using SWA leads to more stable models**. We therefore expect more overlap across random seeds in the results on the SST-2 evaluation data. We also expect SWA to lead to more overlap in mistakes for CheckList items that are captured by part of the vanilla models. We also anticipate (minor) improvements of general

performance in both cases. For CheckList phenomena that are already largely captured or not at all, on the other hand, we do not expect to see major differences between vanilla models and SWA in terms of general performance or overlap.

**We make the following contributions**:

- We explore the effects of SWA on the stability and robustness of ALBERT-large that stem from underspecification.

- We perform the, to our knowledge, first joint study of SWA and CheckList.

- We provide an in-depth analysis of results by going beyond accuracy to look at overlap and agreement between random seeds and Check-List.

- We quantify agreement between different models by calculating overlap ratio and Fleiss' Kappa score on their mistakes.

We find that SWA improves on error rates in general, but results on increased stability are mixed: models with different random seeds still hold onto their own distinct induced biases on linguistic information captured by part of the models in our CheckList evaluation. There is minor improvement in stability on the Fleiss' Kappa score on the development set of SST-2, but results are not conclusive. Finally, we observe a large error rate for one of the random seeds on both SST-2 and CheckList, which also leads to a less strong result on increasing agreement between models.

## 2 Related Work

To the best of our knowledge, we are the first to combine SWA with CheckList and apply it to a BERT-based model to understand its effect on robustness with different random seeds. The work closest to ours has used variations of SWA for investigating the differences in interpretability on CNNs and LSTMs among different random seeds (Madhyastha and Jain, 2019). A similar method to Stochastic Weight Averaging was employed by Xu et al. (2020) with a different objective: improving the fine-tuning process of BERT. They propose to average the BERT model at each time-step and two types of knowledge distillation to improve fine-tuning of the model. The averaging receives slightly better results and their variant of knowledge distillation works the best. However, it is

unclear what the effect of this is on different random seeds.

Instead of looking at a form of ensembling, Hua et al. (2021) investigate the effect of injecting noise in BERT as a regularizer on the stability (sensitivity to input perturbation) of the models and show that fine-tuning performance improves. They point out that this improves generalizability as well, by looking at the difference in accuracy on the training and test set. However, training and test set might contain the same biases and hence might not reveal generalization issues (Elangovan et al., 2021).

**Varying Performance** Most work until now has focused on behavioral changes of models on train and test data when changing an arbitrary choice of the pipeline, such as the random seed (Zhong et al., 2021; Sellam et al., 2021). Investigating the behavior of language models with different pre-training and fine-tuning random seeds on an instance-level, Zhong et al. (2021) find that the fine-tuning random seed is influential for the variation in performance on an instance-level. This contrast in performance is also highlighted by Sellam et al. (2021); they release multiple BERT checkpoints with a different weight initialization and show diverging performance between similarly trained models. Such behavior has also been observed for out-of-distribution samples (McCoy et al., 2020; D'Amour et al., 2020; Amir et al., 2021), where different induced biases are found when the random seed is modified and checkpoints behave differently on unseen data, even when evaluation performance is similar. (D'Amour et al., 2020; Amir et al., 2021). Watson et al. (2021) show that outputs from explainability methods also vary when changing hyperparameters, e.g. the random seed.

**Model Evaluation** Evaluating models on a development set might not expose certain biases or weaknesses a model has acquired due to the possibility of the same biases occurring in the training set. Hence, scalable diagnostic methodologies are useful to investigate a model's capabilities (Wu et al., 2019; Ribeiro et al., 2020; Wu et al., 2021; Goel et al., 2021). Even though these methodologies all focus on evaluation, the approach can vary between the methods. Wu et al. (2021) tackle evaluation from a counterfactual point of view. Wu et al. (2019) not only examine counterfactuals but also grouping queries to ensure that error analysis is scaled to all instances. Likewise, Goel et al. (2021)

exploit such subpopulation grouping, in addition to adversarial attacks, perturbations, and evaluation sets. It is possible to be unaware of certain subpopulations for which the model is weak, and therefore d'Eon et al. (2021) introduce a method that looks for such weak groups. Ribeiro et al. (2020) provide a methodology to analyze robustness toward basic capabilities and operationalize this with different test types (e.g. invariance to specific perturbations, basic capabilities). There are also more task specific efforts for evaluation, such as perturbations for robustness in task-oriented dialog (Liu et al., 2021) and evaluation of bias in a sentiment analysis setting (Asyrofi et al., 2021).

## 3 Method

To examine *Stochastic Weight Averaging*'s effect on model stability due to *underspecification*, we finetune a pretrained ALBERT-large version 2 on the SST-2 dataset. We train two types of models 10 times:[1]

- *Vanilla model*: Model finetuned with the hyperparameter values from Lan et al. (2020)

- *SWA model*: Model finetuned for the first few epochs with the hyperparameter values from Lan et al. (2020) and then switching to a SWA training schedule

For all models, we keep the training protocol the same except for the random seed. We train 10 models with a different random seed per model type. This gives us 20 different models: 10 *vanilla models* and 10 *SWA models*.[2] We then investigate the robustness of each model on CheckList tests and compare the performance of *vanilla models* with *SWA models*.

Due to *underspecification*, the vanilla models are expected to have deviating performances on the tests across different random seeds, while the SWA models are expected to dampen this effect. We make a distinction between the following scenarios and what we expect:

---

1. **Linguistic information captured by all of the models**: We expect all of the models, regardless of the random seed, to be able to perform well on basic capabilities. Hence, we do not expect SWA to make much improvement, as there should not be a different behavior across random seeds. Stability will stay consistent here.

2. **Linguistic information captured by a part of the models**: This type of linguistic information is only captured by a part of the models due to their own induced biases. Hence, we expect that not all vanilla models behave similarly on such instances. With the introduction of SWA, more stability thus more overlap between mistakes is expected.

3. **Linguistic information captured by none of the models**: Some information cannot be captured by the model at all or it is unlikely that the model will be able to handle such information properly. In such cases, we do not expect SWA models to have an increase in performance, though that cannot be ruled out since it is possible that the weight space averaged by SWA is able to capture it. For the former, we do expect a large overlap of mistakes with SWA models since such information is not captured by any of the models.

### 3.1 Stochastic Weight Averaging

*Stochastic Weight Averaging* (SWA) is a cheap approach to create ensembles by averaging over different snapshots over the SGD trajectory, in contrast to the widely used approach of training different models (Izmailov et al., 2018). In essence, SWA ensembles in weight space instead of the usual model space. Due to the ensembling nature of correlated members from the same trajectory, we expect better generalization; a reduction in error rate and more stability in mistakes on unseen data.

