



Data Mining & NLP in e-rara

Projektarbeit CAS Practical Machine Learning 2021

Kathi Woitas, M.A.
2021-04-09

Inhaltsverzeichnis

1	Introduction	3
1.1	E-rara & Bernensia	3
1.2	Scope of work	6
1.3	Approach and Technology	6
2	Named Entity Recognition (NER)	6
2.1	Named entity recognition (NER)	7
2.2	NER machine learning implementation	7
2.3	Flair Framework	9
2.3.1	Available Word Embeddings	9
2.3.2	Contextual String Embeddings (Flair Embeddings)	10
2.3.3	Byte Pair Embeddings	10
3	Used Models	11
3.1	spaCy de_core_news_lg	11
3.2	Flair ner-multi-fast	11
3.3	BPEmb dbmdz-historic-ner-onb	11
4	Implementation	12
4.1	Data access and understanding	12
4.2	Data Preprocessing	12
4.3	Applying ML Models	13
4.3.1	spaCy de_core_news_lg	13
4.3.2	Flair ner-multi-fast	13
4.3.3	BPEmb dbmdz-historic-ner-onb	14
5	Evaluation	14
5.1	Sample	14
5.2	Statistical View	16
5.3	Individual Assessment	16
6	Outcome	17
7	Figures	18
8	Tables	18
9	Bibliography	18

1 Introduction

1.1 E-rara & Bernensia

Historical texts are of wide interest for scholarship, not only for history, but also for several social sciences, like political science and economics, for geography, linguistics and various humanities. Therefore, libraries and archives not only digitize old stock at large, but also deal actively with techniques around digitization and its downstream tasks. Newspaper digitization, for instance, is a major and demanding topic, with its vast quantities and the demanding layouts - along the general issues of old content mentioned before. Therefore, leading libraries more and more offer own solutions to these problems, using elaborate technologies of machine learning and deep learning.

E-rara¹ is a web platform holding digitized Swiss printed works supplied by Swiss institutions, mainly academic libraries. It contains books, book series, leaflets/pamphlets, graphic works, music prints and maps from the 15th to the 20th century (due to copyright reasons). Of approx. 84'600 items 19'800 have fulltext files available.

The University Library Bern holds three special collections on e-rara : « Rossica Europena », « Russian Exile Publications » and « **19th until early 20th century Bernensia** ». The term Bernensia means works, that deal with the history, culture, people, language and geography of the canton Bern. The collection comprises 519 items, of which 270 have a fulltext file. 176 of these are books are in German language².

E-rara's features of data access are sparse. Item metadata is delivered through an OAI-PMH 2.0³ interface in MODS⁴ or METS⁵ format and the OAI-minimal standard Dublin Core⁶. Images of the digitized works are delivered through IIIF⁷ interfaces, as are (very few) metadata fields. These interfaces lack documentation, and information about them has to be gathered manually by exploration. Furthermore, **fulltext files** in TXT format are only available on the dedicated item pages, as shown in Figure 1. There is no batch download or interface for fulltext access.

¹ E-rara: <https://www.e-rara.ch/?lang=en>

² Status of 2021-04-03, see <https://www.e-rara.ch/nav/classification/1395750?&facets=ftmode%3D%22ocr%22%20and%20language%3D%22ger%22%20and%20type%3D%22book%22>

³ The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) 2.0: <http://www.openarchives.org/OAI/openarchivesprotocol.html>

⁴ Metadata Object Description Schema (MODS): <http://www.loc.gov/standards/mods/>

⁵ Metadata Encoding & Transmission Standard (METS): <https://www.loc.gov/standards/mets/>

⁶ Dublin Core Metadata Elements Set (DCMES) v.1.1: <http://purl.org/dc/elements/1.1/>

⁷ International Image Interoperability Framework (IIIF): <https://iiif.io/>

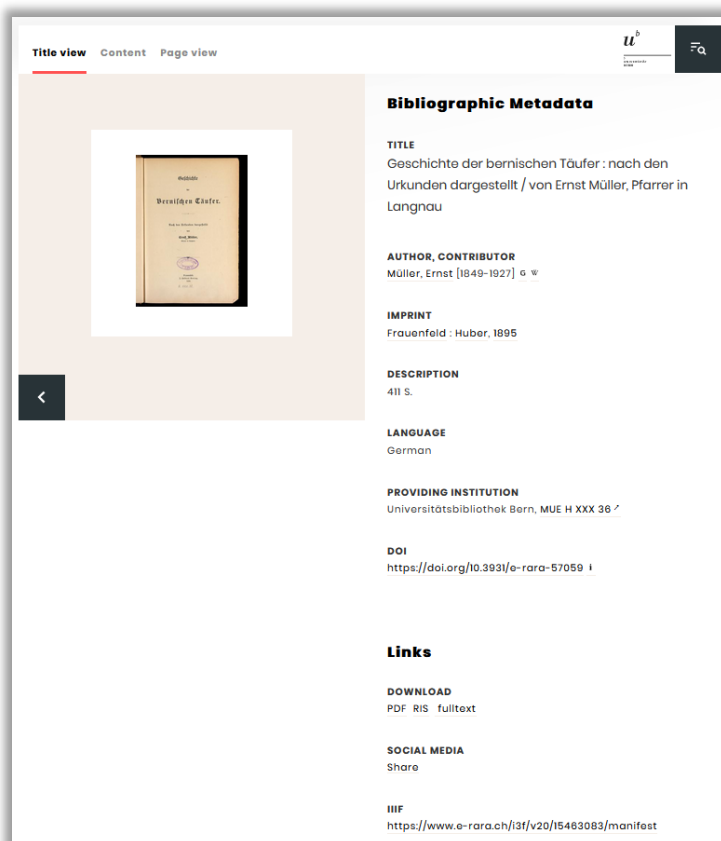


Figure 1: Item view e-rara with download links (PDF, RIS, fulltext)

These older, but rare Bernensia are only marginally indexed with subjects (and according different standards), but keep **precious information about a large number of Bernese localities and geographic sites**. Therefore, NER on LOC entites might give a deeper and more comprehensive insight into these documents and could be used to build better information access around them.

As a major obstacle, the fulltexts of older printed works yet suffer from **suboptimal OCR results**. Publications of the 19th and early 20th century are often set in fonts different from today's, and furthermore, such regional topics often bear list and table contents and illustrated pages. There is no detailed information about OCR quality in e-rara or Bernensia collection, like error rates, available.

