Natural Language Processing

Group 15: Relation Extraction

Project Presentation

Vienna, 20.01.2023

Agenda

- Research Objective
- Implementation and Results
  - Milestone 1
  - Milestone 2
  - Advancements and Improvements
- Conclusion

Agenda

- Research Objective
- Implementation and Results
  - Milestone 1
  - Milestone 2
  - Advancements and Improvements
- Conclusion

The aim of this project was to apply the concept of **relation extraction…**

Topic 3:

Relation Extraction

which is a two-step-process:

1. **Identify** the two relevant entities ("Tag 1", "Tag 2")
2. **Classify** the relationship between those entities

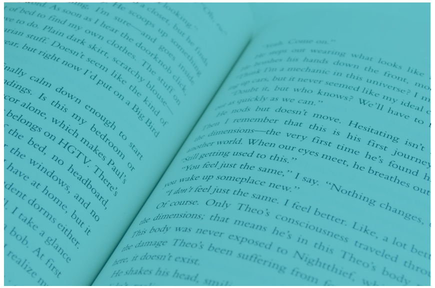
The famous **actress** arrived at the **airport**.

Tag 1 Tag 2

Relationship: **Entity-Destination**

Example

… to a **data set: SemEval-2010** was chosen to provide domain-independent data



SemEval-2010 Task 8

(Hendrickx et al.)

- **Size:** 8.000 (train) + 2717 (test) sentences
- **Target Variable:** 9 classes
- **Domain:** Unspecific/Generic

The team aimed for a **deeper understanding of relation extraction** to figure out solutions that **solve the problem efficiently**

Our targets for this project included:

Understand Problem Space

Learn and start to understand the characteristics and concepts behind relation extraction

Identify what works, and what doesn't

Implementing and comparing baseline models should enable to figure out ways to solve the problem while identifying the ones that do not work

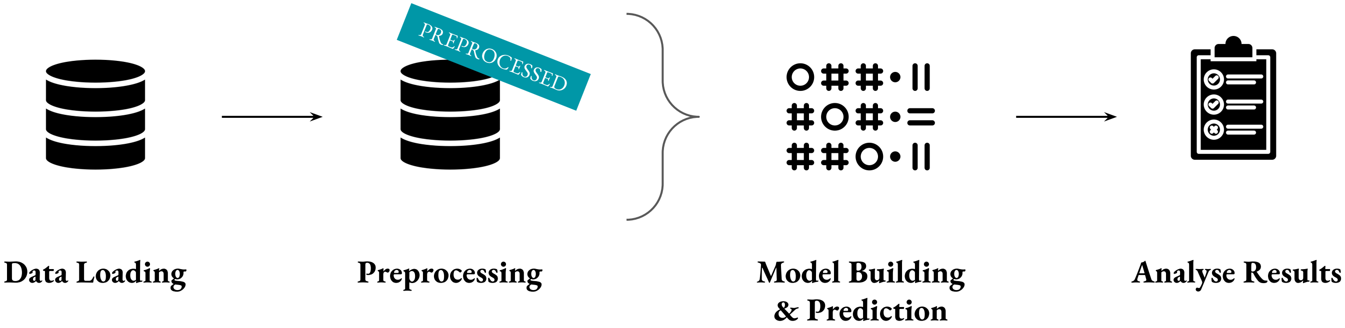
Improve Results

By identifying the weaknesses of baseline implementations, we aim for improving them

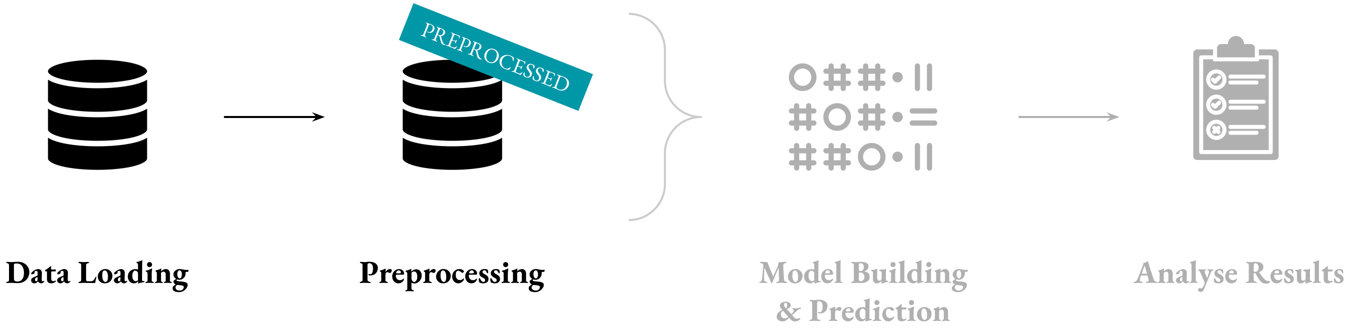
Agenda

- Research Objective
- Implementation and Results
  - Milestone 1
  - Milestone 2
  - Advancements and Improvements
- Conclusion

