Natural Language Processing

Learning Portfolio 5

universität innsbruck

Data Set

The data set is 2022 financial transaction data retrieved from my personal bank account and credit card.

The goal of the data set is to categorize expenses and use the categorization to create an expense report.

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, 271 to 376
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    500 non-null    object
 1   labels  500 non-null    object
 2   betrag  500 non-null    float64
 3   datum   500 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 19.5+ KB
```

Data Labeling

Data was uncategorized when retrieved. I wanted to use a supervised learning approach. Thus, I labeled the data manually using a small python script.

```
{'Umsatz abgerechnet und nicht im Saldo enthalten': 'Ja', 'Wertstellung': '02.01.2023', 'Belegdatum': '31.12.2022', 'Beschreibung': 'PAYPAL *PHILFRIE9235314369001',
Select a category:
1              : Groceries          2              : Dining/Restaurants
3              : Utilities          4              : Transportation
5              : Rent/Mortgage      6              : Entertainment
7              : Travel             8              : Shopping
9              : Health/Wellness    10             : Education
11             : Insurance          12             : Investments
13             : Miscellaneous      14             : Hobbies
15             : Home               16             : Irrelevant
Enter the category number:
```

Data Preprocessing

- Removed all income data
- Removed duplicate information
- Test Data: 10 % - 62 entries
- Validation Data: 9 % - 56 entries
- Train Data: 81 % - 500 entries

```
data['text'] = pd.concat([bank_account_data["Auftraggeber / Begünstigter"],credit_card_data["Beschreibung"]])
data['labels'] = pd.concat([bank_account_data["Category"],credit_card_data["Category"]])
data['betrag'] = pd.concat([bank_account_data["Betrag (EUR)"],credit_card_data["Betrag (EUR)"]])
data['datum'] = pd.concat([bank_account_data["Wertstellung"],credit_card_data["Wertstellung"]])
```

```
Mounted at /content/drive
```

Next, I will be performing some data transformations. I will convert Betrag to a float, as well as datum to a date. Then, I will remove all dat
does not have text information as well as all data that has been marked as "Irrelevant".

```
[2]  data['betrag'] = data['betrag'].str.replace('.', '').str.replace(',', '.').astype(float)
     data['datum'] = pd.to_datetime(data['datum'])
```

```
<ipython-input-2-370278172a7b>:1: FutureWarning: The default value of regex will change from True to False in a
  data['betrag'] = data['betrag'].str.replace('.', '').str.replace(',', '.').astype(float)
<ipython-input-2-370278172a7b>:2: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the defa
  data['datum'] = pd.to_datetime(data['datum'])
```

```
[3]  data = data[data['text'].notna()]
     data = data[data['labels'] != "Irrelevant"]
```

```
[4]  train_data, test_data = train_test_split(data, test_size=0.1, stratify=data['labels'], random_state=42)
     train_data, validation_data = train_test_split(train_data, test_size=0.1, stratify=train_data['labels'], random_state=42)

     print("Length of training, validation and test set")
     print((len(train_data), len(validation_data), len(test_data)))
```

```
Length of training, validation and test set
(500, 56, 62)
```

Training

- Comparison of two pre-trained models: bert-base-uncased vs. finbert-pretrain
- 20 epochs training time
- Learning rate: 6e-05
- Batch size: 32
- Padding and Truncation activated
- Label2Id-Mapping

```
[9] from transformers import BertForSequenceClassification
    import evaluate

    accuracy = evaluate.load("accuracy")
    def compute_metrics(eval_pred):
        predictions, labels = eval_pred
        predictions = np.argmax(predictions, axis=1)
        return accuracy.compute(predictions=predictions, references=labels)

    training_args = TrainingArguments("/content/drive/My Drive/SE_Digital_Organizations/checkpoints/", evaluation_strategy="epoch",
                                      per_device_train_batch_size=32, per_device_eval_batch_size=32,
                                      num_train_epochs = 20, learning_rate = 6e-05)
    model = BertForSequenceClassification.from_pretrained(checkpoint, num_labels=15, id2label=id2label, label2id=label2id)
    data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

    trainer = Trainer(
        model,
        training_args,
        train_dataset=tokenized_datasets["train"],
        eval_dataset=tokenized_datasets["validation"],
        data_collator=data_collator,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics
    )