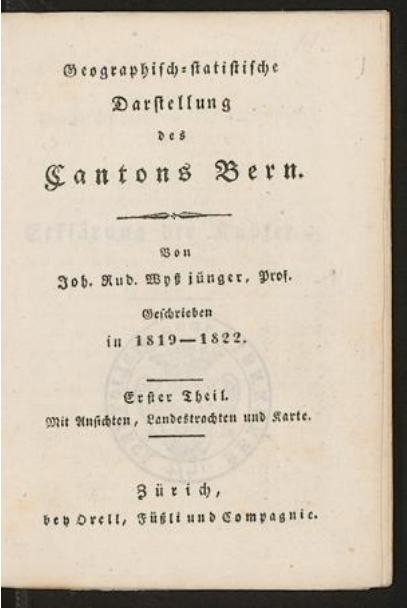

	<p>Geographisch-statistische Darstellung des Cantons Bern / von Joh. Rud. Wyss jünger, Prof. geschrieben in 1819-1822. 1825. https://doi.org/10.3931/e-rara-18002. p. 0</p> <p>Fulltext : Geographisch-statistische Darstellung d c s C a t t t o n s Bern</p> <p>Das NIL</p> <p>Geographisch-statistische Darstellung »- s Kantons Bern. V 0 » Joh. Sind. Wyß jünger, Pros. Geschrieben in 1819 - 1822. Erster Theil. Mit Ansichten, Landestrachten nnd Karte. Zürich, beyOrc11, Füst li » » d § o m p a g » i</p>
	<p>Heinrich Tuerler: Das alte Biel und seine Umgebung: [Tafeln u. Abb. im Text]. Biel 1902. https://doi.org/10.3931/e-rara-28664. p. 1</p> <p>Fulltext:</p> <p>.jHV. ■. ■'v f, ; r / ? r.fi S, J. i < A > r* * * ■ " - '4' J 1 ' V : vü ; ; , i ■. ■. ' / ? \ ^V ; i ' ■. ■. ; ' E v r si ; ' I A'v AA.LMM ' ' ' ' / .. " • ' • ; > r - Ä.A jSV* " ! ■" 1 ; Ä ' ' ' - . A ». •AAv'- äivflÄ ; - ! ; V A 'j ■ y, ' iAf'SVA ' ° * ' ' , ~>' %r 5 ; '7V a... ' 1 .. i fifTU ,...- '</p> <p>Ansicht von viel von Büöen. Flach einer Zeichnung aus der Kieler Lkronik des Verrsius, 162b. Das alle viel und seine Umgebung. Erläuternder Text IMLi von Dr. B. Türlar, Staatsarchivar in Bern. Einleitung. ^)as alte Biel und seine Umgebung" soll eine Heimatkunde in Bildern für dos gange Seelanö sein. Es ist gewidmet seinen Bewohnern und seinen freunden, jedem, dessen Wiege in einer der alt- ehrwürdigen Ortschaften stand, jedem, dem das Land zur neuen Heimat geworden ist, oder der es um seiner Vorzüge willen lieb gewonnen hat. Heue Zeiten haben neue forderungen und Ruffgaben gebracht, und das Rite stürzt in Ruinen. Die Stäöte haben beinahe durchweg die engen segeln gesprengt, die ihnen die Ringmauern umgelegt hatten. Über die ehemaligen Stadtgräben * hinaus dehnen sich die (Zassen aus. Rber auch in den Dörfen fordert der fortschritt gar oft den Ruin des Riten, das seine Verteidiger verloren hat. Es ist hohe Zeit, die Denkmäler aus den Zeiten der Voreltern noch im Bilde durch den Stift des Künstlers festzuhalten, die alten Bilder zu sammeln und den Enkeln zu überliefern. Der größte Raum in diesem Werke, das leicht auf die doppelte oder dreifache OröHe hätte gebracht werden können, kommt der Stadt Biel zu wegen ihrer gröttern Rus- dehnung und Bedeutung. Wenn dieses Verhältnis auch im Texte besteht, so geschieht es deswegen, weil Biel ein reichhaltiges Rrchiv besitzt und der Verfasser dasselbe besser als andere kennt. .A. /U : [■'> (!</p>

Figure 2: Examples of digitized imgaes and their fulltext

1.2 Scope of work

Scope of this work is a – as far as known - first-time utilization of the Bernensia fulltexts with NLP methods. More precisely, I will explore different methods for NER on localities in Bernensia items, and judge the outcomes for a future reuse of this approach.

1.3 Approach and Technology

The proposed approach comprises four steps:

1. Data access and understanding:
Bernensia items on e-rara are to be identified and their metadata and fulltexts are to be retrieved. Metadata is to be read out.
2. Data Preprocessing:
Fulltexts are to be preprocessed.
3. Applying ML models:
Several NER methods on LOC entities are to be applied.
4. Evaluation:
A suitable evaluation method is to be constructed, as there is no gold standard data for Bernensia LOC entities.
5. Outcome:
A suggestion is to be made about the usefulness of a further NER exploration of Bernensia items.

Technology:

Elaborate NLP methods need a certain amount of RAM (and disk space). To cope with this, the cloud computing facility of Google Colab⁸, and the High Performance Cluster (HPC) of the University of Bern, UBELIX⁹, is used (along own machine).

2 Named Entity Recognition (NER)

Named entity recognition (NER) is a certain downstream task in **Natural Language Processing (NLP)**. NLP comprises a whole plethora of methods and tasks around text (and transcribed speech). To get an impression, what use cases are today fulfilled with NLP methods, the state-of-the-art directory Papers with Code¹⁰ distinguishes 405 different tasks, with famous main classes of tasks like

- Text Classification
- Sentiment Analysis
- Text Generation
- Text Summarization
- Question Answering
- Machine Translation.

NLP methods broadly can be divided into two main classes :

1. Linguistic approaches
2. Statistical approaches.

Linguistic approaches, which have origins in computer linguistics decades ago, create generally applicable, rule-based language models. In use, text is analysed through full syntactic parsing, i.e. its underlying linguistic structure is reconstructed and used for further analysis.

The main derived linguistic features are

- Morphological: part-of-speech (POS) tags with word classes like verb, proper noun, adjective
- Syntactic: nested phrase tags like nominal phrases, verb phrases, prepositional phrases
- Semantic: deployed using the syntactic dependencies and external vocabularies like thesauri or ontologies.

⁸ Google Colab: <https://colab.research.google.com/>

⁹ UBELIX: <http://www.id.unibe.ch/hpc>

¹⁰ Papers with Code Natural language Processing, <https://paperswithcode.com/area/natural-language-processing>

Statistical approaches create probabilistic (or generative) language models, looking at text as aggregation of recurring patterns with a certain distribution. To derive these patterns, machine learning from a vast number of documents is performed. In use, the occurrence of these patterns trigger the (implicit) assigned and most likely rule or action.

The differentiation between this to classes of NLP methods is important basic knowledge, but in practice to a certain point outdated, as usually the two approaches are combined. For instance, in a machine-learning-driven analysis probabilistic features are used as well as defined rules and linguistic treebank information.

2.1 Named entity recognition (NER)

Named entity recognition (NER) as a certain NLP downstream task, that identifies terms and phrases that refer to specific types of entities, whereby **entities** are certain objects of interest in the text, and to specific types of **relations**, which are certain connections between those entities.

There are four basic classes of entities to be recognized, but there are also a growing number of NER taggers that differentiate more classes.

- Person (PER)
- Locality (LOC)
- Organisation (ORG)
- Miscellaneous (MISC).

Entities often refer to real-world entities like individual persons (e.g. *Ada Lovelace*), corporations (e.g. *Alphabet*), places (e.g. *Matterhorn*), but can also be imaginary ones or concepts (e.g. *Gregor Samsa*, *Enterprise*) and generic terms (e.g. *market place*, *mother*). Named entity disambiguation (NED), i.e. entity individualisation and named entity linking (NEL), i.e. referencing the entity to certain name spaces, are tasks, that build upon NER.