We employ a strategy where the SWA models are trained in the same manner as the vanilla models for the first two epochs. This cut-off epoch is chosen empirically, by observing that the vanilla models start converging around 2-3 epochs. We make use of the Adam optimizer instead of the SGD optimizer since the former optimizer is used for the training of ALBERT. From the third epoch, the learning rate drops to a constant learning rate and at every end of the epoch, the model weights are

---

[1] We provide all code at https://github.com/cltl/robustness-albert

[2] The experiments originally contained five random seeds, of which *Random Seed 0* had exceptionally poor performance of 90.83 accuracy on the development set. This was far from the reported validation accuracy of 94.9 (https://github.com/google-research/albert#albert). For the camera-ready version, we trained an additional five seeds, which confirmed that the anomalous one is indeed an outlier.

averaged with the running average weights. With a high constant learning rate, the model is able to explore other solutions that are close to the local minimum that was found after two epochs and close to convergence. The respective constant learning rates of each random seed can be found in Table 1. The values for the learning rates are found empirically on the development set with the following candidate learning rates: {6e-06, 7.5e-06}.[3]

|  | SWA Learning Rate |
| --- | --- |
| Random Seed 0 | 6e-06 |
| Random Seed 1 | 7.5e-06 |
| Random Seed 2 | 6e-06 |
| Random Seed 3 | 6e-06 |
| Random Seed 4 | 7.5e-06 |
| Random Seed 5 | 6e-06 |
| Random Seed 6 | 6e-06 |
| Random Seed 7 | 6e-06 |
| Random Seed 8 | 6e-06 |
| Random Seed 9 | 7.5e-06 |

Table 1: The constant SWA learning rates for each random seed.

## 3.2 SST-2 Dataset

We use the binary version of the Stanford Sentiment Treebank dataset[4] (Socher et al., 2013), which consists of human-annotated sentences from movie reviews originating from `rottentomatoes.com` for a sentiment classification task. This version of the dataset is also included in the GLUE task (Wang et al., 2018). We use this dataset since sentiment analysis is an interesting task to study underspecification as it is a more subjective task, making rigorous, multifaceted evaluation even more important. The training set consists of 67349 phrases, while the validation and test dataset consist of 872 and 1821 sentences respectively. We use the training and validation set for the training procedure, while the test set is used for the generation of specific CheckList items.

---

[3]We looked at the learning rates in examples from the original paper at `https://github.com/timgaripov/swa#examples` where some SWA learning rates are half of the original learning rate and explored close candidate learning rates. From previous initial experiments learning rate $5e-06$ did not work and was thus left out in these sets of experiments.

[4]`https://nlp.stanford.edu/sentiment/index.html`

## 3.3 Checklist Evaluation

CheckList is a methodology to test basic and linguistic capabilities of a model, similar to behavioral testing in software engineering (Ribeiro et al., 2020). They make a distinction between three types of tests:

**Minimum Functionality Test (MFT)**: Small examples to test for basic capabilities. We test if each instance has the specified label.

**Invariance Test (INV)**: Tests that apply perturbations to the input and expect the prediction to stay consistent, *regardless of the correctness of the prediction*. The original input together with its perturbations is seen as one test case.

**Directional Expectation Tests (DIR)**: Tests where the output is expected to change in a specific way, when the input is modified: the confidence is expected to change in a specific direction. Similar to INV tests, the original input with modifications is seen as a test case.

In this paper, we consider different MFTs, INVs, and DIRs tests for sentiment analysis. We check for basic capabilities and robustness. Each trained model is evaluated on our CheckList set up and their performances are compared. We expect that vanilla models make more mistakes than SWA models and qualitatively make less overlapping mistakes due to each model having their own different induced biases. On the other hand, SWA models are expected to have more overlapping mistakes, due to its ensembling and explorative nature in the weight space.

We created 18 CheckList capability tests by adapting tests from the CheckList GitHub repository[5] to the use-case in this paper. For reasons of space, we refer to individual capability tests with transparent names followed by the test size, only using short explanations when the name by itself is not sufficiently clear. For tests that perturb the input and are not created from scratch, we use the test set from SST-2. Each original input can be augmented more than once, depending on the capability. These tests are followed by two numbers when introduced: the number of original items and total items with perturbations included. A full overview of the CheckList capabilities and their sizes can be found in Table 7 in Appendix D.

---

[5]`https://github.com/marcotcr/checklist/blob/master/notebooks/Sentiment.ipynb`

# 4 Results

This section presents the outcome of our experiments. We first provide results on the original dataset and then the results on CheckList items. Lastly we examine how stable vanilla and SWA models are by looking at the label agreement between models trained from different seeds.

## 4.1 Stochastic Weight Averaging

As mentioned in the previous section, we originally ran our experiments on five random seeds and added five additional seeds after observing that one seed performed lower than all others. When we compare the accuracy of the vanilla models with the SWA models on the validation set of SST-2 in Table 2, it is evident that most of the SWA models perform slightly better than the vanilla models. The only exceptions are *Random Seed 0, 7,* and *8*. Upon running our experiments on five additional seeds, *Random Seed 0* remains the only seed that has an accuracy around 0.90, confirming that it is an outlier. The SWA versions of the other two random seeds might not outperform their vanilla counterparts but achieve a close accuracy.

Due to the outlying behavior of *Random Seed 0*, we leave its results out of the rest of the analysis, to avoid noise from this model influencing the analysis. We present the complete results with *Random Seed 0* included in Appendix C.

|               | Vanilla | SWA    |
|---------------|---------|--------|
| Random Seed 0 | **0.9083** | 0.8991 |
| Random Seed 1 | 0.9507  | **0.9541** |
| Random Seed 2 | 0.9450  | **0.9495** |
| Random Seed 3 | 0.9507  | **0.9541** |
| Random Seed 4 | 0.9450  | **0.9461** |
| Random Seed 5 | 0.9495  | **0.9507** |
| Random Seed 6 | 0.9450  | **0.9472** |
| Random Seed 7 | **0.9438** | 0.9392 |
| Random Seed 8 | **0.9461** | 0.9450 |
| Random Seed 9 | 0.9415  | **0.9461** |

Table 2: Accuracy on the validation set of SST-2 for the vanilla and SWA models of the different random seeds.

## 4.2 CheckList Evaluation

### 4.2.1 Vanilla Model Results

**Error Rates** We show the failure rate for each capability per vanilla model in Figure 1a. For the *Movie Sentiments* (n=58), *Single Positive Words* (n=22), *Single Negative Words* (n=14), and *Sentiment-laden Words in Context* (n=1350) capabilities there are no mistakes made by any of the vanilla models. On *Add Positive Phrases* (n=500, m=5500), only *Random Seed 8* makes mistakes with a very small error rate. Similarly, on *Movie Industries Sentiments* (n=1200) only *Random Seed 8* and *Random Seed 2* make mistakes, again with very small error rates that would not be visible on the plot. Hence for clarity, these capabilities are left out of the plot.

There is not much variation in the error rate for most of the capabilities. The most variation in performance among the random seeds can be observed for the capability that tests negations of positive sentences, with a neutral sentiment in the middle of the sentence: *Negation of Positive, neutral words in the middle* (D) (n=500). Interestingly, it is evident that particular random seeds can deal with negation better than others: *Random Seed 1, 4,* and *5*. These random seeds have the lowest error rate for both *Negation of Positive Sentences* (C) (n=1350) and *Negation of Positive, neutral words in the middle*.

