Enhancing NER Performance: Exploring the Impact of Introduced Noise on Twitter Text Data

Bozhidara Pesheva bope@itu.dk
Cristina Avram cria@itu.dk
Gabriela Zhelyazkova gazh@itu.dk
Lili Raleva lilr@itu.dk

Link to our Github repository[1]

Abstract

This report synthesizes the process our group went through to research how the introduction of perturbations influences the robustness of Named Entity Recognition (NER) models. We worked with a pre-trained BERT model, which had an added classification layer on top. After determining the extent of alteration of the train and test datasets, we fine-tuned five different models and tested each one on multiple test sets. To evaluate the performances, we tracked the test f1-scores of the different models and plotted them. Overall, the more perturbations the test sets had, the worse the performance of all models became. Surprisingly, the more and noisier data the NER-tagger was trained on, the better its predictions got. Regardless of the declining test performance on noisier data, noisier models did perform better on all test sets.

1 Introduction

To carry out the task of Named Entity Recognition, we decided on using a pre-trained BERT model from Hugging Face - DistilBERT-base-uncased[2], pre-trained on a dataset consisting of 11,038 unpublished books and English Wikipedia. Our group would later fine-tune the model using the TweeBank dataset. Since most of the text on social media is produced by regular users, the language used is mainly colloquial.

However, mistyped words are a common sight in online communication. This motivated us to look deeper into NER tagging in the context of social media posts. Our group wanted to explore the possibility of making a NER-tagger more robust to misspellings. To do that, we introduced different amounts of perturbed words in the training and test datasets used for fine-tuning and testing the pre-trained BERT model. Our research question is as follows: How does the introduction of perturbations in the training set impact the robustness of a NER-tagger used on Twitter data?

2 Related Work

Upon inspecting different datasets we could use for our project, we found Jiang et al.'s TweeBank paper[3] where the authors introduce four-class named entities to text data, consisting of tweets (blocks of text, published on the Twitter.com platform). In the Error Analysis section of the aforementioned research paper, the authors identified some difficulties regarding the NER-tagger - words sustaining typos and irregular capitalization did not seem easy to detect as entities. The researchers then go on to propose the introduction of perturbed text data during the model training. We followed up on this idea and decided to investigate what effect a perturbed train set would have on the robustness of a NER model.

The work of Moradi and Samwald [4]provided valuable insights, particularly in its fourth section, which discussed various character-level perturbations. The authors used them to evaluate the robustness of natural language processing systems. These insights allowed us to choose the types of character-level perturbations which we will be implementing to artificially generate noisy data.

Part of our methods were also inspired by the work of Jason Wei and Kai Zou[5], focusing on data augmentation. Their report suggests that simple data augmentation operations can boost performance on text classification tasks. Therefore, we implemented concepts and suggestions, introduced in the paper in our noise insertion functions. Wei

[1] https://github.itu.dk/bope/NLP_project2024.git
[2] https://huggingface.co/distilbert/distilbert-base-uncased
[3] Hang Jiang, Yining Hua, Doug Beeferman, and Deb Roy. 2022.
[4] Milad Moradi and Matthias Samwald. 2021
[5] Jason Wei and Kai Zou. 2019

and Zous paper proposes including the new altered sentences in the dataset, leading to its expansion, an approach that we also decided to implement.

3 Data

The Twitter data we are working with consists of words alongside their true label, where the possible labels are: "O" for words that are not named entities, "PERS" for persons, "LOC" for locations, "ORG" for organizations, and "MISC" for named entities which do not fall in any of the categories as mentioned above. Due to some named entities consisting of multiple words, the beginning of a named entity is denoted with the prefix "B", while the continuation has the prefix "I" (see example Table 1).

Sentence							
RT	**@USER1274**	**:**	**High**	**School**	**Musical**	**3**	**URL167**
O	O	O	B-MISC	I-MISC	I-MISC	**I-MISC**	O

Table 1: Example of B and I prefix usage

The TweeBank dataset is split into a training set with 1639 tweets, an evaluation set with 710 tweets, and a test set with 1201 tweets stored in the bio file format.

We noticed that the data contains minimal grammatical errors, but it contains many abbreviations (for example, "TV", "ATM", "DMs", "FBI"), internet terminology (for example, "sadting", "hustling"), colloquial expressions (for example, "hit up the gym", "hits your inbox") and intentional misspellings and letter repetitions (for example, "heyy", "sorrryyz", "textinnggg", "dudeee").