NER bears serious challenges. As seen, the potential number of entities is great, but more important, terms and phrases representing entities are often polysemous (e.g. *May* – time, part of date, (part of) person, several localities), and/or sometimes nested (e.g. *AXA Winterthur*).

Practically, NER can be seen as a sequence labeling task. NER in use tags tokens of the text along a certain schema, for instance:

- IOB/IOB2, meaning *inside...*, *outside...*, *beginning...* of an entity,
 - BIOES meaning *beginning...*, *inside...*, *outside...*, *ending...* of an entity and *single* entity,
- often in combination assigned entity class:
- PER, LOC, ORG, MISC.

NER can be accomplished via different basic approaches:

1. A dictionary/list of entities can be used for (fuzzy) string matching (e.g. gazetteers).
2. Rule-based: A set of rules will be defined to identify string parameters, which bear entities (e.g. word with a first capital letter inside a sentence in English), maybe combined with statistical models (Mikheev, Moens and Grover, 1999)
3. Multiclassification task in supervised machine learning, using various features derived from the token(s) in question.

Of course, these approaches can and will be combined in practice.

2.2 NER machine learning implementation

It is obvious that dictionary/list-based approaches to NER are limited to given entities and therefore they are mostly feasible for smaller domains. Also, they may suffer from ambiguity.

Rule-based approaches require a lot of intellectual work, and are limited to the language and text type the rules were defined for.

For historic texts, NER with these methods is largely limited as they are

- generally (large) collections of different scopes, topics and text types
- in historic language, bearing different name and orthographic variants
- in old fonts leading to OCR errors
- sometimes bi- or multilingual.

Machine learning (ML) approaches for NER on the other hand basically offer methods to deal with these issues :

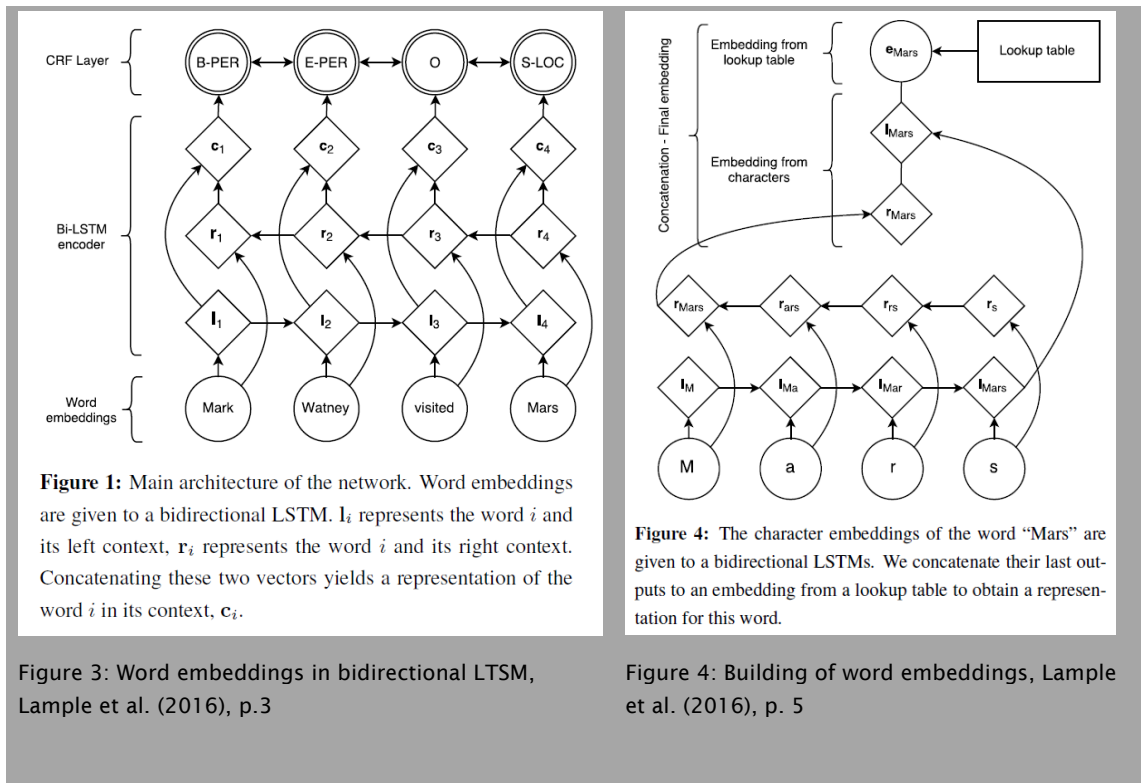
- NER classifiers can be trained on specific collections to cope with their peculiar language aspects
- Particular NER classifiers subduct from the token/word by itself and are therefore more robust against historic spelling variants and/or OCR errors.

Historically, basic ML approaches for NER are Maximum Entropy Classifier, Support Vector Machines (SVM), and Conditional Random Fields (CRF) (Krishnan and Ganapathy, 2005), more currently approaches rely on deep learning methods like neural nets (Lample *et al.*, 2016).

Not surprisingly, features used by NER taggers changed with the available technology. Classic features used by basic ML approaches are

- word level features, describing a token with case, punctuation, numerical value and special characters (Nadeau and Sekine, 2007, pp. 12-13)
- list lookup features, making use of general word lists and entity lists (Nadeau and Sekine, 2007, pp. 13-15)
- document and corpus features, like multiple occurrences or entity coreferences (Nadeau and Sekine, 2007, pp. 15-17).

Lample et al. (2016) introduced bi-directional **Long Short-term Memory Network** (LSTM) in NER, which eliminates the need of costly language-specific resources and classic features: For each word, a LSTM computes in each sentence the sequence before (forward LSTM) and, in reverse order, after (backward LSTM) the token. The final representation of a word is obtained by concatenating its left and right context representations, i.e. a proper representation of a word in its context. For the final tagging decision of the LSTM outputs a CRF Layer is implemented. Inputs to the LSTM are of course not the tokens per se, but a representation of them making use of the vector space model of text representation (Salton, Wong and Yang, 1975). Here, the word vectors are build at core with a bidirectional LSTM too, on the basis of their characters (*character embeddings*). This character-level representation is then concatenated with a word-level representation from a word lookup-table, building the final **word embedding as input** to the LSTM-CRF for NER.



Current approaches largely rely on neural networks and use different kind of word embeddings as inputs, roughly speaking features deriving from the surroundings of the token in question.

2.3 Flair Framework

Flair (Akbik *et al.*, 2019) is a state-of-the-art framework for NLP in the PyTorch ecosystem. It supports many tasks in text analytics, like sequence labeling, text classification, similarity learning and text regression. It delivers a convenient wrapper around several types of word embedding approaches and moreover, a method for combining these (*stacked embeddings*). Furthermore it includes classes for data and model loading, training, evaluation. Flair features also its signature *Flair embeddings*.

2.3.1 Available Word Embeddings

Flair makes different classes of popular and experimental embeddings easily accessible, an overview is provided in Akbik et al. (2019, pp. 55–56).