**Overlap Ratios** A similar error rate, however, does not mean that the errors occur for the same instances. Hence, we analyze the overlap of errors of the vanilla models per capability. We calculate an overlap ratio by dividing the intersection of the failures of two random seeds by the union of those same failures. In contrast to the error rates, the overlap ratios are on an instance-level instead of case-level. There is no overlap of errors between the models for the capability *Add Positive Phrases*. The capability with the highest overlap ratio is *Movie Genre Specific Sentiments* (A) (n=736), which checks for sentiments that are fitting or not for specific genres: e.g. a scared feeling after watching a horror movie. This indicates that most of the models make similar mistakes for this capability. When looking at the mistakes, all the models misclassify sentences about horror movies being terrifying, scary, frightening or calming, a comedy movie being serious and a drama movie being funny instead of serious. In general, most of the vanilla models have a low overlap ratio, with the only exceptions being *Negation of Positive, neutral words in the middle* (D) and *Temporal Sentiment Change* (B) (n=2152). The latter capability contains sentences where the sentiment changes over time. These two contain certain random seeds that
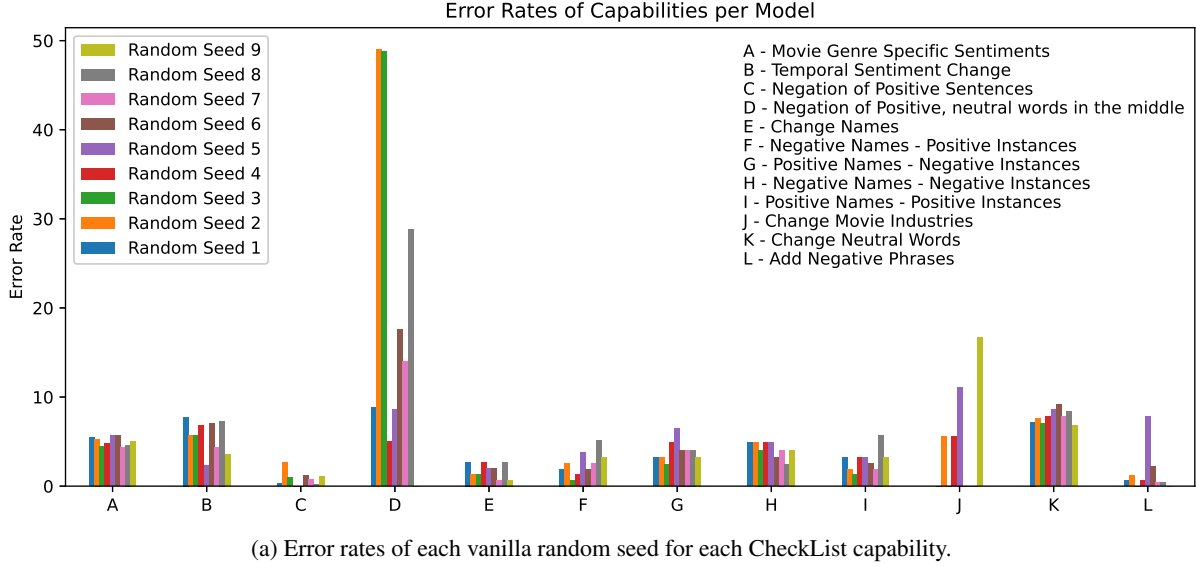
(a) Error rates of each vanilla random seed for each CheckList capability.
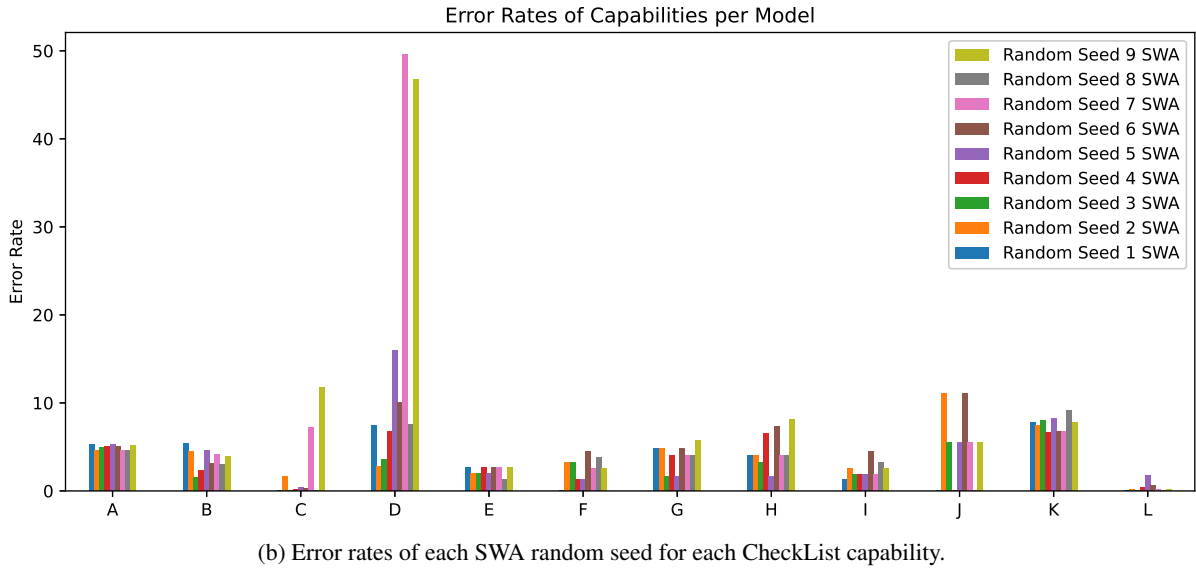


(b) Error rates of each SWA random seed for each CheckList capability.

Figure 1: Comparison of error rates per capability of vanilla and SWA models.

achieve a higher overlap ratio, as we can see in the spread of the box for these capabilities.

### 4.2.2 SWA Model Results

**Error Rates** Error rates for the SWA models per capability can be found in Figure 1b. In general, we can observe a (slight) reduction in error rate with SWA models compared to vanilla models. On *Add Positive Phrases*, only *Random Seed 5* and *Random Seed 6* have a slight increase in error rate. The latter is also the only one to make a mistake on *Movie Industries Sentiments*[6]. The largest drop can be seen for *Negation of Positive, neutral words in the middle* (D), where the diverging performance seen

---

[6]These error rates are too low (0.2% - 0.35%) to be visible in the plot, hence left out.

for the vanilla models has been reduced for most random seeds, except for *Random Seed 7* and *Random Seed 9*, whose error rates increase significantly. Similar behavior can be observed for *Negation of Positive Sentences* (C), where only the SWA versions of *Random Seed 7* and *9* have an increase in error rate. This suggests that the SWA solution for these two random seeds is worse in handling negation than their corresponding vanilla versions. For other capabilities, the error rate mostly reduces slightly or stays the same. The only exceptions are *Positive Names - Negative Instances* (G) (n=123, m=1353) and *Negative Names - Negative Instances* (H) (n=123, m=1353), where Negative Names are names that tend to occur in negative reviews in the training data, similarly for positive names, and we