However, we must acknowledge the challenges that the TweeBank dataset presents. There seem to be very little unintentional misspellings and grammatical errors in the dataset, despite their prevalence in real-world data. According to linguistic research on the topic, the most common types of errors at a character level include forgotten characters (deletion), swapped characters, added characters (insertion), and incorrect or missing capitalization [6]. This complexity adds a layer of difficulty to our analysis, which we are prepared to tackle.

4 Methods

For the task, we decided to add different noise levels to the training and test data. The noise we add is random insertion, deletion of a character in a word, and random swapping of 2 letters. We only apply the perturbation functions on words and ignore tokens that only contain punctuation, symbols, or emojis.

The insertion function, a key component of our data perturbation process, involves adding a specific number of random English alphabet letters to a percentage of words in a number of sentences. Similarly, the swapping function randomly exchanges two letters in a number of words and sentences. Lastly, the deletion function removes a letter from a percentage of words and sentences, including the possibility of deleting the first and last letter or any letter in between. These functions, when applied strategically, introduce controlled variations in the dataset.

The functions are standardized to take the same hyper parameters as input, the number of characters to modify, the percentage of words in a sentence to perturb, and the percentage of sentences in the overall dataset.

We have decided to perturb only a single character in each word (or, in the case of the swap function, a single pair of characters).

In order to decide the percentage of words to modify in each sentence, we plotted a histogram of the sentence lengths in Fig. 1, and noticed that most sentences in the training set are around ten words long. Due to the short length of the sentences, a tiny percentage of perturbed words in each sentence would result in no changes or only a small perturbation. Our group decided on altering 15% of the words in a sentence, since in this way we would have at least 2 words changed on average.

We experimented with different amounts for the percentage of perturbed sentences, which resulted in different levels of added noise to the data. From the training set, we sample different percentages of the sentences: 5%, 10%, 20% and 30%.

To augment the test and train data, perturbed sentences were added to their respective datasets, as opposed to being overwritten. This decision was made due to the small number of sentences and to guarantee that there are the same number of non-altered sentences in each training set to ensure that the models trained on different training sets are comparable.

Model implementation, tokenization, and training are done using HuggingFace's transformers library and the PyTorch framework. We pre-process the data by tokenizing the sentences using DistilBERT's pre-trained subword tokenizer.

[6] Milad Moradi and Matthias Samwald. 2021

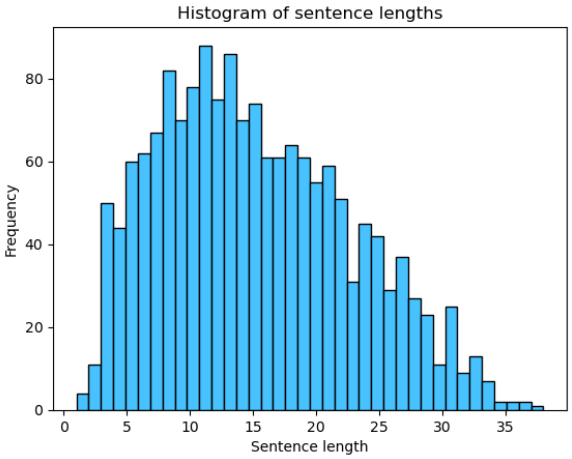

Figure 1: Frequency plot of the sentence lengths in the training set

Name	Train set size	Test set size
Set 0%	1639	1201
5%	1720	1261
10%	1802	1321
20%	1966	1441
30%	2130	1561

Table 2: Number of sentences in each set

The data that we used to train the models is the training set without added noise, as well as the perturbed training sets with 5, 10, 20, and 30% of noisy sentences added. With each training set, we trained a NER tagger that uses the language model DistilBERT-base-uncased [7], with a linear layer for classification.

In order to mitigate the risk of overfitting, we trained the models using the EarlyStopping callback with a patience of 2, meaning that after 2 epochs in which our chosen stopping criteria metric would not improve, the training would be stopped, saving the best model so far. We chose the validation set span-f1 score as our stopping criteria, as it is one of the most relevant evaluation metrics for a NER-tagging task. Initially, we used 3 epochs of training with no early stopping, which are the default parameters for the Trainer from HuggingFace, but we noticed that at the end of training our metrics were still low, and improving at each epoch, so we hypothesized that our model could further be trained to improve its performance. The hyperparameters of our model can be seen in Table 3.