For milestone 1 a **standard classifier provided baseline** results for the next steps



For milestone 1 a **standard classifier provided baseline** results for the next steps



Before classifying the text sentences they are being **loaded and preprocessed** using proven methods

First the raw **text data gets loaded...**

Data Loading

```
train, test = load_data()
test_keys = load_keys_data()
```

... before being **preprocessed** for the classification stage.

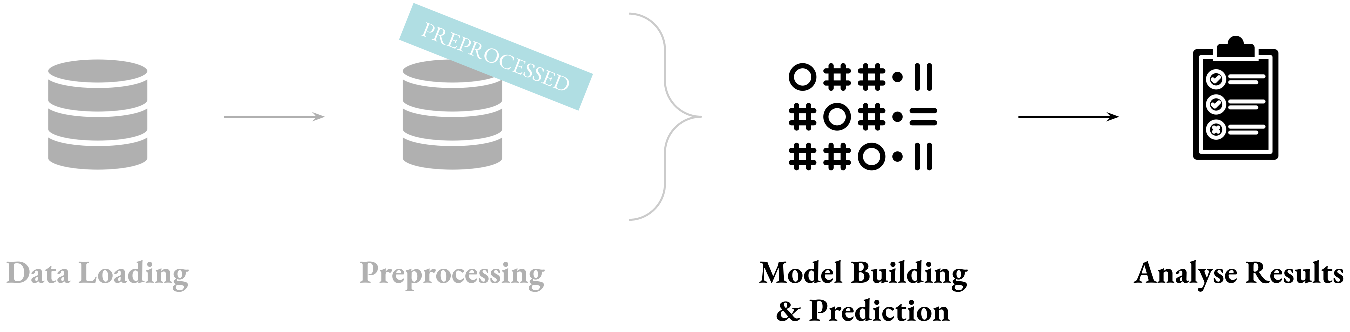
Data Preprocessing

```
# factorize relation classes
factorize_relations(train, test)

# tokenize text
CountVectorizer(ngram_range=(1,3)).fit_transform(train)

# count term frequencies including inverse document frequency
TfidfTransformer(use_idf=True).fit_transform(train)
```

For milestone 1 a **standard classifier provided baseline** results for the next steps



The result of milestone 1 with a Multinomial Naive Bayes classifier was an **accuracy score of 60.51%**

Applying **Multinomial Naive Bayes** as classifier …

Multinomial Naive Bayes

```
classifier = MultinomialNB(alpha=0.01).fit(train_X, train_y)
predictions = classifier.predict(test_X)
```

… led to the **first baseline performance results:**

`Accuracy = 60.51%`

(F1 Score: 0.78)

While the model performed well, it was shown that it the **data bias negatively impacted accuracy** and that **fewer target classes worsens** it more

Our learnings from milestone 1 include…

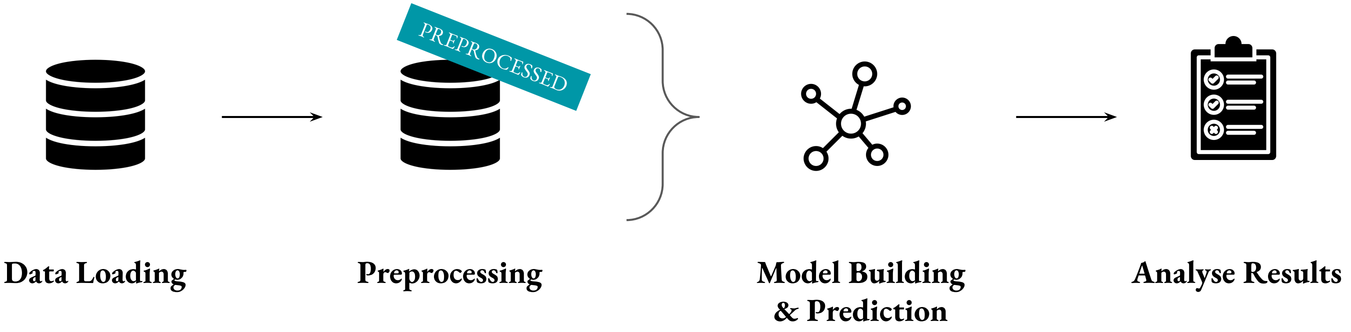
Decent results for a simple model with efficient implementation