(a) Comparison of variation in error rates between vanilla (red boxes) and SWA models (blue boxes), showcased per CheckList capability. Outliers are indicated with a circle.
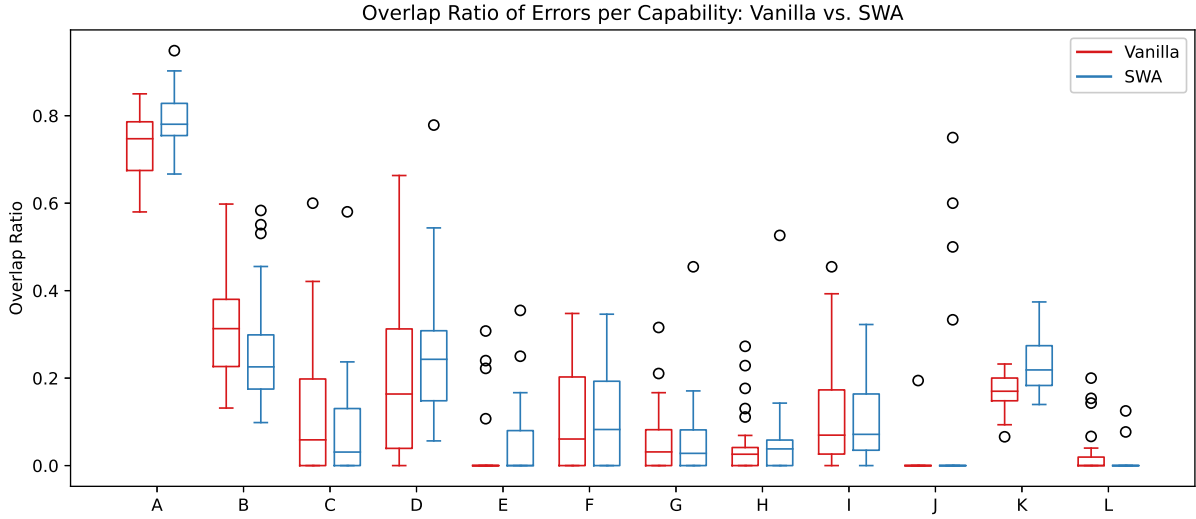


(b) Comparison of variation in overlap ratios between vanilla (red boxes) and SWA models (blue boxes), showcased per CheckList capability. Outliers are indicated with a circle.

Figure 2: Comparing the variation of error rates and overlap ratios per capability for vanilla and SWA models. Legend for the x-axis can be found in Figure 1

insert these names in negative instances of the test set. More details are provided in Appendix D.

**Overlap Ratios**   The overlap ratio for most capabilities remains low. Notably, the spread of overlap ratio for *Movie Genre Specific Sentiments* (A) increases from the vanilla models. All of the models still struggle with understanding that horror movies being terrifying, scary or frightening is positive, and calming is negative. This is in line with the expectations of SWA not improving (much) on capabilities that are not captured by any of the models. We find increase in overlap for *Change Names* (E) (n=147, m=1617), *Negative Names -*

*Negative Instances* (H), and *Change Neutral Words* (K) (n=500, m=3846), in accordance with our expectation of SWA bringing more stability. There is a different trend, against expectations, for *Add Negative Phrases* (L), *Negation of Positive Sentences* (C), and *Temporal Sentiment Change* (B), where the large variation of overlap of vanilla models is reduced significantly. For the rest of the capabilities, the overlap ratio appears to stay somewhat the same.

Overall, there are three different outcomes when comparing stability with SWA to vanilla models: (1) Good performance of vanilla models stays con-

sistent for the SWA models. (2) Large variations in error rates with vanilla models are reduced with SWA, but the overlap of mistakes does not increase and might decrease for some cases. (3) Overlap ratio with SWA does not necessarily increase, when error rates of the vanilla models are somewhat similar and remain the same for the SWA models. As such, we do not find evidence to confirm our hypothesis based on overlap between the outcomes on CheckList items.

## 4.3 Fleiss' Kappa

To further investigate the stability of SWA models, we measure the inter-model agreement on the misclassifications with the use of Fleiss' Kappa (Fleiss, 1971). This measure is used for inter-annotator agreement which can be related to the nine random seeds. In our case the annotators and the predictions being their annotations, used for both the vanilla and SWA models. Negative values or values close to zero are considered to indicate a rather low agreement, while the higher the value, the more agreement there is.

|  | **Vanilla** | **SWA** | **Difference** |
|---|---|---|---|
| *With Random Seed 0* | 0.205964 | 0.247299 | 0.041335 |
| *Without Random Seed 0* | 0.226725 | 0.360317 | 0.133592 |
| *With Random Seed 0* | 0.3984 | 0.4381 | 0.03967 |
| *Without Random Seed 0* | 0.3881 | 0.4106 | 0.0225 |

Table 3: Fleiss' Kappa values of the vanilla and SWA models on the agreement on the misclassifications on the development set. The upper block is with the first five random seeds and the lower is with all 10.

The results on the development set in Table 3 illustrate a significant increase in agreement for SWA models, when considering the initial four random seeds, without outlier *Random Seed 0*. While the agreement is still on the lower side, hinting at the presence of induced biases, the increase indicates more agreement on errors between the models and lesser distinct mistakes. We hence look at the Fleiss' Kappa values with the additional five random seeds incorporated. The Fleiss' Kappa agreement increases significantly in general for both the vanilla and SWA models. We now only observe a small increase in agreement when applying SWA

compared to the vanilla models.

|  | **Vanilla** | **SWA** | **Difference** |
|---|---|---|---|
| Negation of Positive Sentences | 0.029640 | 0.020448 | -0.009192 |
| Negation of Positive, neutral words in the middle | 0.107637 | 0.142219 | 0.034582 |
| Movie Genre Specific Sentiments | 0.581853 | 0.660138 | 0.078285 |
| Temporal Sentiment Change | 0.248653 | 0.290926 | 0.042273 |
| Change Names | -0.091694 | -0.084096 | 0.007598 |
| Negative Names - Positive Instances | 0.006975 | 0.006021 | -0.000954 |
| Positive Names - Negative Instances | -0.069162 | -0.076226 | -0.007064 |
| Negative Names - Negative Instances | -0.082486 | -0.069141 | 0.013346 |
| Positive Names - Positive Instances | 0.012704 | 0.035196 | 0.022492 |
| Change Movie Industries | -0.072503 | -0.052239 | 0.020264 |
| Change Neutral Words | 0.087306 | 0.135759 | 0.048453 |
| Add Negative Phrases | -0.031328 | -0.062053 | -0.030724 |

Table 4: Fleiss' Kappa values of the vanilla and SWA models on the agreement on CheckList mistakes per capability. The first part of the table shows the MFT capabilities, the second part are the INV capabilities, and the third part are the DIR capabilities.

We calculate the Kappa measure on the predictions of all the random seeds on the CheckList items as well. For the tests that measure basic capabilities (MFTs), we look at the agreement on predictions of errors. With tests that perturb an input (INVs), the instances that flip the output prediction are considered as a failure, so we check for model agreement on flipping for an instance. Similarly, for capabilities that test a directional change in confidence (DIRs), instances that go against the expected direction are considered failures and we compare model agreement on if they change in the same direction.

The Kappa values for the CheckList mistakes in Table 4 stay mostly unchanged with slight increases or decreases in agreement. This is in accordance with the results observed for the development set mistakes: it appears that SWA does not provide the stability across random seeds and still suffers from its own induced biases. Generally, the agreement is on the lower side. The Kappa values for *Movie Industries Sentiments* and *Add Positive Phrases* were 0.0 for both vanilla and SWA models and hence left out of the table. For *Movie Genre Specific Sentiments* we see a large agreement and the biggest

increase in agreement with SWA. This corresponds to the high overlap ratio for the same capability.

