Law-related text classification

Andrea Madunic

Textual analysis of urban greening projects in the press

- GrünStattGrau
- 25 magazines about urban greening
- 1456 pages in total

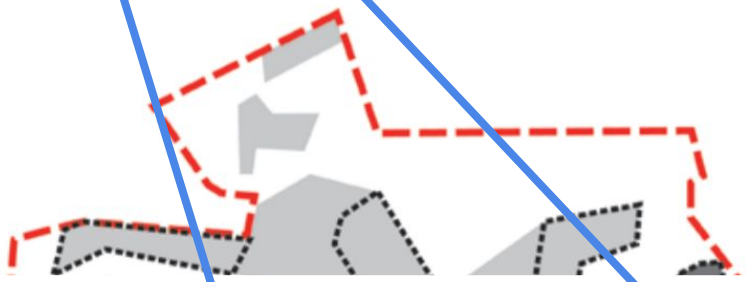
Law-related text classification using ULMFiT & BERT

- Preselecting sentences containing a set of specific legal words
- Choosing sentences randomly
- Annotating those sentences as “relevant” or “irrelevant”
- Fine-tuning a classifier using pretrained ULMFit & BERT models
- Classifying rest of the text and highlighting relevant sentences

im Bebauungsplan", Seite 20).

Die Fixierung von Dachbegrünungen in Bebauungsplänen (bzw. Grünordnungsplänen) wird bereits seit über 30 Jahren erfolgreich eingesetzt und gehört damit zu den Förderinstrumenten mit der längsten Historie. Insbesondere die Inhalte des § 9 Abs. 1 Nr. 25 des BauGB, die Mitte der 8[illegible]er-Jahre um Festsetzungsmöglichkeiten für „Teile

Falls die Begründung nicht auf Grundlage der Eingriffsregelung erfolgen kann, bedeutet dies nicht, dass die Dachbegrünung keine Berücksichtigung im Bebauungsplan findet. Aufgrund der zahlreichen Vorteile begrünter Dächer für die Stadtökologie liefert sowohl das Baugesetzbuch



Förderung

§

Bebauungsplan

Gesetz

Verlag
Deutscher Dachgärtner Verband e. V. (DDV)
Postfach 2025
72610 Nürtingen
Tel.: 07022 301378
E-Mail: contact@dachgaertnerverband.de
Internet: www.dachgaertnerverband.de

1. Auflage – Nürtingen: Deutscher Dachgärtner Verband e. V. (DDV) 2016

ISBN: 978-3-9814184-2-2

Gefördert durch die Deutsche Bundesstiftung Umwelt (DBU): Aktenzeichen 30299

Data Split

- 50 randomly chosen sentences (out of 40831 sentences)

- 42 sentences for training
- 4 sentences for validation
- 4 sentences for testing

- **Proof of concept**

Test set → [1,0,1,0]

Legally Relevant	Legally Irrelevant
Ersetzt ist eine Beeinträchtigung, wenn und sobald die beeinträchtigten Funktionen des Naturhaushalts in dem betroffenen Naturraum in gleichwertiger Weise hergestellt und das Landschaftsbild landschaftsgerecht neu gestaltet ist (§ 15 Abs.	Aufgrund seiner vielfachen positiven Effekte hat die Bundesregierung das Thema ,,Grün in der Stadt" erstmals im Jahr 2013 auf die Agenda gesetzt.
Gesetzliche Grundlagen: · Lokale Abwassersatzungen · Kommunales Abgabengesetz · Grundsatzurteile u. a. OVG Münster 2007 (NRW), VGH Kassel 2009 (Hessen), VGH Mannheim 2010 (Baden-Württemberg)	Begrünungen für unsere Stadt bietet auch zielgruppengerechte Details und die Umsetzung der

Trying out on common classification algorithms

- Rocchio
- Naive Bayes
- Support Vector Machine
- Random Forest
- Deep Learning

[1, 1, 1, 1]

- Boosting
- K Nearest Neighbors

[0, 0, 0, 0]

Trying out on common classification algorithms

- Bagging
  - [0,1,0,0]
  - [1,1,1,1]
  - [0,0,0,1]
  - …

ULMFiT

- Universal Language Model Fine-tuning
- Proposed by
  - fast.ai's **Jeremy Howard** and
  - NUI Galway Insight Center's **Sebastian Ruder**
- effective transfer learning method
- can be applied to any task in NLP

ULMFiT - Using fast.ai

```
[ ]  # Language model data
     data_lm = TextLMDataBunch.from_df(train_df = df_trn, valid_df = df_val, path = "")

     # Classifier model data
     data_clas = TextClasDataBunch.from_df(path = "", train_df = df_trn, valid_df = df_val, vocab=data_lm.train_ds.vocab, bs=32)
```

```
[ ]  #create a learner object that will directly create a model, download the pre-trained weights, and be ready for fine-tuning
     learn = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.7)
```

```
[ ]  # train the learner object with learning rate = 1e-2
     learn.fit_one_cycle(1, 1e-2)
```