    trainer.train()
```

https://huggingface.co/bert-base-uncased
https://huggingface.co/FinanceInc/finbert-pretrain

universität innsbruck

Training

Overall, models scored almost equally for on the validation batches. However, while losses sometimes rose again for bert-base-uncased, finbert appeared to be a little more consistent. Thus, I have chosen finbert for testing.

BERT

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	1.991472	0.553571
2	No log	1.505513	0.696429
3	No log	1.173102	0.732143
4	No log	1.049907	0.714286
5	No log	0.988733	0.750000
6	No log	0.975981	0.732143
7	No log	0.901059	0.767857
8	No log	1.037390	0.750000
9	No log	1.111377	0.750000
10	No log	1.015163	0.767857
11	No log	1.011235	0.767857
12	No log	1.076835	0.803571
13	No log	1.084914	0.767857
14	No log	1.088718	0.767857
15	No log	1.131391	0.767857
16	No log	1.112833	0.767857
17	No log	1.093048	0.785714
18	No log	1.106076	0.750000
19	No log	1.111182	0.767857
20	No log	1.112820	0.767857

FINBERT

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	1.700762	0.517857
2	No log	1.212084	0.660714
3	No log	1.039800	0.714286
4	No log	1.041923	0.714286
5	No log	0.929369	0.750000
6	No log	0.940357	0.785714
7	No log	1.037811	0.750000
8	No log	1.004157	0.785714
9	No log	1.012097	0.785714
10	No log	1.129425	0.785714
11	No log	1.061307	0.785714
12	No log	1.035096	0.785714
13	No log	1.089739	0.785714