While SWA globally cuts down on error rate, it appears that this does not necessarily translate to improvement in stability: there is still disagreement in the labels assigned by individual models. Even with SWA, the models appear to make different errors on CheckList as confirmed by the low Kappa values and overlap ratio. For some capabilities the spread of the overlap ratio is on the higher side, indicating that *some* random seed models are closer to each other in terms of decision making, but this does not hold for *all*.

## 5 Discussion

This research illustrates the potential impact of random seeds. First, our original sample of 5 seeds contained an outlier that performed far worse than the other seeds (as well as the original study). Second, while initial results on the SST-2 development set were promising when looking at the 4 random seeds that showed normal behavior, these results did not hold when adding 5 additional random seeds. This highlights the necessity for proper analysis and the fragility of deep language models. Possibly, the initial random seeds were closer to each other in the weight space and hence SWA appeared to increase the agreement significantly. The additional random seeds could lie farther away, thus subsiding the increased agreement. In the future, more comprehensive research on the proximity and behavior of different random seeds could therefore be useful.

Even though CheckList provides an easy way to investigate the capabilities of a model, automatizing some tests can be hard. There can be situations in which labels indicated for a specific capability might not hold for a certain test case. For instance, negating a negative sentence might not always lead to a positive sentence, it can also be neutral. Similarly, we applied negations on some instances from the test set but the label is not required to flip, depending on the placement of the negation. Therefore, we leave out the results in our conclusions as the labels did not always make sense upon investigation. In some instances, it is also unclear what the resulting label should be. We have added the results for these specific capabilities in Appendix B for completeness. For further experiments, we would like to manually generate some CheckList capabilities to ensure validity of the labels. This will also enable us to focus on the creation of more subjective tests, cases that are less black-and-white than the tests conducted in this research. We can then gain more insights into the fragility of models when it comes to border cases.

## 6 Conclusion

We combine SWA with the CheckList methodology to explore the effects of SWA on the robustness of a BERT-based model (ALBERT-large) on different random seeds and apply it to a sentiment analysis task. To understand how SWA affects the stability amongst different random seeds, we analyze in-depth the results and mistakes made on the development set and CheckList test items and provide error rates, overlap ratios, and Fleiss' Kappa agreement values. While SWA is able to reduce the error rate in general amongst most of the random seeds, on the CheckList tests, there are still some capabilities that models make their own distinct mistakes on with SWA incorporated. The stability on the development set also improves only slightly. In the future, we would like to create more hand-crafted CheckList capabilities for further rigorous study. Furthermore, it could be useful to thoroughly investigate the impact of adjacency of random seeds on their error agreement.

## References

Silvio Amir, Jan-Willem van de Meent, and Byron Wallace. 2021. On the impact of random seeds on the fairness of clinical classifiers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3808–3823, Online. Association for Computational Linguistics.

Muhammad Hilmi Asyrofi, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung, Zhou Yang, and David Lo. 2021. Biasfinder: Metamorphic test generation to uncover bias for sentiment analysis systems. *arXiv preprint arXiv:2102.01859*.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel,

Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*.

Greg d'Eon, Jason d'Eon, James R. Wright, and Kevin Leyton-Brown. 2021. The spotlight: A general method for discovering systematic errors in deep learning models.

Aparna Elangovan, Jiayuan He, and Karin Verspoor. 2021. Memorization vs. generalization : Quantifying data leakage in NLP performance evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1325–1335, Online. Association for Computational Linguistics.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria. Association for Computational Linguistics.

Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness gym: Unifying the NLP evaluation landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 42–55, Online. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Hang Hua, Xingjian Li, Dejing Dou, Chengzhong Xu, and Jiebo Luo. 2021. Noise stability regularization for improving BERT fine-tuning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3229–3241, Online. Association for Computational Linguistics.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI).

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.

2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jiexi Liu, Ryuichi Takanobu, Jiaxin Wen, Dazhen Wan, Hongguang Li, Weiran Nie, Cheng Li, Wei Peng, and Minlie Huang. 2021. Robustness testing of language understanding in task-oriented dialog. 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 2467–2480, Online. Association for Computational Linguistics.

Pranava Madhyastha and Rishabh Jain. 2019. On model stability as a function of random seed. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 929–939, Hong Kong, China. Association for Computational Linguistics.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Thibault Sellam, Steve Yadlowsky, Jason Wei, Naomi Saphra, Alexander D'Amour, Tal Linzen, Jasmijn Bastings, Iulia Turc, Jacob Eisenstein, Dipanjan Das, et al. 2021. The multiberts: Bert reproductions for robustness analysis. *arXiv preprint arXiv:2106.16163*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Matthew Watson, Bashar Awwad Shiekh Hasan, and Noura Al Moubayed. 2021. Agree to disagree: When deep learning models with identical architectures produce distinct explanations. *arXiv preprint arXiv:2105.06791*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763, Florence, Italy. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. 2020. Improving bert fine-tuning via self-ensemble and self-distillation. *arXiv preprint arXiv:2002.10345*.

Ruiqi Zhong, Dhruba Ghosh, Dan Klein, and Jacob Steinhardt. 2021. Are larger pretrained language models uniformly better? comparing performance at the instance level. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3813–3827, Online. Association for Computational Linguistics.

## A   Technical Details

For model training, we make use of the HuggingFace (Wolf et al., 2019) pipeline and train the models on a single GeForce RTX 2080 Ti. We use the same hyperparameter settings as reported by Lan et al. (2020). The visualization of the learning rate schedules can be seen in Figure 3.
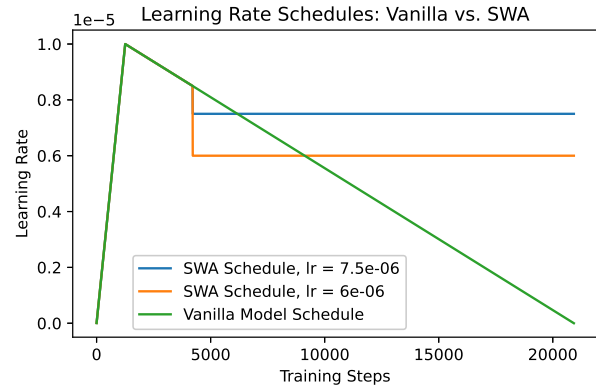


Figure 3: All the different learning rate schedules start identically with 1256 warmup steps to 1e-5. For the vanilla models (the green line) the learning rate anneals linearly to 0 till the 20935th step. This is in accordance with the hyperparameters reported in Lan et al. (2020). For the SWA models, after the second epoch, the learning rate drops to one of the specified learning rates (blue or orange lines) and stays constant.

As the HuggingFace pipeline does not provide the labels for the test set of SST-2, we match the phrases of the test set in HuggingFace with the phrases in the SST-2 dataset from the *dictionary.txt* file, downloaded from GLUE,[7] to get their phrase IDs. Then we use those IDs to extract the labels from *sentiment_labels.txt*. Every label above $0.6$ is mapped to *positive* and equal to or lower than $0.4$ is mapped to *negative*, as mentioned in the instructions of the *README.md* file. Some sentences are matched manually as they differ only in British vs. American English spelling.