Class	Type	Pretrained?
WordEmbeddings	classic word embeddings (Pennington et al., 2014)	yes
CharacterEmbeddings	character features (Lample et al., 2016)	no
BytePairEmbeddings	byte-pair embeddings (Heinzerling and Strube, 2018)	yes
FlairEmbeddings	character-level LM embeddings (Akbik et al., 2018)	yes
PooledFlairEmbeddings	pooled version of FLAIR embeddings (Akbik et al., 2019b)	yes
ELMoEmbeddings	word-level LM embeddings (Peters et al., 2018a)	yes
ELMoTransformerEmbeddings	word-level transformer LM embeddings (Peters et al., 2018b)	yes
BertEmbeddings	byte-pair masked LM embeddings (Devlin et al., 2018)	yes
DocumentPoolEmbeddings	document embeddings from pooled word embeddings (Joulin et al., 2017)	yes
DocumentLSTMEEmbeddings	document embeddings from LSTM over word embeddings	no

Table 1: Summary of word and document embeddings currently supported by FLAIR. Note that some embedding types are not pre-trained; these embeddings are automatically trained or fine-tuned when training a model for a downstream task.

Figure 5: Word embedding classes in Flair

Also, some famous NLP datasets are available via load method, e.g. the CoNLL-03 (Tjong Kim Sang and De Meulder, 2003) dataset, which is widely used as NER benchmark for German language texts. Loaded NLP datasets can easily transformed into a ready-to-use Corpus object with train-dev-eval data split. Flair also offers a selection of pre-trained models for different NLP tasks, among them a standard 4-class NER tagger for German, English, Dutch and Spanish trained on CoNLL-03.

Since Flair is gaining awareness, the distribution of third party pre-trained models via Flair is flourishing. An overview of available models is provided on Flair's Github account (*flairNLP/flair*, 2021)¹¹ and via the HuggingFace directory using the «Flair»-filter¹².

2.3.2 Contextual String Embeddings (Flair Embeddings)

Contextual string embeddings as introduced by Akbik/Blythe/Vollgraf (2018) are a promising new kind of word embeddings. They are trained without any explicit notion of words, i.e. words are modeled solely on basis of the symbol sequence of their surrounding. They are *dynamic word embeddings*, meaning that the same word will have different embeddings depending on its contextual use. The embedding is concatenated from forward and backward hidden state outputs in sentential context (Akbik, Blythe and Vollgraf, 2018, pp. 1640–1642).

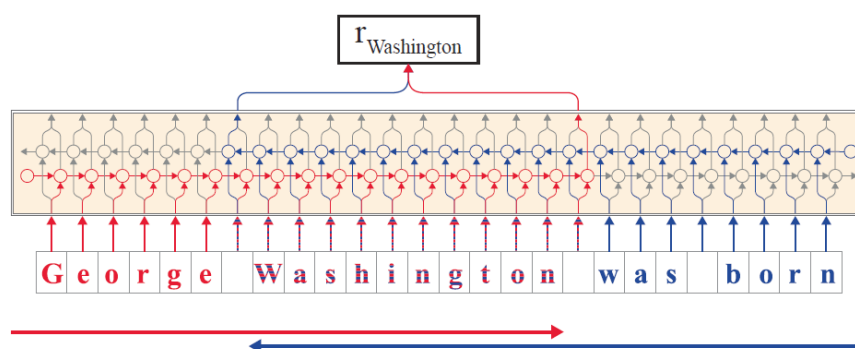


Figure 2: Extraction of a contextual string embedding for a word (“Washington”) in a sentential context. From the forward language model (shown in red), we extract the output hidden state after the last character in the word. This hidden state thus contains information propagated from the beginning of the sentence up to this point. From the backward language model (shown in blue), we extract the output hidden state before the first character in the word. It thus contains information propagated from the end of the sentence to this point. Both output hidden states are concatenated to form the final embedding.

Figure 6: Building of contextual string embeddings, Akbik/Blythe/Vollgraf 2018, p. 1641

2.3.3 Byte Pair Embeddings

Byte-Pair embeddings as introduced by Sennrich/Haddow/Birch (2016) and broadly implemented by Heinzlerling/Strube (2018) are subword-level embeddings, which views text as a sequence of symbols and iteratively merges the most frequent symbol pair into a new symbol. The number of merge operations thus determines the length of the resulting character sequences.

¹¹ flairNLP/flair Tutorial 2: Tagging your Text, List of Pre-Trained Sequence Tagger Models:

https://github.com/flairNLP/flair/blob/master/resources/docs/TUTORIAL_2_TAGGING.md#list-of-pre-trained-sequence-tagger-models

¹² HuggingFace Flair model hub: <https://huggingface.co/models?filter=flair>

Merge ops	Byte-pair encoded text
5000	豊 田 駅 (と よ だ え き) は 、 東 京 都 日 野 市 豊 田 四 丁 目 に あ る
10000	豊 田 駅 (と よ だ え き) は 、 東 京 都 日 野 市 豊 田 四 丁 目 に あ る
25000	豊 田 駅 (と よ だ え き) は 、 東 京 都 日 野 市 豊 田 四 丁 目 に あ る
50000	豊 田 駅 (と よ だ え き) は 、 東 京 都 日 野 市 豊 田 四 丁 目 に あ る
Tokenized	豊 田 駅 (と よ だ え き) は 、 東 京 都 日 野 市 豊 田 四 丁 目 に あ る
10000	豊 田 站 是 東 日 本 旅 客 鐵 道 (JR 東 日 本) 中 央 本 線 的 鐵 路 車 站
25000	豊 田 站 是 東 日 本 旅 客 鐵 道 (JR 東 日 本) 中 央 本 線 的 鐵 路 車 站
50000	豊 田 站 是 東 日 本 旅 客 鐵 道 (JR 東 日 本) 中 央 本 線 的 鐵 路 車 站
Tokenized	豊 田 站 是 東 日 本 旅 客 鐵 道 (JR 東 日 本) 中 央 本 線 的 鐵 路 車 站
1000	to y od a _station is _a _r ail way _station _on _the _ch ū ō _main _l ine
3000	to y od a _station _is _a _railway _station _on _the _ch ū ō _main _line
10000	toy oda _station _is _a _railway _station _on _the _ch ū ō _main _line
50000	toy oda _station _is _a _railway _station _on _the _ch ū ō _main _line
100000	toy oda _station _is _a _railway _station _on _the _ch ū ō _main _line
Tokenized	toyoda station is a railway station on the chūō main line

Figure 7: Byte-Pair Encoding underlying BytePair embeddings, Heinzerling et al. (2018), p. 2989

3 Used Models

3.1 spaCy de_core_news_lg

Spacy¹³ is a performative and extensive standard NLP framework, covering presumably all tasks on linguistic and statistical text processing. It delivers pretrained processing pipelines in many languages and offers libraries for NLP machine learning. For German, spaCy provides three standard models (also called *pipelines*)¹⁴, differing in their size. The pipelines fulfill e.g. tokenization, sentence splitting, POS tagging, syntactic/dependency parsing, lemmatizing and vectorizing, and also NER tagging for the four main classes. Lately, spaCy also offers a transformer-based model for German (bert-base-german-cased), which yet lacks NER tagging.