14	No log	1.101471	0.785714
15	No log	1.107210	0.785714
16	No log	1.121649	0.785714
17	No log	1.110373	0.785714
18	No log	1.102099	0.785714
19	No log	1.111193	0.785714
20	No log	1.113370	0.785714

universität innsbruck

Testing

- 93.55 % accuracy (single hit)
- 95.16 % accuracy (triple hit)

Astonishing result for 16 classes and given how small the text pieces of bank transaction are.

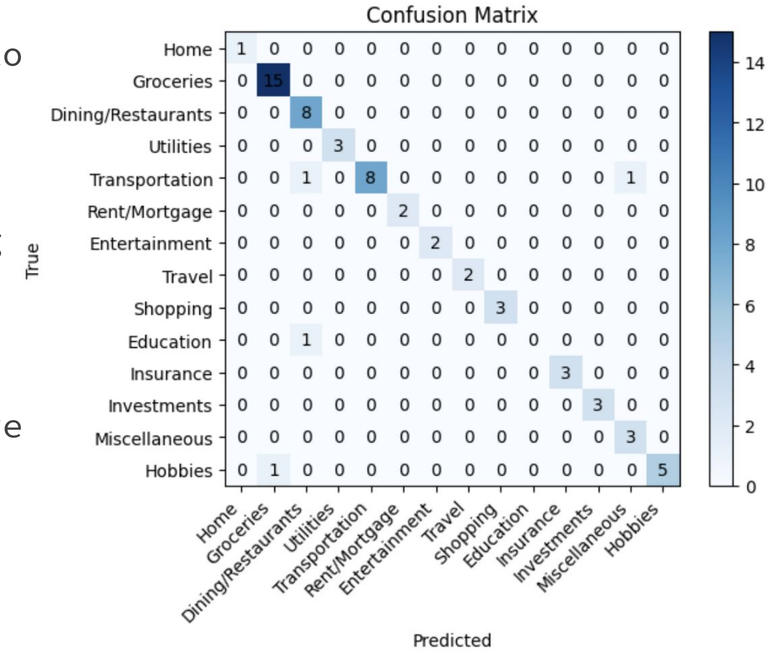
```
Single and Triple Hit Accuracy
(0.9354838709677419, 0.9516129032258065)
Category to wrong prediction single
{'Transportation': 2, 'Health/Wellness': 1, 'Hobbies': 1}
Category to wrong prediction triple
{'Transportation': 2, 'Health/Wellness': 1}
```

Confusion Matrix

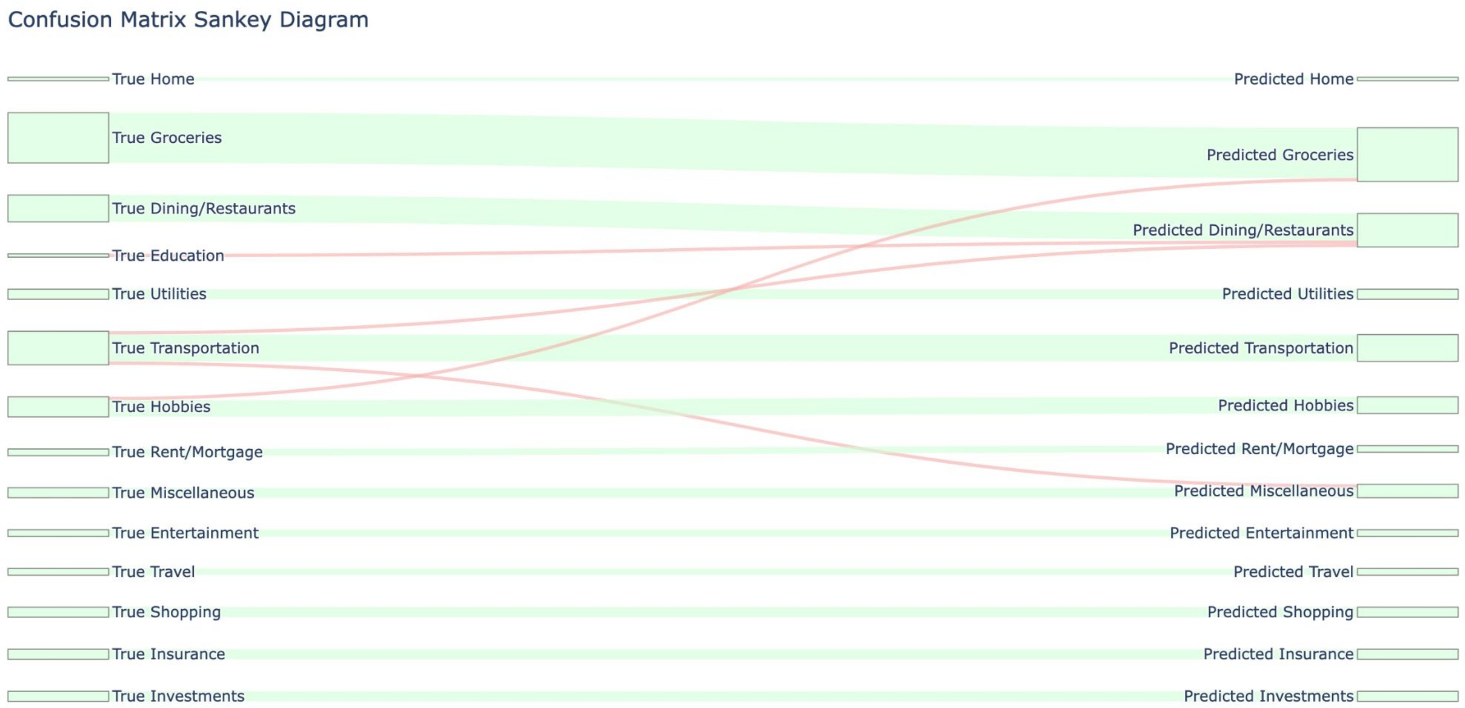
Visualizing what predictions are likely to be wrong.

The most mistaken category is “Transportation” with 2 transaction being falsely predicted.

The most likely mistake category is “Dining/Restaurants” with 2 false positive categorizations



Sankey Diagram

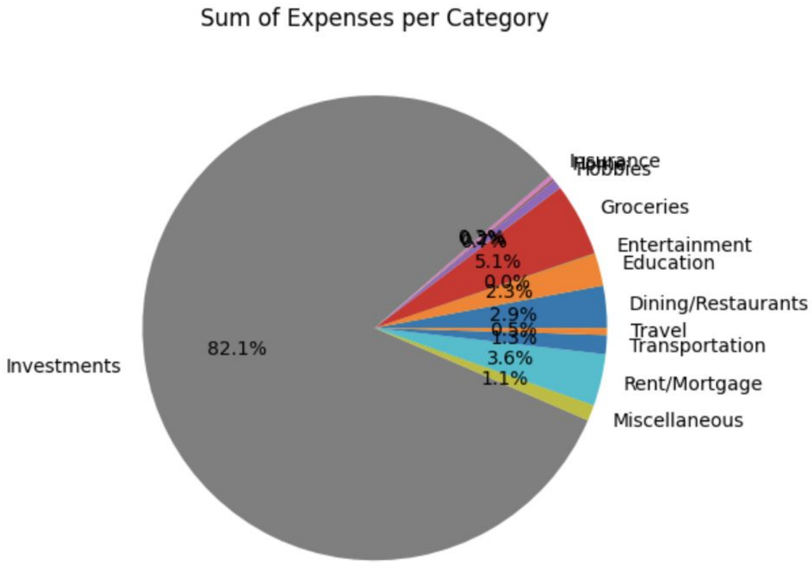


Going Live

Training data was from 2022 when I lived in Frankfurt, Germany.

I created an expense report for April 2023 and there were more predictions errors.

Data drift: My transactions have changed, because I lived in Innsbruck and was on vacation in Italy



Kontakt

Fabian Leuk
12215478
fabian.leuk@student.uibk.ac.at