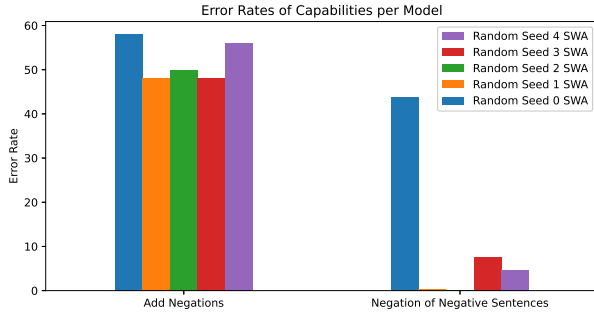## B   Results of Excluded Capabilities

For completeness, we also show the results for capabilities excluded from our analysis. For *Add Negations* and *Negation of Negative Sentences* we generated automatic test cases but the labels were not always correct upon investigation. Hence, we left these two capabilities out of the analysis.

In Table 5 we show the Fleiss' Kappa values, the error rates per capability for the vanilla and SWA models can be found in Figure 4a and Figure 4b, respectively. The variation in error rates and overlap ratios between vanilla and SWA models can be found in the Figures 5a and 5b respectively. All the results are with the five initial random seeds, *Random Seed 0* included.

---

[7] https://gluebenchmark.com/tasks
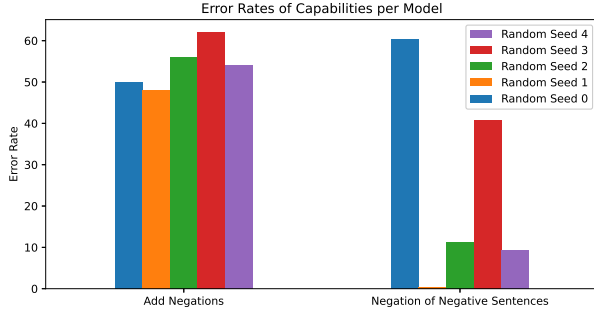
(a) Error rates of each vanilla random seed for the excluded capabilities.



(b) Error rates of each SWA random seed for the excluded capabilities.

Figure 4: Comparison of error rates of vanilla and SWA models for the excluded capabilities.

| | Vanilla | SWA | Difference |
|---|---|---|---|
| Add Negations | 0.475553 | 0.470416 | -0.005136 |
| Negation of Negative Sentences | -0.025882 | -0.066728 | -0.040845 |

Table 5: Fleiss' Kappa values of the excluded capabilities.

## C   CheckList Results with Random Seed 0

We present our results on CheckList with *Random Seed 0* as well for transparency. We again present the Fleiss' Kappa values for the CheckList capabilities in Table 6. The error rates of each capability per vanilla and SWA models can be found in Figures 6a and 6b. We also plot the variation in error rates (Figure 7a) and overlap ratios (Figure 7b).
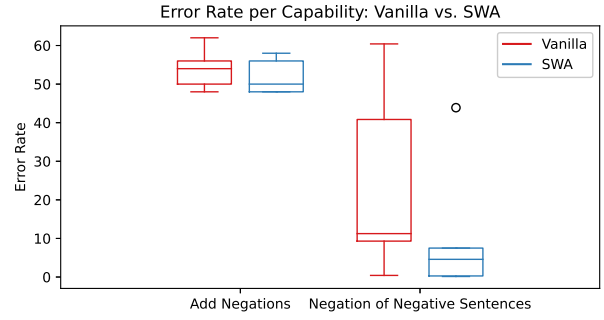


(a) Comparison of variation in error rates between vanilla (red boxes) and SWA models (blue boxes), for the excluded capabilities.
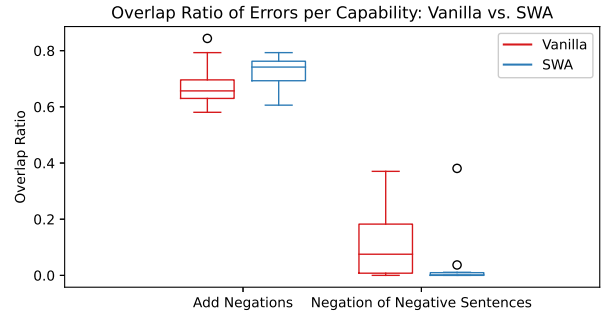


(b) Comparison of variation in overlap ratios between vanilla (red boxes) and SWA models (blue boxes), for the excluded capabilities.

Figure 5: Comparing the variation of error rates and overlap ratios for the excluded capabilities.
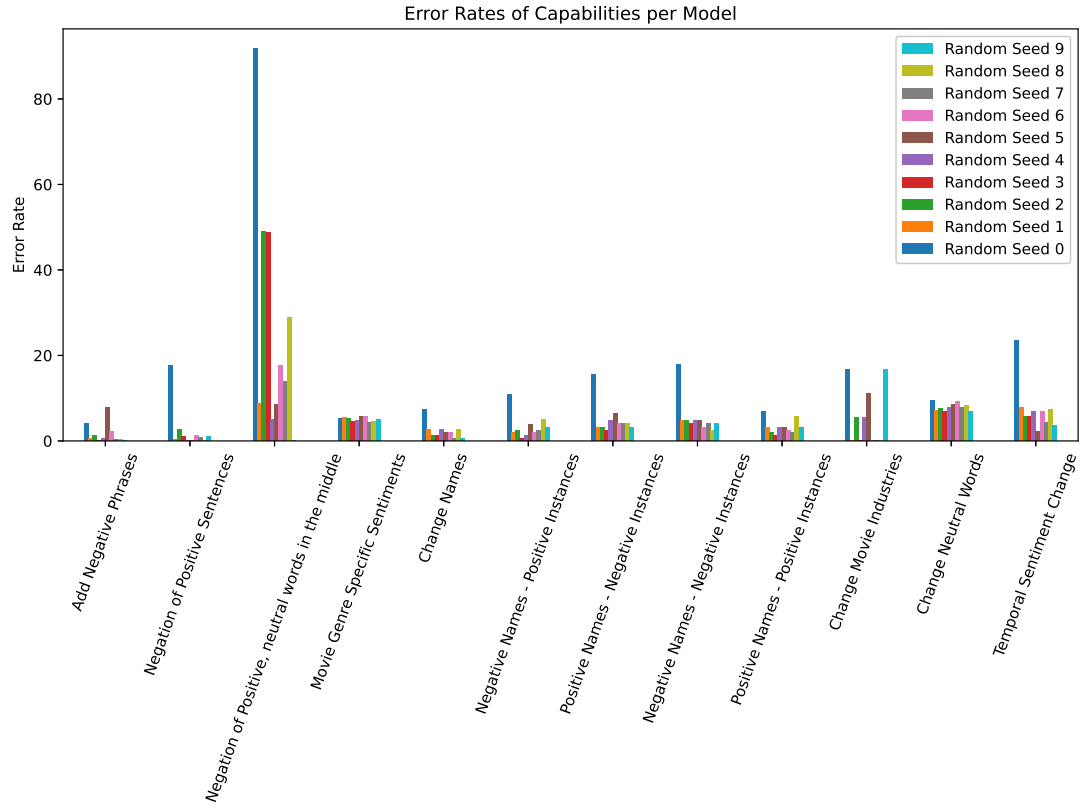
## D   CheckList Capabilities

In Table 7 we describe each CheckList capability that we test for. For perturbing capabilities such as *Negative names, Positive instances* and its other variants, we extract names from the SST-2 training set with Spacy (Honnibal et al., 2020). Due to false positives, we manually remove names that do not refer to a person, such as movie names and historical figures. Per name, we calculate the mean of labels of the instances it occurs in. This way, we can select positive and negative names to perturb test set instances with.