[7] https://huggingface.co/distilbert/distilbert-base-uncased

Training Parameter	Value
Number of training epochs	5
Evaluation metric	validation span-f1
Learning rate	2e-5
Optimizer	Adam
EarlyStopping patience	2

Table 3: Training parameters

5 Results

After getting the 5 models trained on data with different levels of noise, we predict with each of them using the different test sets. Results of the span-f1 are shown in table 4. At first glance, the largest difference in the metric is less than 10%, so we decided to also visualize the performance of each model in the line chart on Figure 2 for readability, in addition to the results table.

	Models				
Test set	**M0%**	**5%**	**10%**	**20%**	**30%**
Set 0%	0.518	0.539	0.543	0.556	0.557
5%	0.515	0.533	0.543	0.554	0.551
10%	0.506	0.525	0.531	0.545	0.546
20%	0.501	0.524	0.532	0.538	0.537
30%	0.492	0.509	0.511	0.528	0.528

Table 4: Performance of the models on the different test sets

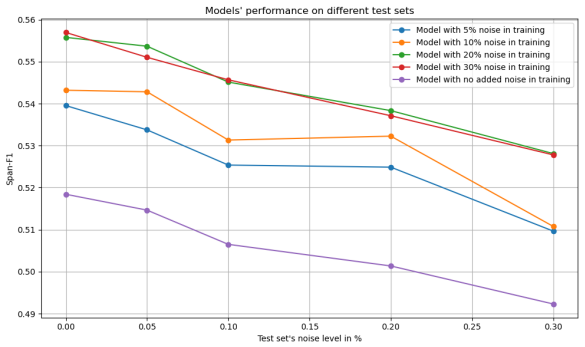

Figure 2: Visualisation of the performance of the models on the different test sets

6 Analysis

6.1 Naming conventions

For the remainder of this section we will be using several naming conventions in our analysis. The term "noise" describes words with perturbations, such as missed, added, and swapped letters. We refer to the different models depending on the amount

of altered data added during the training of the aforementioned models. For example, the "20% model" is the model trained on the data with 20% added noise, etc.

6.2 Model comparison

The graph provides a view of the f1-scores derived from our diverse testing scenarios. Notably, the models trained on data with 20% and 30% added noise consistently outperformed the rest, while the 'clean' model, trained on data without additional noise, demonstrated the poorest performance. This trend highlights the importance of noise in the training data, as the more noise is introduced, the better the models perform on the original test set. The increase in dataset size due to more perturbations leads to longer training, which might explain why the noisiest models perform well on both noisy and clean data.

As observed, the noisier the test set gets, the more the performance of all models decreases. Nevertheless, the robustness of the models increases by augmenting the training sets with sentences, containing perturbations. This can be seen by following the red line, which displays the f1-scores of the 30% model. It is the smoothest line on the chart, signifying that the performance does not suffer abruptly at any point, but rather slowly and consistently declines.

Determining the model that performs the best can be challenging if we only take the visualization into account. Thus, we checked the scores from the tests: the 30% model gave the highest f1-score at 0.557 when tested on the clean data, followed by the 20% model.

When we averaged all the test f1-scores, the 20% model prevailed over the 30% one by a bit the former's mean f1-score stood at 0.544, while the latter had an average of 0.543.

The clean model was the worst-performing one out of the set. Its highest performance was 0.518 when tested on the clean test dataset. This could partly be due to the fact that it is the model trained on the least amount of data. Identical to all other models, the clean model's f1 score kept on dropping as more perturbations were introduced into the testing data, as seen on the Table 4.

6.3 Class-specific performance

We decided to investigate whether there is a trend in the model's ability to predict individual named entity categories. When we look at the f1 score of each class we observe that Person is always the entity with highest f1 (around 0.7), followed by Location, Organisation and, lastly, Miscellaneous. Details for the 10% model can be seen in Table. 6 and Figure 3 in Appendix. We hypothesized that this could be due to the nature of human names they are often unique and easily discernible and some might have specific prefixes and suffixes, which make them easier to detect. Location is a similar case most locations have unique names, which are hardly used out of context. On the other hand, under the Miscellaneous label we have a collection of all different types of entities, which may consist of words with no specific meaning, when used out of context, thus making them harder to spot. We can also take the training data into account the amount of named entities present in the datasets were, potentially, not enough to provide sufficient training for all different named entities. Some entities were more present than others, consequently leading to a less accurate detection.