For my exploration I chose the largest available spaCy model de_core_news_lg¹⁵. The model was trained on modern texts (TIGER¹⁶ and WikiNER¹⁷ corpus) from a German newspaper and Wikipedia articles. It reveals a F1-score: 85.12 in NER tagging.

3.2 Flair ner-multi-fast

*Flair ner-multi-fast*¹⁸ (Akbik, Bergmann and Vollgraf, 2019) is the genuine Flair multilingual 4-class-NER tagger. It is trained on CoNLL-03 for four languages (English, German, Dutch and Spanish). It implements the genuine Flair embeddings in a LSTM-CRF. It's *fast* flavour is aligned to CPU use instead of GPU. It reveals a performance of F1-score 85,72 on the CoNLL-03 German revised test set.

3.3 BPEmb dbmdz-historic-ner-onb

The Byte-pair embedding based model dbmdz-historic-ner-onb¹⁹ is build by the Bavarian State Library, and a 4-class-NER tagger trained on Austrian historic newspapers (1710-1873) - which features historic German language of the 19th century as well as OCR errors. It is optimized for GPU-use and reveals a F1-score of 85.69.

¹³ spaCy v3.0, <https://spacy.io/>

¹⁴ spaCy, German models: <https://spacy.io/models/de>

¹⁵ spaCy de_core_news_lg, https://github.com/explosion/spacy-models/releases/tag/de_core_news_lg-3.0.0

¹⁶ TIGER Corpus, <https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger/>

¹⁷ WikiNER Corpus, https://figshare.com/articles/dataset/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462500

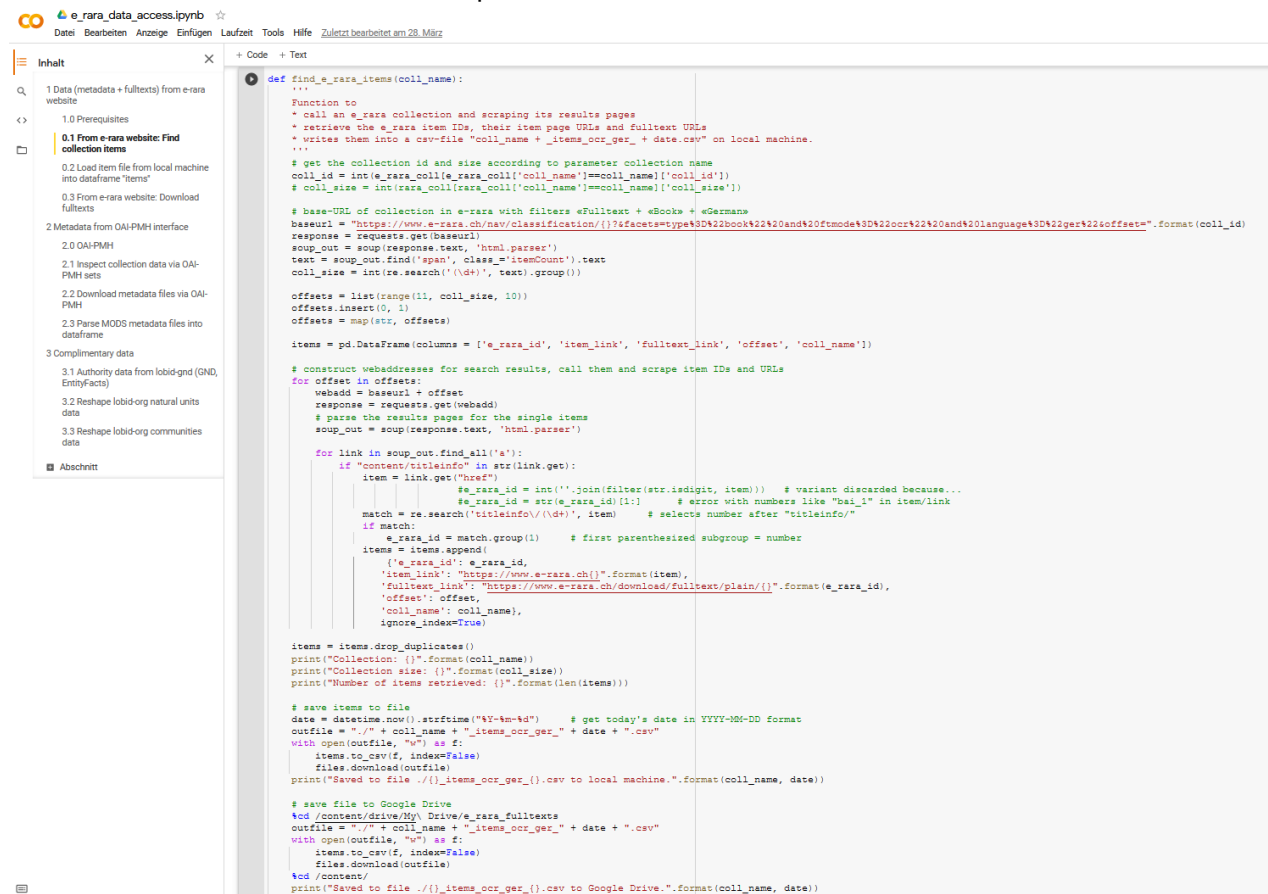
¹⁸ HuggingFace, Flair ner-multi-fast: <https://huggingface.co/flair/ner-multi-fast>

¹⁹ HuggingFace, dbmdz-historic-ner-onb: <https://huggingface.co/dbmdz/flair-historic-ner-onb>