As reviews were predominantly about Hollywood, we also perturbed instances talking specifically about it. We compile a list of around 10 other movie industries,[8] based on how many movies are produced[9] and revenue.
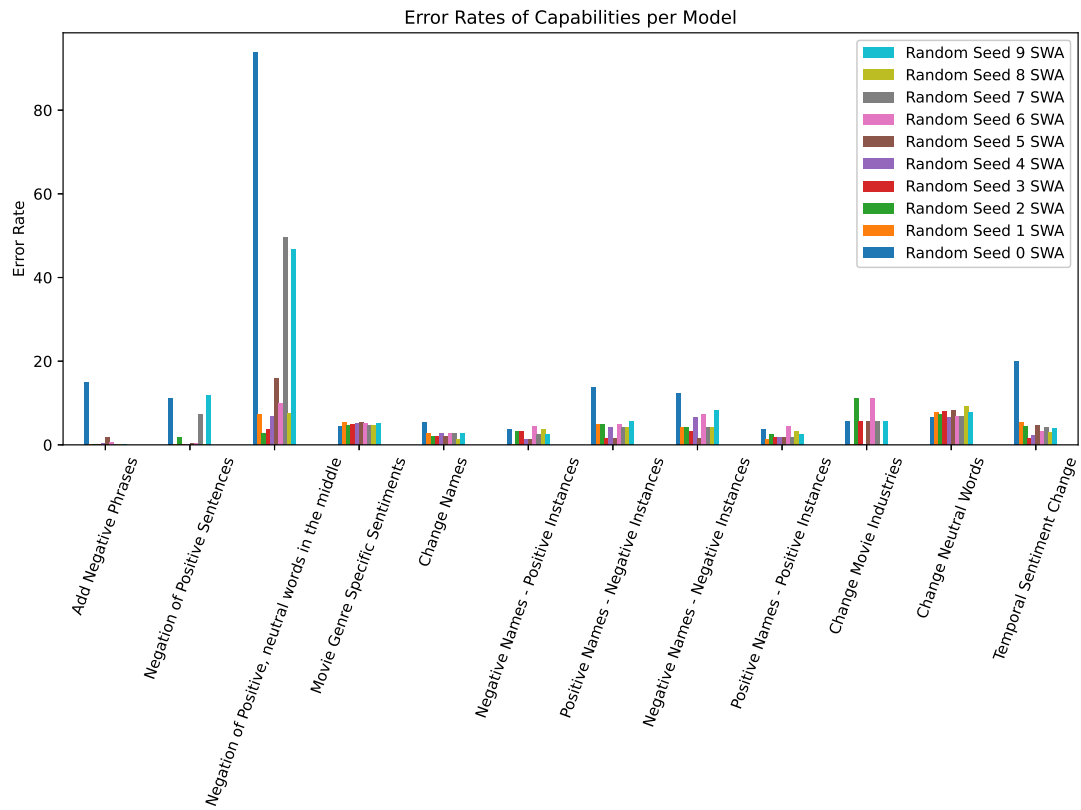
---

[8]https://en.wikipedia.org/wiki/List_of_Hollywood-inspired_nicknames
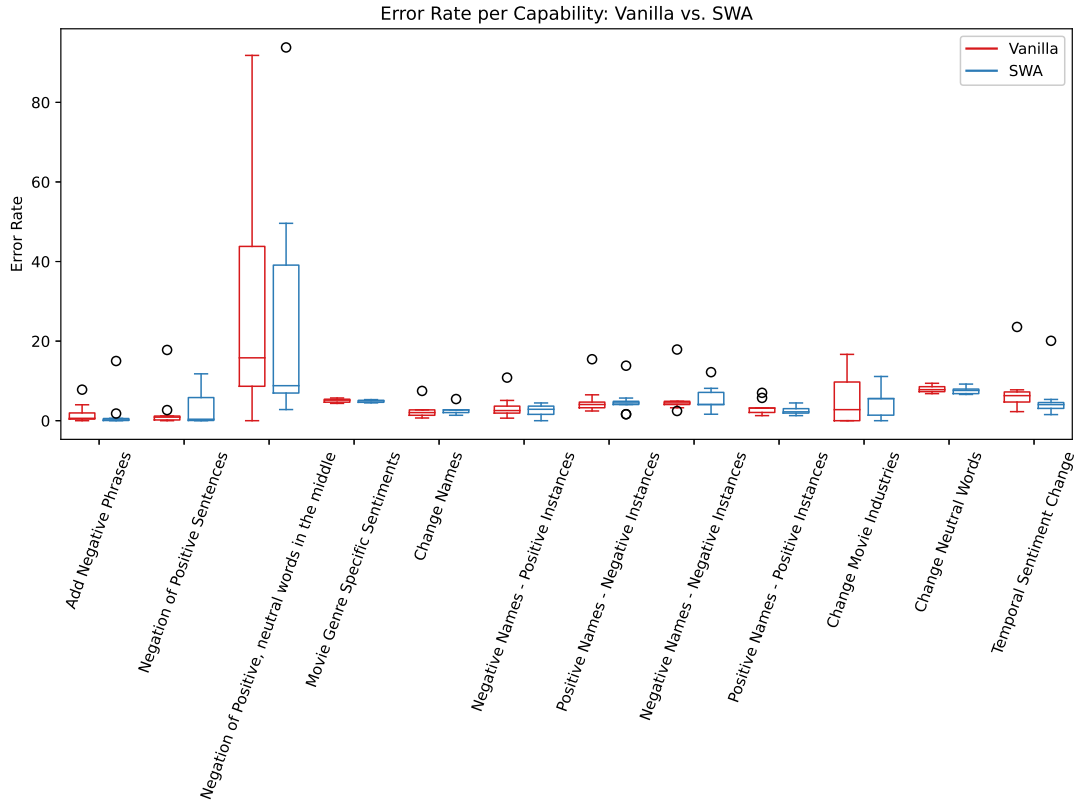[9]https://en.wikipedia.org/wiki/Film_industry#Statistics