The model trained on the data with 10% added noise shows an interesting pattern, as it is the only one that doesn't show a decline in performance when more noise is introduced in one of its prediction settings - 10% to 20% noise in test sets. We observed a sample of sentences with errors in the spans' predictions. Pattern shows that the two named entities that are most often confused for each other are Organisation and Miscellaneous, but in general the most frequent error type is NE→O. Examples can be seen in Table 5.

Sentence						
MISCELLANEOUS						
#NowPlaying	**Get**	**Another**	**Boyfriend**	**Of**	**@USER1659**	
O	B-MISC	I-MISC	I-MISC	O	O	
O	O	O	O	O	O	
LOCATION						
North	**Ends**	**getting**	**a**	**theatre**	😍🤘	**URL819**
B-LOC	I-LOC	O	O	O	O	O
O	O	O	O	O	O	O
ORGANIZATION						
Guess	**Who**	**"**	**American**	**Woman**	**"**	
B-ORG	I-ORG	O	B-MISC	I-MISC	O	
O	O	O	B-MISC	I-MISC	O	
PERSON						
RT	**@USER1897**	**park**	**bom**	**high**	**note**	URL733
O	O	B-PER	I-PER	O	0	O
O	O	O	O	O	O	O

Table 5: Difference between true and predicted tags

7 Discussion

Our study provides insights into how adjustments to training data can enhance Named Entity Recognition (NER) performance, particularly for social media where text is often laden with errors like random insertions, deletions, and character

swaps. By introducing these types of controlled disturbances into our training datasets, we've improved the robustness of NER models, making them more suited for use on social media platforms.

Additionally, it's important to note that both the training and test datasets were altered by us, introducing errors into a clean dataset sourced from TweeBank. This method ensures that the test conditions closely replicate the training environment, providing a robust framework for evaluating the models' effectiveness against social media text errors.

However, we acknowledge that this is not an accurate representation of misspellings in real social media data, as our functions randomly select words and characters to modify, which is not entirely the case in real life. To more accurately mimic real world misspellings, we would have to take into account multiple factors, such as keyboard layouts, language-specific grammar, word complexity, etc.

However, our findings also shed light on a critical issue. While models trained with noisier data showed better performance on clean and slightly noisy test sets, their performance declined when faced with higher levels of noise. This discrepancy underscores a gap between our training conditions and the diverse errors found in actual social media text, indicating a need for training approaches that more accurately mimic real-world complexities.

It is also worth noting that by adding the altered sentences to the data instead of replacing the originals, we increase the training data for each consecutive model. Therefore a question could be raised whether the performance improvement we are observing is solely because of the increased noise in training, or the larger and more diverse training set.

Typographical errors greatly impact applications such as sentiment analysis and user engagement tracking. Future research should explore using higher levels of noise and incorporating various error types, including semantic and grammatical mistakes, to enhance model generalization further.

8 Limitations

Our study, despite its limitations, provides significant insights into enhancing NER systems for social media texts. One of the main limitations is the relatively small size of the TweeBank dataset used for training and testing. The limited amount of data can restrict the model's ability to learn and generalize effectively across the broad spectrum of language use found on social media. A larger dataset would help the model capture various linguistic features and typographical errors, potentially improving robustness and accuracy.

The methodology for introducing noise into the dataset, while diverse, remains quite synthetic. It fails to fully capture the complexity and patterns of mistakes made by real social media users. That might be problematic when modifying the test set, as testing models should always be done on data that is as close to the real-world as possible. A more realistic approach, considering factors such as keyboard layouts, language proficiency, and typing speed, could bridge the gap between our simulated conditions and actual social media interactions.

The choice of noise levels (e.g., 20% and 30%) and their application may not accurately reflect the true nature of noise in social media texts. The percentages chosen and the uniform application of noise across different texts do not consider the variable nature of errors across different users and contexts. This "one-size-fits-all" approach to noise introduction might not be the most effective way to train models that need to operate in highly diverse and dynamic environments.

To overcome these limitations, future research should consider expanding the dataset by incorporating more data from TweeBank or by integrating datasets from other sources that capture a broader range of social media interactions. Additionally, developing more sophisticated methods for simulating typographical errors could significantly enhance the realism of training datasets. This might involve machine learning techniques to analyze error patterns based on user demographics or typing behavior, creating a more accurate and representative training environment.

In summary, while our study advances the field of NER for social media text, these limitations highlight the need for more nuanced modeling and dataset preparation approaches to ensure that NER systems are genuinely effective in real-world applications.