Data bias gets translated to a **model bias**, especially in target class "other"

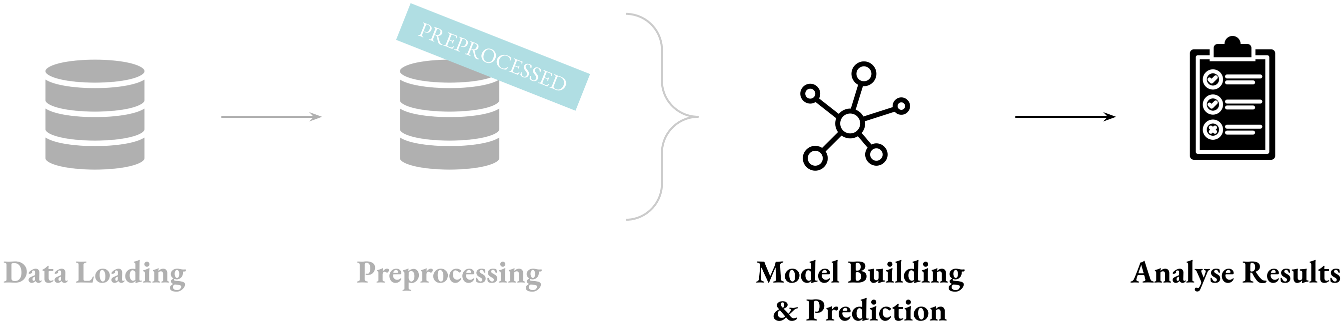
Fewer target classes leads to worse performance of the model

Agenda

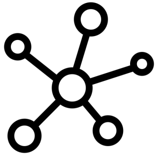
For milestone 2 a **deep learning neural network provided baseline** results



For milestone 2 a **deep learning neural network provided baseline** results



The model of choice was a **BoW classifier**, known for its simplicity

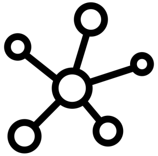


Deep learning classifier:

Bag-of-Words (BoW)

- **Simple classification model** for natural language processing (NLP) tasks
- Based on **frequency of words** in sentence
- **Disregards grammar** and **word order**

The BoW model was implemented **with one embedding layer**, two combined activation functions and avg, min and max as pooling layers



Deep learning classifier:

Bag-of-Words (BoW)

def forward()

```
embedded = self.embedding(text)
pooled = F.max_pool2d(embedded, (embedded.shape[1], 1)).squeeze(1)
return F.log_softmax(self.linear(pooled), dim=1)
```

Embedding

```
self.embedding = nn.Embedding(vocab_size, embedding_dim,
padding_idx=3000)
self.embedding.weight.requires_grad =  True
```

Activation Functions

Softmax & LogSoftmax

Pooling Layers

Average, Min, Max

With 10 epochs, the model performed with an **accuracy of 58.xx%**

Applying **BoW** as classifier …

… led to the **following results:**

BoW Classifier *

```
classifier = BoWClassifierWithEmbedding(OUTPUT_DIM, INPUT_DIM, EMBEDDING_DIM)
classifier.training_loop(train_iterator, valid_iterator)
predictions = classifier.predict(valid_iterator)
```

TRAINING

`F1 = 76.75%`

VALIDATION

`F1 = 58.17%`

* for **Epochs=10** as more epochs led to more overfitting

Milestone 2 showed that the model **performed only mediocre** - partly due to overfitting, partly possible due to the data bias

Our learnings from milestone 2 include…

During the training phase, the model started to **overfit strongly**

Data bias gets translated to a **model bias**, especially in target class "other"

Nr. of embedding layer dimensions heavily impacts runtime

Agenda

- Research Objective
- Implementation and Results
  - Milestone 1
  - Milestone 2
  - Advancements and Improvements
- Conclusion