4 Implementation

4.1 Data access and understanding

To retrieve Bernensia metadata and fulltext files a comprehensive Python script resp. Jupyter Notebook was created to question e-rara OAI-PMH interface for metadata and e-rara website for fulltext files. Both, metadata files in XML (MODS) format, and fulltexts in TXT format, were downloaded as separate files. After that, reading out XML files for metadata fields, and integrating fulltexts into a dataframes was accomplished.



```
def find_e_rara_items(coll_name):  
    """  
    Function to  
    * call an e_rara collection and scraping its results pages  
    * retrieve the e_rara item IDs, their item page URLs and fulltext URLs  
    * writes them into a csv-file "coll_name + _items_ocr_ger_ + date.csv" on local machine.  
    """  
    # get the collection id and size according to parameter collection name  
    coll_id = int(e_rara_coll[e_rara_coll['coll_name']==coll_name]['coll_id'])  
    # coll_size = int(e_rara_coll[e_rara_coll['coll_name']==coll_name]['coll_size'])  
    # base-URL of collection in e-rara with filters «Fulltext» + «Book» + «German»  
    baseurl = "https://www.e-rara.ch/nav/classification/176facets=type%3D%22Book%22%20and%20format%3D%22ocr%22%20and%20language%3D%22ger%22offset=" + coll_id  
    response = requests.get(baseurl)  
    soup_out = soup(response.text, 'html.parser')  
    text = soup_out.find('span', class_='itemCount').text  
    coll_size = int(re.search('(\d+)', text).group())  
    offsets = list(range(11, coll_size, 10))  
    offsets.insert(0, 1)  
    offsets = map(str, offsets)  
    items = pd.DataFrame(columns = ['e_rara_id', 'item_link', 'fulltext_link', 'offset', 'coll_name'])  
    # construct webaddresses for search results, call them and scrape item IDs and URLs  
    for offset in offsets:  
        webadd = baseurl + offset  
        response = requests.get(webadd)  
        # parse the results pages for the single items  
        soup_out = soup(response.text, 'html.parser')  
        for link in soup_out.find_all('a'):  
            if "content/titleinfo" in str(link.get()):  
                item = link.get("href")  
                # e_rara_id = int(''.join(filter(str.isdigit, item))) # variant discarded because...  
                # e_rara_id = str(e_rara_id)[1:] # error with numbers like "bai_1" in item/link  
                match = re.search("titleinfo/(.*)", item) # selects number after "titleinfo/"  
                if match:  
                    e_rara_id = match.group(1) # first parenthesized subgroup = number  
                    items = items.append(  
                        {'e_rara_id': e_rara_id,  
                         'item_link': "https://www.e-rara.ch/{}".format(item),  
                         'fulltext_link': "https://www.e-rara.ch/download/fulltext/plain/{}".format(e_rara_id),  
                         'offset': offset,  
                         'coll_name': coll_name},  
                        ignore_index=True)  
    items = items.drop_duplicates()  
    print("Collection: {}".format(coll_name))  
    print("Collection size: {}".format(coll_size))  
    print("Number of items retrieved: {}".format(len(items)))  
    # save items to file  
    date = datetime.now().strftime("%Y-%m-%d") # get today's date in YYYY-MM-DD format  
    outfile = "/" + coll_name + "_items_ocr_ger_" + date + ".csv"  
    with open(outfile, "w") as f:  
        items.to_csv(f, index=False)  
        files.download(outfile)  
    print("Saved to file ./{}_items_ocr_ger_{}.csv to local machine.".format(coll_name, date))  
    # save file to Google Drive  
    !cd /content/drive/My\ Drive/e_rara_fulltexts  
    outfile = "/" + coll_name + "_items_ocr_ger_" + date + ".csv"  
    with open(outfile, "w") as f:  
        items.to_csv(f, index=False)  
        files.download(outfile)  
    !cd /content/  
    print("Saved to file ./{}_items_ocr_ger_{}.csv to Google Drive.".format(coll_name, date))
```

Figure 8: Example function for finding items on e-rara website according to their collection and save a CSV-file of items with their fulltext links to local machine and Google Drive.

4.2 Data Preprocessing

As mentioned beforehand, the OCR quality of Bernensia items is suboptimal. There are OCR errors in transcribing words correctly, and there is OCR noise, which stems from illustrations etc. To get rid of the most of obvious OCR noise, but keep the text with sentence ends and almost all words and sentences, only a light-weight normalisation of the fulltexts took place.

```

# Preprocess the raw text (remove punctuation and drop out words with length < 3)
import nltk
nltk.download('punkt') # "Punkt" = standard classifier for sentence segmentation
from nltk import word_tokenize

print("Successfully imported necessary modules")

def remove_punctuation(wordlist):
    punctuation = [',', ';', ':', '(', ')', '[', ']', '{', '}', '\\'', '\\'', '\\'', '\\'', '\\'', '\\'', '\\-', '«', '»', '£', '^', '~', '!', '@', '•', '■', '♦', '§']
    wordlist_stripped = [w for w in wordlist if w not in punctuation]
    wordlist_stripped = [w for w in wordlist_stripped if len(w) > 2]
    return wordlist_stripped

def preprocess_nltk(text):
    wordlist = nltk.word_tokenize(text, language='german')
    wordlist = remove_punctuation(wordlist)
    wordlist = ' '.join(wordlist)
    return wordlist

```

Figure 9: Script for shallow text preprocessing.

4.3 Applying ML Models

Due to different constraints in necessary RAM, NER models were fulfilled on different machines.

4.3.1 spaCy de_core_news_lg

The spaCy German large model was applied directly in a Google Colab Notebook on basis of corpus dataframe, which holds the clean texts. Sentence tokenizing was necessary because of partly large text files.

```

[ ] # Applying large SpaCy german model on clean_text and write LOC entities per clean text
from nltk import sent_tokenize

corpus['spacy'] = ''
for index in corpus.index:
    loc_ents = []
    sentences = nltk.sent_tokenize(corpus['clean_text'][index], language='german')
    for s in sentences:
        doc = nlp(s)
        loc_ents.append([ent.text for ent in doc.ents if ent.label_ == 'LOC'])
    corpus['spacy'][index] = loc_ents

```

Figure 10: Application of the spaCy German large model

4.3.2 Flair ner-multi-fast

The genuine Flair ner-multi-fast model was applied partly in a Google Colab Notebook, and partly on the high performance cluster (HPC) of the University of Bern, UBELIX²⁰. The model was not capable to compute all fulltexts correctly. There were several items which led to a break-off of the job, and trying to compensate this failure with more RAM power was not successful. It was also obvious that text length was not the critical parameter for this behaviour. The model also needs a sentence splitting before NER tagging.

²⁰ UBELIX, <https://ubelix.unibe.ch/>

```
[ ] from flair.data import Sentence
    from flair.models import SequenceTagger

# load the NER tagger
# See available sentence tagger: https://github.com/flairNLP/flair/blob/master/resources/docs/TUTORIAL_2_TAGGING.md#list-of-pre-trained-sequence-tagger-models
tagger = SequenceTagger.load('flair/ner-multi-fast') # size: 1.51 GB

2021-04-06 02:39:26,892 loading file /root/.flair/models/ner-multi-fast/d0ca1daace2b097b04a886b4be80d82634229555eb2da7079b1b102579fd3835.7b305379b36567738bc455e399f4e4b3

[ ] #Import segtok library to split the paragraph into sentences
    from segtok.segmenter import split_single

corpus['ner-multi-fast-I'] = ''
for index in corpus.index[155:176]:
    # use splitter to split text into list of sentences
    sentences = [Sentence(sent, use_tokenizer=True) for sent in split_single(corpus['clean_text'][index])]
    tagger.predict(sentences) # predict tags for sentences
    loc_ents = []
    for s in sentences:
        for token in s.tokens:
            tag = token.get_tag('ner')
            if tag.value in ['S-LOC', 'B-LOC', 'E-LOC', 'I-LOC']:
                loc_ents.append([token.text, tag.value])
            corpus['ner-multi-fast-I'][index] = loc_ents

# Write to Google Drive
!cd /content/drive/My\ Drive/e_rara_fulltexts/bernensia
outfile = "/corpus_bernensia_ger_LOC_ner-multi-fast-155.csv"
with open(outfile, "w") as f:
    corpus.to_csv(f, index=False, columns=['e_rara_id', 'clean_text_length', 'ner-multi-fast-I'])
!cd /content/
print("Saved to file {} to Google Drive.".format(index))
```

Figure 11: Application of the Flair model with data subsetting due to job break-offs