9 Conclusion

After conducting multiple experiments, the following conclusions can be drawn - the overall differences between the models performances are rather small, but a trend is noticeable. The augmentation of the train datasets with perturbed sentences did improve the span-f1 test scores of the base

model. Yet, the 20% and 30% models displayed similar scores, with the latter having a slightly smoother rate of change. Regarding the augmentation of the test sets - all models' performances worsen when faced with data containing more typos. In regards to individual label tagging - the Person NEs were the easiest to detect, while the Miscellaneous ones were the hardest.

10 References

GitHub Repository. NLP Project 2024. https://github.itu.dk/bope/NLP_project2024.git, 2024.

Hugging Face. distilbert-base-uncased. https://huggingface.co/distilbert/distilbert-base-uncased.

Hugging Face NLP course. https://huggingface.co/learn/nlp-course/chapter0/1?fw=pt.

Nicoletta Calzolari et al. Proceedings of the 13th Language Resources and Evaluation Conference. https://aclanthology.org/2022.lrec-1.780/, 2022.

Milad Moradi and Matthias Samwal. Evaluating the Robustness of Neural Language Models to Input Perturbations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*. https://aclanthology.org/2021.emnlp-main.117/, 2021.

Jason Wei and Kai Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP 2019)*. https://aclanthology.org/D19-1670/, 2019.

11 Appendix

11.1 Compliance with Generative AI Policy

To comply with Generative AI policy we list the reasons we used AI in this project: ChatGPT was used for guidance in tutorials, trivial questions about LaTeX syntax and to generate Python code for creating charts and visualisations only. We used Grammarly to fix grammatical and typographical errors.

11.2 Contributions

Abstract, Introduction, Related work, Conclusion - Lili Raleva

Data, Methods - Cristina Avram

Methods, Results - Bozhidara Pesheva

Analysis - Lili Raleva, Gabriela Zhelyazkova, Cristina Avram, Bozhidara Pesheva

Discussion, Limitations - Gabriela Zhelyazkova

Even though everyone had an assigned part to write, we all discussed all parts of this report, and we all participated in the correction process. Everyone contributed equally to this project's process.

11.3 Additional results

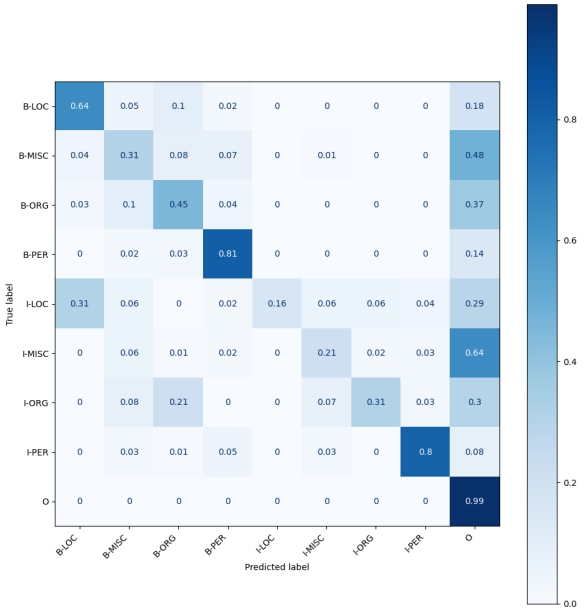

Figure 3: Confusion matrix of class predictions for the 10% model on test set with 10% noise

	clean	5% noise	10% noise	20% noise	30% noise
LOC	0.486	0.483	0.500	0.474	0.442
MISC	0.241	0.235	0.219	0.214	0.230
ORG	0.416	0.409	0.404	0.427	0.381
PER	0.730	0.730	0.705	0.693	0.692

Table 6: Results of model trained on 10% noise on the different test sets

11.4 Links to trained models

All of our trained models can be accessed from the HuggingFace hub with the following paths:
Model trained on the training set with 0% noise: https://huggingface.co/gabizh/dbbuc_OG
Model trained on the training set with 5% noise: https://huggingface.co/cria111/dbbuc_5p
Model trained on the training set with 10% noise: https://huggingface.co/gabizh/dbbuc_10p
Model trained on the training set with 20% noise: https://huggingface.co/bozhidara-pesheva/dbbuc_20p
Model trained on the training set with 30% noise: https://huggingface.co/lilzzz/dbbuc_30p