(a) Error rates of each vanilla random seed for each CheckList capability.
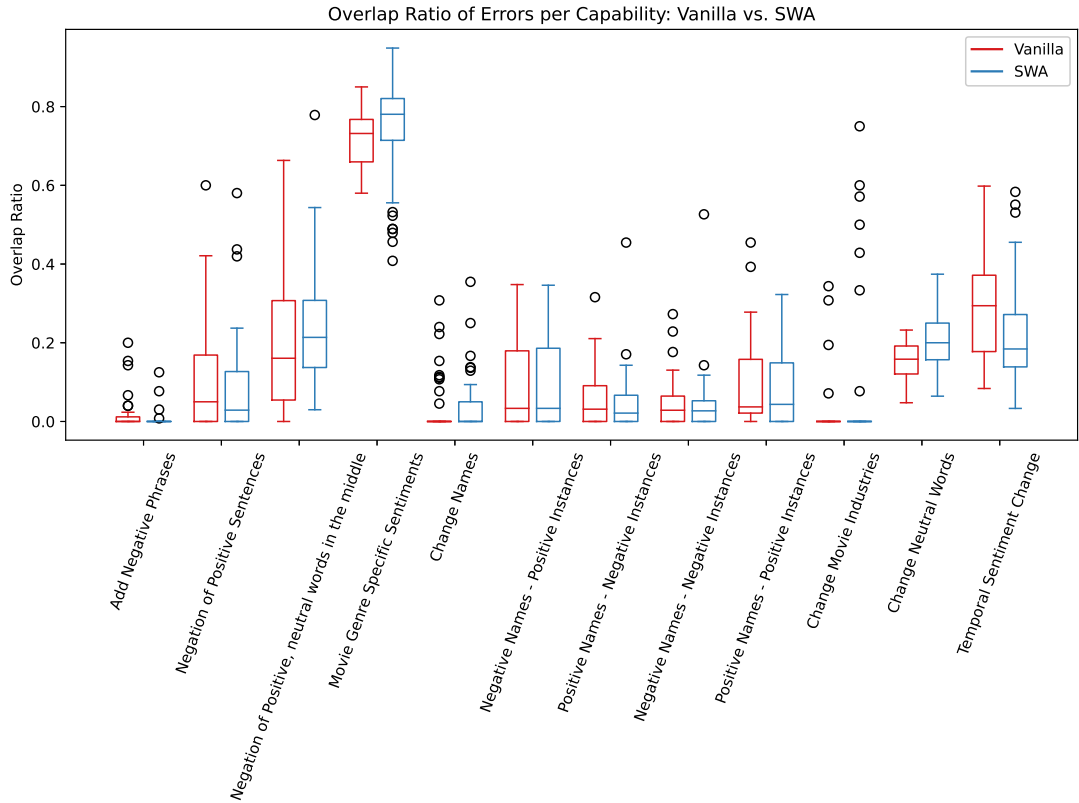


(b) Error rates of each SWA random seed for each CheckList capability.

Figure 6: Comparison of error rates per capability of vanilla and SWA models with all the 10 random seeds.

(a) Comparison of variation in error rates between vanilla (red boxes) and SWA models (blue boxes), showcased per CheckList capability. Outliers are indicated with a circle.



(b) Comparison of variation in overlap ratios between vanilla (red boxes) and SWA models (blue boxes), showcased per CheckList capability. Outliers are indicated with a circle.

Figure 7: Comparing the variation of error rates and overlap ratios per capability for vanilla and SWA models, including results from *Random Seed 0*.

|  | **Vanilla** | **SWA** | **Difference** |
|---|---|---|---|
| Negation of Positive Sentences | 0.048956 | 0.031777 | -0.017179 |
| Negation of Positive, neutral words in the middle | 0.132425 | 0.150507 | 0.018082 |
| Movie Genre Specific Sentiments | 0.583256 | 0.590209 | 0.006952 |
| Sentiment-laden Words in Context | 0.000000 | 0.000000 | 0.000000 |
| Temporal Sentiment Change | 0.335020 | 0.428743 | 0.093723 |
| Movie Industries Sentiments | -0.048387 | -0.036866 | 0.011521 |
| Change Names | -0.046527 | -0.074413 | -0.027887 |
| Negative Names - Positive Instances | -0.007656 | -0.007270 | 0.000386 |
| Positive Names - Negative Instances | -0.044586 | -0.067630 | -0.023044 |
| Negative Names - Negative Instances | -0.046272 | -0.061678 | -0.015406 |
| Positive Names - Positive Instances | 0.001146 | 0.017189 | 0.016043 |
| Change Movie Industries | -0.041426 | -0.042046 | -0.000620 |
| Change Neutral Words | 0.072667 | 0.124769 | 0.052101 |
| Add Positive Phrases | -0.058824 | -0.043796 | 0.015028 |
| Add Negative Phrases | -0.051318 | -0.021295 | 0.030023 |

Table 6: Fleiss' Kappa values of the vanilla and SWA models on the misclassifications on the CheckList tests separately, all the 10 random seeds, including outlier *Random Seed 0*. The first part of the table shows the MFT capabilities, the second part are the INV capabilities, and the third part are the DIR capabilities.

| Capability | Test Type | #Examples | Description | Original Check-List Name |
|---|---|---|---|---|
| Single Positive Words | MFT | 22 | Positive words that should be predicted *positive* (e.g. beautiful, brilliant, enjoyed). | |
| Single Negative Words | MFT | 14 | Negative words that should be predicted *negative* (e.g. hate, disliked, dreaded). | |
| Sentiment-laden Words in Context | MFT | 1350 | Sentences that contain positive or negative words about movie-related sentiments (e.g. "I despise that director"). | |
| Temporal Sentiment Change | MFT | 2152 | Sentences that contain a change in sentiment over time regarding movies. Expectations are that depending on the final sentiment, the predictions are positive or negative. | used to, but now |
| Negation of Positive Sentences | MFT | 1350 | Sentences with a negation of a positive sentiment. Predictions should be negative. | Simple negations: negative |
| Negation of Positive, neutral words in the middle | MFT | 500 | Negation of a positive sentence, with a neutral sentiment in the middle. Should be negative. | Simple negations: not negative |
| Movie Genre Specific Sentiments | MFT | 736 | Sentences with sentiments that are specific for a movie genre, such as horror movies being scary (positive). Based on the sentiment, prediction should be negative or positive. | |
| Movie Sentiments | MFT | 58 | Simple sentences about movies. Based on the sentiment, could be predicted as negative or positive. | |
| Movie Industries Sentiments | MFT | 1200 | Sentences that talk about movie industries in a positive and negative way and the predictions should be accordingly. | |
| Change Neutral Words | INV | 500 / 3846 | Neutral words are changed with BERT, such as "that", "this", "of", etc. Expect the prediction to stay the same. | change neutral words with BERT |
| Change Names | INV | 147 / 1617 | Changes the name in a review. Expectation is that the prediction stays the same. | |
| Negative names - Positive instances | INV | 157 / 1727 | Change names in positive reviews with names from training set that were only in negative reviews. Expectation is that predictions stay the same. | Polarizing Negative Names - Positive Instances |
| Positive names - Negative instances | INV | 123 / 1353 | Change names in negative reviews with names from training set that were only in positive reviews. Expectation is that predictions stay the same. | Polarizing Positive names - Negative instances |
| Negative names - Negative instances | INV | 123 / 1353 | Change names in negative reviews with names from training set that were only in negative reviews. Expectation is that predictions stay the same. | Polarizing Negative names - Negative instances |
| Positive names - Positive instances | INV | 157 / 1727 | Change names in positive reviews with names from training set that were only in positive reviews. Expectation is that predictions stay the same. | Polarizing Positive names - Positive instances |
| Change Movie Industries | INV | 18 / 252 | Changes a movie industry (Hollywood) to another one in movie reviews. After changing, the prediction should stay the same. | |
| Add Positive Phrases | DIR | 500 / 5500 | Add a positive phrase to a review, expecting the prediction confidence for positive to go up. | |
| Add Negative Phrases | DIR | 500 / 5000 | Add a negative phrase to a review, expecting the prediction confidence for negative to go up. | |

Table 7: Overview of all the CheckList capabilities, with the test type, amount of examples, and a description of the capability provided. For clarity, we also provide the name of the capabilities originally used in the code, if they are not the same.