4.3.3 BPEmb dbmdz-historic-ner-onb

Applying the Byte-pair model dbmdz-historic-ner-onb took massive more RAM and was not executable in Google Colab, and was performed as Python script on UBELIX, too. Up to 24 nodes with 40-50 GB per CPU were used to compensate the GPU-optimization of the model.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# install and import modules
from flair.data import Sentence
from flair.models import SequenceTagger
from segtok.segmenter import split_single
import pandas as pd

# load the NER tagger
# See available sentence tagger: https://github.com/flairNLP/flair/blob/master/resources/docs/TUTORIAL_2_TAGGING.md
tagger = SequenceTagger.load('dbmdz/flair-historic-ner-onb') # size: 444MB GPU

# load corpus file
infile = "corpus_bernensia_ger_LOC_SpaCy.csv"
with open(infile, 'r') as f:
    corpus = pd.read_csv(f, encoding="UTF-8", usecols=[1,3])

corpus['historic-ner-onb'] = ''
for index in corpus.index:
    # use splitter to split text into list of sentences
    sentences = [Sentence(sent, use_tokenizer=True) for sent in split_single(corpus['clean_text'][index])]
    tagger.predict(sentences) # predict tags for sentences
    loc_ents = []
    for s in sentences:
        for token in s.tokens:
            tag = token.get_tag('ner')
            if tag.value in ['S-LOC', 'B-LOC', 'E-LOC', 'I-LOC']:
                loc_ents.append([token.text, tag.value])
            corpus['historic-ner-onb'][index] = loc_ents