ULMFiT - Using fast.ai

```
[ ]  #save encoder for classification later
     learn.save_encoder('ft_enc')

     #use the data_clas object to build a classifier with the fine-tuned encoder
     learn = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.7)
     learn.load_encoder('ft_enc')
```

```
[ ]  #fit the model again
     learn.fit_one_cycle(1, 1e-2)
```

epoch	train_loss	valid_loss	accuracy	time
0	0.742044	0.695062	0.500000	00:04

[1, 1, 1, 1]

ULMFiT - Code

- Tutorial on Text Classification (NLP) using ULMFiT and fastai Library in Python

  https://www.analyticsvidhya.com/blog/2018/11/tutorial-text-classification-ulmfit-fastai-library/

BERT

- Bidirectional Encoder Representations from Transformers
- Published by researchers at Google AI
- Pre-trained BERT-Model can be fine-tuned easily
- Trained on WikiPedia & BookCorpus
- State of the art in a wide variety of NLP tasks:
  - Question Answering
  - Sentiment Analysis
  - **Document Classification**
  - …

Result: 0.75 accuracy → [1, 0, 1, 1]

Legally Relevant	Legally Irrelevant
Ersetzt ist eine Beeinträchtigung, wenn und sobald die beeinträchtigten Funktionen des Naturhaushalts in dem betroffenen Naturraum in gleichwertiger Weise hergestellt und das Landschaftsbild landschaftsgerecht neu gestaltet ist (§ 15 Abs.	Aufgrund seiner vielfachen positiven Effekte hat die Bundesregierung das Thema ,,Grün in der Stadt" erstmals im Jahr 2013 auf die Agenda gesetzt.
Gesetzliche Grundlagen: · Lokale Abwassersatzungen · Kommunales Abgabengesetz · Grundsatzurteile u. a. OVG Münster 2007 (NRW), VGH Kassel 2009 (Hessen), VGH Mannheim 2010 (Baden-Württemberg)	Begrünungen für unsere Stadt bietet auch zielgruppengerechte Details und die Umsetzung der

Using Hugging face's BERT version

- Model architecture: BertForSequenceClassification
- Pretrained model weights: 'bert-base-multilingual-cased'
- Optimizer: BertAdam
- Dataset is converted to a 'TensorDataset'
- Sampled using 'RandomSampler'
- Batch loaded using 'DataLoader'

Pretrained model weights

- BERT-Base, Uncased
- BERT-Large, Uncased
- BERT-Base, Cased
- BERT-Large, Cased
- BERT-Base, Chinese
- BERT-Base, Multilingual (not recommended)

- BERT-Base, Multilingual Cased
  - 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters

```
model = BertForSequenceClassification.from_pretrained('bert-base-multilingual-cased',
                num_labels=num_labels)
```

pytorch_pretrained_bert

...

from pytorch_pretrained_bert.tokenization import BertTokenizer

from pytorch_pretrained_bert.modeling import BertModel, BertForSequenceClassification

from pytorch_pretrained_bert.optimization import BertAdam, WarmupLinearSchedule

from pytorch_pretrained_bert.file_utils import PYTORCH_PRETRAINED_BERT_CACHE, WEIGHTS_NAME, CONFIG_NAME

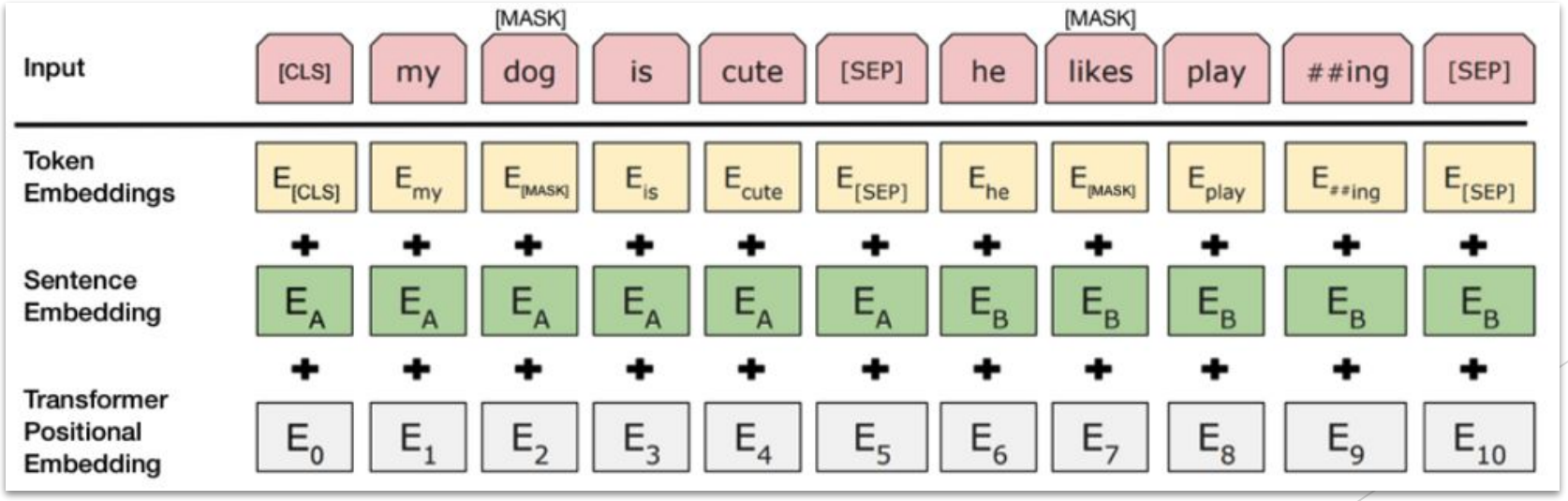
...

```
optimizer = BertAdam(optimizer_grouped_parameters,
                     lr=learning_rate,
                     warmup=warmup_proportion,
                     t_total=num_train_optimization_steps)
```

