# spaCy NER Project Report


Anant Katrina (kevin.kraus@studium.uni-hamburg.de)

Kevin Kraus (kevin.kraus@studium.uni-hamburg.de)

# Task summary

1. Train a new spaCy NER model for German using the *GermaNER* dataset
2. Load the existing *spaCy GermaNER* model and update it with the *GermaNER* data
3. Evaluate the models.

# Code

First of all we preprocessed the data and re-created the sentence structure, as well as the required sentence-entity tuples for training spacy NER models:

```
def build_model_data(data):

    data = data[data.index != '#']

    data = data.iloc[:, :-1]

    data['NER'] = data['NER'].str.replace(r'\S+PER\w*', 'PER', regex=True)
    data['NER'] = data['NER'].str.replace(r'\S+LOC\w*', 'LOC', regex=True)
    data['NER'] = data['NER'].str.replace(r'\S+ORG\w*', 'ORG', regex=True)
    data['NER'] = data['NER'].str.replace(r'\S+OTH\w*', 'MISC', regex=True)

    data_processed = []
    sentences = []
    block_sent = []
    block_ents = []

    round = 0
    current_pos = 0

    for row in data.iterrows():
        if int(row[0]) == 1 and round != 0:
            current_pos = 0
            sent = ' '.join([str(word) for word in block_sent])
            sentences.append(sent)
            ent = {'entities': block_ents}
            train_point = (sent, ent)
            data_processed.append(train_point)

            block_sent = []
            block_ents = []

        try:
            block_sent.append(row[1].tokens)
            if row[1].NER != 'O':
                block_ents.append((current_pos, current_pos + len(row[1].tokens), row[1].NER))
            current_pos = len(row[1].tokens) + current_pos + 1

        except Exception as e:
            continue

        round += 1
    return data_processed
```

After that, we used the training function presented in practice class as basis for further work. Adding a custom function to create batches and some more little tweaks to get the training correctly, we started training on different batch size/ iteration combinations. Since training took quite long (full dataset of 29.000 rows with $batchsize = 400$ and $iterations = 5$ took about an hour), with no significant reduction of loss, we constraint the training data to 5000 rows, for the sake of conducting more different hyperparameter experiments. Worth to mention, it could definitely be that we have a design flaw in our code, but since our team only consisted of two people, we had no time to dwell on that for too long.

To keep it short, here are only small code chunks, for example our custom batch creator:

```
def get_batches(l, n):
    for i in range(0, len(l), n):
            yield l[i:i + n]
```

Lastly, we had to re-assemble the structure of the initial data, to match with the perl's script required data form. For that reason, we created another function, to reconstruct the required structure:

```
  print('Started transformation for perl evaluation')
    batches = get_batches(test, batch_size)
    for batch in batches:
        bnum += 1
        print(f'Batch number {bnum} of {blen}')
        for txt, _ in batch:
            doc = nlp_new(txt)
            for ent in doc.ents:
                ent_map = {ent.text: ent.label_}
                entities_map.update(ent_map)

    for token in tokens:
        fill1.append('O')
        fill2.append('O')
        if token in entities_map.keys():
            ner = entities_map.get(token)
            if ner not in re_mapped_entities:
                ent = 'B-' + ner
                re_mapped_entities.append(ent)
            else:
                ent = 'I-' + ner
                re_mapped_entities.append(ent)
        else:
            re_mapped_entities.append('O')

    re_mapped_entities = [re.sub(r'(\S+)MISC\w*', r'\1OTH', ent) for ent in re_mapped_entities]

    print('Finished transformation')
```

It simply reconstructs the required structure based on the predictions with the updated *gerNER mode* and the initial test dataset.

# Results

> 💡 DISCLAIMER: As already mentioned we had issues with training time. Hence, we reduced the training data with a random part of the full data ($seed(1)$) to achieve more time for trying and testing different variations of the model. Nevertheless, there are two models trained on the full data set.

## Model Comparision

In the following, we'll compare the two *spaCy models,* one based on a blank model and the other on a updated version of the existing *spaCy GermaNER*. We documented only six models, for the sake of shortness.

For the comparision we used 5000 rows (~16%) of test data, with following hyperparameter settings:

```
# Setup 1
batch_size = 400
iterations = 5
dropoutrate = 0.2
rows = 5000
```

| *metrics (strict, combined eval)* | Blank model | Updated model |
|---|---|---|
| Accuracy | 93.83% | 88.05% |
| Precision | 13.28 % | 8.39% |
| Recall | 23.91 % | 32.79% |
| FB1 | 17.08 | 13.36 |

Training **without entity** normalisation was outperformed by the one with entity normalisation.

```
# Setup 2
batch_size = 200
iterations = 2
dropoutrate = 0.2
rows = ALL
```

| *metrics (strict, combined eval)* | Blank model | Updated model |
|---|---|---|
| Accuracy | 93.15% | 57.33% |
| Precision | 14.29% | 2.12% |
| Recall | 29.72% | 28.02% |
| FB1 | 19.30 | 3.94 |

One could assume, that the updated model is beneficial in terms of a) computation time (no full training required) and b) (probably) performance (in terms of metrics). Concluding one can say, the way to go is to take an existing *NER* model (if available) and update it on ones specific needs.
This way, it is possible to quickly setup up a custom *NER* model, with few lines of code. The only disadvantage we could notice, that one has to be conform with the platforms data standards. Thus, it requires some time and effort to transform available data into the correct format. But after this is done, everything else builds up quickly.

Unfortunately, we could not prove our assumption based on our results. There may be several reasons, why either the training had mistakes or the format of our .perl input was faulty. We'd be glad for feedback on the code, to understand where our possible error lies.

Link to drive with models:

https://drive.google.com/drive/folders/1y96Viv_qJSv6a17KTo4jdD_CQpg6SvLG?usp=sharing