The **weaknesses of the model of milestone 2** formed part of the basis of our next steps

To build on the two milestones, we opened up **3 project streams...**

Data Bias

↓

RESEARCH

Overfitting

↓

IMPLEMENTATION

Model Comparison

↓

IMPLEMENTATION

The **weaknesses of the model of milestone 2** formed part of the basis of our next steps

To build on the two milestones, we opened up **3 project streams...**

Data Bias

↓

RESEARCH

Overfitting

↓

IMPLEMENTATION

Model Comparison

↓

IMPLEMENTATION

The research about data bias delivered **two theoretical approaches...**

RESEARCH

Researching the problem of data bias brought
two possible approaches:

Multiple Models

Creating a corrected model by training and evaluating a series of models and ultimately control the bias along multiple dimensions

Bias to Node

Adding extra biases to the nodes of the model to counteract the bias of the data - however this approach lacks transparency

… that the team ultimately decided **not to implement** to focus on other areas

RESEARCH

Researching the problem of data bias brought
two possible approaches:

DECISION: **DO NOT IMPLEMENT**

Multiple Models

Creating a corrected model by training and evaluating a series of models and ultimately control the bias along multiple dimensions

… to Node

… e nodes of the mode… the bias of the data - however … s approach lacks transparency

The **weaknesses of the model of milestone 2** formed part of the basis of our next steps

To build on the two milestones, we opened up **3 project streams...**

Data Bias

↓

RESEARCH

Overfitting

↓

IMPLEMENTATION

Model Comparison

↓

IMPLEMENTATION

To tackle the problem of overfitting, **three measures were implemented** to counteract

In order to avoid overfitting, the following measures were taken:

Weight Regularization

```
self.optimizer = optim.Adam(self.model.parameters(), lr=lr,
weight_decay=1e-5)
```

Dropout

```
# Dropout to overcome overfitting
self.dropout = nn.Dropout(0.25)
```

Early Stopping

```
# Add early_stopping
self.early_stopping = EarlyStopping(tolerance=5, min_delta=10)
```

The performance **did not improve significantly** however

In order to avoid overfitting, the following measures were taken:

Weight Regularization

```
self.optimizer = optim.Adam(self.model.parameters(), lr=lr,
weight_decay=1e-5)
```

Dropout

```
overcome overfitting
Dropout(0.25)
```

RESULT: **NO SIGNIFICANT IMPROVEMENT**

Early Stopping

```
# Add early_stopping
self.early_stopping = EarlyStopping(tolerance=5, min_delta=10)
```

The **weaknesses of the model of milestone 2** formed part of the basis of our next steps

To build on the two milestones, we opened up **3 project streams...**

Data Bias

RESEARCH

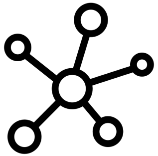
Overfitting

IMPLEMENTATION

Model Comparison

IMPLEMENTATION

An **LSTM** was chosen for a second deep learning model as comparison



Deep learning classifier #2:

Long Short-Term Memory (LSTM) Classifier

- **Tackling vanishing gradient** problem
- More control over the **short/long term storage of information** in neurons
- Require **more training data** than traditional RNNs

The **LSTM** model **performed worse** than the BoW model which might be due to the rather small data set

Deep learning classifier #2:

Long Short-Term Memory (LSTM) Classifier

LSTM led to the **following results:**

TRAINING

`F1 = 87.68%`

VALIDATION

`F1 = 51.44%`

Agenda

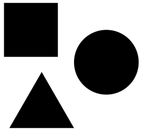
Research Objective

Implementation and Results

- Milestone 1
- Milestone 2
- Advancements and Improvements

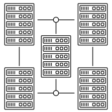
Conclusion

The experiment showed us the importance of **simplicity, enough data as well as efficient testing** cycles for achieving better results



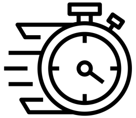
Simplicity (sometimes) wins

While in certain areas such as NLP or computer vision more complex neural models perform best, **depending on the setting simpler models can sometimes bring better performance**



Complexity needs data

While data is crucial already in every machine learning task, its **importance just increases with the complexity of a model** - especially when working with neural networks



Quick feedback matters

When testing hypothesis and new implementations, being able to get results quick matters a lot - therefore **efficient models and/or working with subsets of data** helps a lot

Thank you for your attention - **any questions?**

Q&A