outfile = "corpus_bernensia_ger_LOC_dbmdz-historic-ner-onb-I.csv"
with open(outfile, "w") as f:
    corpus.to_csv(f, index=False, columns=['e_rara_id', 'historic-ner-onb'])
```

Figure 12: Application of the BPEmb model

5 Evaluation

5.1 Sample

Only items with successful NER tagging of all three approaches were used for evaluation. To avoid any individual preferences and to get a realistic view of the given and produced data, the first 52 items (minus the two which failed in the Flair approach) were selected for evaluation. Since data was derived

from the e-rara website in an alphabetically order, a good random sample from the population of 176 can be assumed.

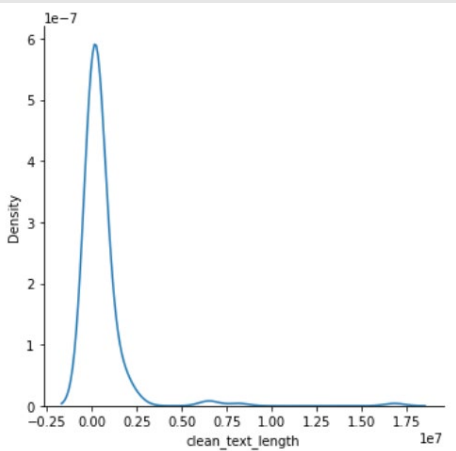
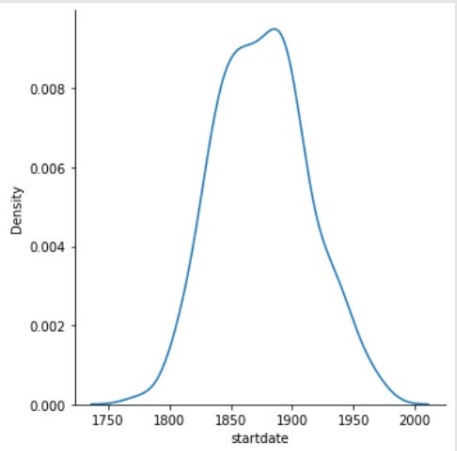
Population (n=176)	
text length (cleaned text) in characters	1.6k – 16.8M (mean: 0.5M, median: 110k)
publication year	1776 – 1971 (mean: 1875, median: 1875)
	

Table 1: Statistics of the population

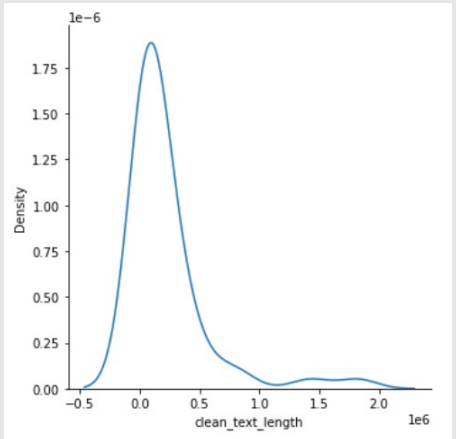
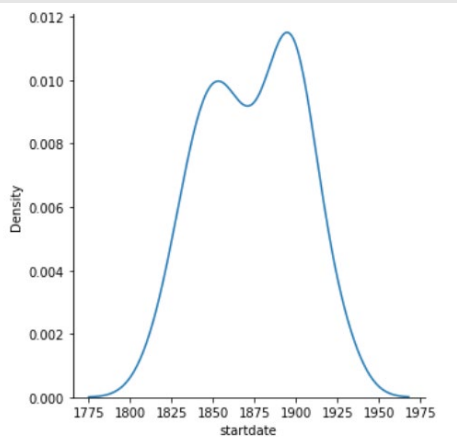
Corpus (n=50)	
text length (cleaned text) in characters	8k – 1.8M (mean: 219k, median: 95k)
publication year	1815 – 1928 (mean: 1873, median: 1876)
	

Table 2: Statistics of the sample

Looking at basic statistics of the corpus will fortify this assumption. As with the population, the vast majority of the items is placed in the 19th and early 20th century. Regarding the text length, also a wide range is represented, but extreme values are cut.

5.2 Statistical View

Since no ground truth data is available, the usual evaluation methods with precision, recall and Fx-scores do not apply. Because three NER taggers were used in parallel on the same corpus, another method is suggested for comparing their results : We first might look at the quantities of LOC tags, and secondly compare their set by intersection and symmetric differences.

	spaCy (de news lg)	Flair multi-ner-fast	BP historic-ner-onb
LOC count	[13, 23'095] total: 73'174	[5, 17'362] total: 50'326	[4, 19'694] total: 64648
LOC length (character)	[0, 147] mean=9.61 med=9	[1, 163] mean=8.83 med=8	[0, 42] mean=7.50 med=7
LOC set (unique counts)	[10, 10'417] med= 229	[2, 7099] med=98	[2, 6369] med=99
Intersection with spaCy	-	[1, 3623] med=50	[0, 2464] med=45
Symmetric difference with spaCy	-	[10, 10'270] med=220	[12, 11'858] med=274
Intersection with Flair	-	-	[1, 2415] med=41
Symmetric difference with Flair	-	-	[4, 8638] med=143

Table 3: LOC statistics and matching ranges of the LOC sets of the three models

Despite there are differences in the basics statistics of count of the three taggers, their ranges seem reasonable in comparison. A slightly diminishing character length of the entities might take place. We might observe that the spaCy tagger delivers the most LOC tags, which presumably applies for small and large texts likewise. But speaking in median numbers, only a quarter of those tags are congruent with the other NER taggers. Flair and BytePair embedding taggers are here more similar by their numbers. Here, roughly the half of their own LOC set is also delivered by the other tagger. This might be a sign that more false tags are delivered by spAcy.

Looking at the symmetric differences of the LOC sets, a similar slight difference in median numbers could be read out. But there seems to be also a notable number LOC tags, which are detected only by the Flair or BytePairs tagger.

5.3 Individual Assessment

Since there are notable differences in th LOC sets of the three NER taggers, one might take a look into the assigned tags themselves by a random choice. The general conclusions from statistic view can be confirmed : spaCy tags most NERs, and presumably has also the longest NER regarding character length. Moreover, the assumption, that the greater number of detected LOCs maybe a sign of more false positive results, can be carefully kept.

e-rara_id	spaCy	Flair	BPEmb
17996385 Ehrrerbietige Vorstellung des großen Kantonalvereins im Kanton Bern, welcher sich zu Aufhebung von	'Kanton Bern', 'Republik', 'Canton Bern', 'Republik', 'Allmenden', 'Weiden', 'Menschengedenken', 'Dieß', 'Erde', 'Reichern', 'Dorf magnaten', 'Reichern', 'Aermern',	'Bern', 'Republik Bern', 'Canton Bern', 'Nothdurft', 'Solothurn', 'Nothdurft, SLOC'	'Bern', 'Bern', 'Bern', 'Solothurn, '

eingeschlichenen Mißbräuchen und Vorrechten... (1835)	'Ueberfluß', 'Solothurn', 'Ste'		
22598462 Entwurf-Gesetz über die Correction der Gewässer des Seelandes (1861)	'Juragewässer-', 'Ueberschwemmungen', 'Möösern', 'Seelandes', 'Bieler-' , 'Aare', 'Aare', 'Aarberg', 'Büren', 'Leugeneu', 'Ausmündung', 'Aare', 'Großen Mooses Obermoos', 'Aarberg', 'Nidau', 'Erlach', 'Kantonen', 'Bund', 'Bundes', 'Mehrwerth', 'Privaten Corrections- und Entsumpfungsge-', 'Bundes', 'Landes', 'Grundeigenthums', 'Zinsfüße', 'Zinsbetreffniß', 'Regierungs-Rath', 'Der Staat', 'Vund', 'Aare', 'Büren', 'Morgenthal', 'Zihl', 'Neuen- burger- und Bielersee', 'Parthieen', 'Großen Mooses Utermoos', 'Mitgäbe', 'Kantonen', 'Bern'	'Seelandes', 'Aare', 'Aare', 'Aarberg', 'Büren', 'Aare', 'Mooses Obermoos', 'Aarberg Nidau', 'Erlach', 'Bern', 'Großen Rathe', 'Aare', 'Büren', 'Morgenthal', 'Bielersee', 'Mooses Utermoos', 'Bern', SLOC'	'Msern', 'Bieler', '- see', 'Aarberg', 'Bren', 'Mooses', 'Obermoos', 'Aarberg', 'Nidau', 'Erlach', 'Bern', 'Bren', 'Morgenthal', '- burger', 'Bielersee', 'Mooses', 'Utermoos', 'Bern', 'Bern', '

Table 4: Two examples of assigned LOC tags

6 Outcome

The exploration of different methods for NER on localities in Bernensia items led to satisfying results. Not surprisingly, approaches with a stronger robustness against OCR errors and trained on historic data outperform the standard NLP framework implementation. A future use of NER for historic texts might bring a deeper indexing level and better access to specific research interests.

For a fruitful further analysis, a deeper comparison between Flair and BytePair embedding results should be done. Also, exploration of a bigger corpus with more special text types could be illuminative to clarify the boundaries of these approaches.

7 Figures

Figure 1: Item view e-rara with download links (PDF, RIS, fulltext)	4
Figure 2: Examples of digitized images and their fulltext	5
Figure 3: Word embeddings in bidirectional LSTM, Lample et al. (2016), p.3	9
Figure 4: Building of word embeddings, Lample et al. (2016), p. 5	9
Figure 5: Word embedding classes in Flair	9
Figure 6: Building of contextual string embeddings, Akbik/Blythe/Vollgraf 2018, p. 1641	10
Figure 7: Byte-Pair Encoding underlying BytePair embeddings, Heinzerling et al. (2018), p. 2989	11
Figure 8: Example function for finding items on e-rara website according to their collection and save a CSV-file of items with their fulltext links to local machine and Google Drive.	12
Figure 9: Script for shallow text preprocessing.	13
Figure 10: Application of the spaCy German large model	13
Figure 11: Application of the Flair model with data subsetting due to job break-offs	14
Figure 12: Application of the BPEmb model	14

8 Tables

Table 1: Statistics of the population	15
Table 2: Statistics of the sample	15
Table 3: LOC statistics and matching ranges of the LOC sets of the three models	16
Table 4: Two examples of assigned LOC tags	17

9 Bibliography

Akbik, A. *et al.* (2019) 'FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP', in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 54-59. doi: 10.18653/v1/N19-4010.

Akbik, A., Bergmann, T. and Vollgraf, R. (2019) 'Multilingual Sequence Labeling With One Model', in. *Northern Lights Deep Learning Workshop 2019, Tromsø ('North Pole'), 9-10 January 2019*, p. 3. Available at: <https://alanakbik.github.io/papers/nldl2019.pdf>.

Akbik, A., Blythe, D. and Vollgraf, R. (2018) 'Contextual String Embeddings for Sequence Labeling', in *Proceedings of the 27th International Conference on Computational Linguistics. COLING 2018*, Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1638-1649. Available at: <https://www.aclweb.org/anthology/C18-1139> (Accessed: 2 April 2021).

flairNLP/flair (2021). flair. Available at: <https://github.com/flairNLP/flair> (Accessed: 2 April 2021).

Heinzerling, B. and Strube, M. (2018) 'BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages', in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. LREC 2018, Miyazaki, Japan: European Language Resources Association (ELRA). Available at: <https://www.aclweb.org/anthology/L18-1473> (Accessed: 8 April 2021).

Krishnan, V. and Ganapathy, V. (2005) 'Named Entity Recognition'.

Lample, G. *et al.* (2016) 'Neural Architectures for Named Entity Recognition', *arXiv:1603.01360 [cs]*. Available at: <http://arxiv.org/abs/1603.01360> (Accessed: 9 April 2021).

- Mikheev, A., Moens, M. and Grover, C. (1999) 'Named entity recognition without gazetteers', in *Ninth Conference of the European Chapter of the Association for Computational Linguistics*.
- Nadeau, D. and Sekine, S. (2007) 'A survey of named entity recognition and classification', *Lingvisticæ Investigationes*, 30(1), pp. 3–26. doi: 10.1075/li.30.1.03nad.
- Salton, G., Wong, A. and Yang, C. S. (1975) 'A vector space model for automatic indexing', *Communications of the ACM*, 18(11), pp. 613–620. doi: 10.1145/361219.361220.
- Sennrich, R., Haddow, B. and Birch, A. (2016) 'Neural Machine Translation of Rare Words with Subword Units', in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2016, Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. doi: 10.18653/v1/P16-1162.
- Tjong Kim Sang, E. F. and De Meulder, F. (2003) 'Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition', in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147. Available at: <https://www.aclweb.org/anthology/W03-0419> (Accessed: 2 April 2021).