BERT - Preprocessing

Original sentence:

Aufgrund dieser Rechtsvorschriften ist von zwei Anwendungsbereichen auszugehen: der naturschutzrechtlichen Eingriffsregelung und der Eingriffsregelung in der Bauleitplanung.



BERT - Preprocessing

- Special tokens ([CLS], [SEP] are added to the sentence)
- Tokenization is done with a special BertTokenizer
- **Tokenized:**

  [CLS] Aufgrund dieser Rechts ##vor ##schriften ist von zwei Anwendung ##sbereich ##en aus ##zug ##ehen : [SEP] · der natur ##schutz ##recht ##lichen Ein ##griff ##sr ##ege ##lung und · der Ein ##griff ##sr ##ege ##lung in der Bau ##lei ##t ##plan ##ung . [SEP]

BERT - Preprocessing

- input_ids: list of numerical ids for the tokenized text
- Each vector needs to be padded to fixed length (128 or 512)
- **Tokens to Input Ids:**
  - 101 31513 11906 79037 19360 64998 10298 10166 11615 53671 68440 10136 10441 21062 35947 131 102 217 10118 84051 53057 20913 12924 12210 88508 106986 46471 25497 10130 217 10118 12210 88508 106986 46471 25497 10106 10118 18727 36777 10123 31609 10716 119 102 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

BERT - Preprocessing

- *Segment Ids* - an array with sentence index for each token
- *Mask Ids* - an array with '1' for a token and '0' for the padding
- **Mask vector:**
  - 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- **Sentence Segment Ids:**
  - 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

BERT - Preprocessing

Original sentence:

Aufgrund dieser Rechtsvorschriften ist von zwei Anwendungsbereichen auszugehen: der naturschutzrechtlichen Eingriffsregelung und der Eingriffsregelung in der Bauleitplanung.

After preprocessing:

input_ids
input_mask
segment_ids

Dataset is converted to a ‘TensorDataset’

```
all_input_ids = torch.tensor([f.input_ids for f in features], dtype=torch.long)
all_input_mask = torch.tensor([f.input_mask for f in features], dtype=torch.long)
all_segment_ids = torch.tensor([f.input_type_ids for f in features], dtype=torch.long)

all_example_index = torch.arange(all_input_ids.size(0), dtype=torch.long)
# outputs:
all_label_ids = torch.tensor([f.label_id for f in features], dtype=torch.long)
```

```
train_data = TensorDataset(all_input_ids, all_input_mask, all_segment_ids, all_label_ids)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=12)
```

BERT - Training

```
[ ]  for _ in trange(num_train_epochs, desc="Epoch"):
         tr_loss = 0
         nb_tr_examples, nb_tr_steps = 0, 0
         for step, batch in enumerate(tqdm(train_dataloader, desc="Iteration")):
             batch = tuple(t.to(device) for t in batch)
             input_ids, input_mask, segment_ids, label_ids = batch

             loss = model(input_ids, segment_ids, input_mask, labels=label_ids)

             if n_gpu > 1:
                 loss = loss.mean() # mean() to average on multi-gpu.
             if gradient_accumulation_steps > 1:
                 loss = loss / gradient_accumulation_steps

             if fp16:
                 optimizer.backward(loss)
             else:
                 loss.backward()

             tr_loss += loss.item()
             nb_tr_examples += input_ids.size(0)
             nb_tr_steps += 1
             if (step + 1) % gradient_accumulation_steps == 0:
                 optimizer.step()
                 optimizer.zero_grad()
                 global_step += 1
```

Conclusion and Next Steps

- BERT achieves best result
- But the data set is far too small for good evaluation

- Creating larger training/testing set
- Fine-tuning BERT using different parameters

- Using external data sets (legislative texts, WikiPedia articles about environment,...)

BERT - Out of the box

Bert-As-A-Service:
https://github.com/hanxiao/bert-as-service

Fast-BERT:
https://github.com/kaushaltrivedi/fast-bert

SciBERT:

https://github.com/allenai/scibert

AwesomeBERT:
https://github.com/Jiakui/awesome-bert

A guide to simple text classification with bert:

https://mc.ai/a-guide-to-simple-text-classification-with-bert/

Thank you for your attention!