Transfer Learning for Historical Corpora: An Assessment on Post-OCR Correction and Named Entity Recognition

Konstantin Todorov[a], Giovanni Colavizza[a]

[a] *University of Amsterdam, The Netherlands*

Abstract

Transfer learning in Natural Language Processing, mainly in the form of pre-trained language models, has recently delivered substantial gains across a range of tasks. Scholars and practitioners working with OCRed historical corpora are thus increasingly exploring the use of pre-trained language models. Nevertheless, the specific challenges posed by historical documents, including OCR quality and linguistic change, call for a critical assessment of the use of pre-trained language models in this setting. We consider two shared tasks, ICDAR2019 (post-OCR correction) and CLEF-HIPE-2020 (Named Entity Recognition, NER), and systematically assess using pre-trained language models with data in French, German and English. We find that using pre-trained language models helps with NER but less so with post-OCR correction. Pre-trained language models should therefore be used critically when working with OCRed historical corpora. We release our code base, in order to allow replicating our results and testing other pre-trained representations.

Keywords

Digital cultural heritage, Transfer learning, Multi-task learning, BERT, Post-OCR correction, Named Entity Recognition (NER)

1. Introduction

The digitisation of written records of cultural value, such as books, historical newspapers and archival materials, has been advancing since decades [46]. While much remains to be digitised, by now scholars and the public have at their disposal large corpora of digitised records [44]. As a consequence, the interlocked questions of their accessibility and use for research come to the forefront. For example, most digitised historical records still have to be made fully searchable like other large textual collections such as the Web.

There are several factors which make the accessibility of collections of digitised historical records, and their use as data for research, challenging [41, 18, 8]. Those include the need to use Optical or Handwritten Character Recognition (O/HCR) to extract texts from images, which is error prone and of varying quality. Challenges also include language variability over time and the lack of linguistic resources for automatic processing. These challenges both call for, but at the same time question, the use of modern machine learning techniques to improve the accessibility of digitised historical records. Perhaps the most successful technique in this respect is *transfer learning*.

Transfer learning aims to transfer knowledge from a general-purpose source task to a specialised target task [38, 48]. Transfer learning can help in (i) gaining faster convergence; (ii) use

CHR 2020: Workshop on Computational Humanities Research, November 18–20, 2020, Amsterdam, The Netherlands

k.todorov@uva.nl (K. Todorov); g.colavizza@uva.nl (G. Colavizza)

0000-0002-7445-4676 (K. Todorov); 0000-0002-9806-084X (G. Colavizza)

fewer computation resources; (iii) overcome the lack of linguistic resources, such as annotated data [13]. For transfer learning to be successful, the source and target domains and/or tasks need to be related in some way. Examples of transfer learning include multitask learning [14, 16, 23, 43], when two or more related tasks are learning jointly, and sequential transfer learning [60, 42, 40, 15], when the source task is learned first and then some components of the original architecture are used and adapted to the target task.

Both multi-task and sequential transfer learning provide advantages. Multitask learning often allows to achieve better generalisation [5], it provides a model with the capacity to *eavesdrop*, i.e., learn to do one task through another, and finally it is a form of regularisation [54]. Sequential transfer learning, on the other hand, has the benefit of fast adaptation to new tasks, sometimes even without additional training. A widely successful example of sequential transfer learning in Natural Language Processing (NLP) are *language models* used as embeddings to represent textual inputs, such as BERT [17].

Recently, transfer learning has started to be applied on historical collections in a variety of ways. Examples include the representation and measurement of semantic change [52, 24] or extracting named entity information [31]. Nevertheless, several questions remain open, in particular due to the challenges which historical corpora pose. It remains unclear when (i.e., for which tasks, languages, etc.) and how (i.e., taking which approach) transfer learning can be successfully applied on historical collections. Given that most pre-trained language models have been trained on modern-day, high-resource languages (e.g., Wikipedia in English), their applicability to historical collections deserves more than an afterthought.

In this work, we start bridging the gap of systematically assessing transfer learning for historical textual collections. We consider two tasks: the *ICDAR2019 Competition on Post-OCR Text Correction* [45] and the *CLEF-HIPE-2020* challenge on Named Entity Recognition, Classification and Linking [20]. These tasks are of importance to practitioners as they directly influence the usability and accessibility of digitised historical collections. We propose a general architecture made of a modular embedding layer, which allows us to perform ablation studies using combinations of newly trained and pre-trained embeddings, and task-specific layers. For both tasks, we consider English, German and French as languages, and use the task data and evaluations.

2. Empirical setup

In order to assess the added value of transfer learning on a variety of tasks, we use a general approach illustrated in Figure 1. We represent the input using a modular embedding layer which can include combinations of newly trained and pre-trained embeddings. Embeddings can be at the character, sub-word or word level, and can be combined flexibly. In this way, we can perform ablation studies and measure the impact of using pre-trained embeddings. Each task is performed using the embedding layer to create an input representation, followed by task-specific layers and evaluation. In this section, we describe the challenges and our approach to them. Further details are provided as an appendix.

2.1. Post-OCR Correction

OCR is an often noisy process which introduces errors in the extracted text. One option to improve the quality of its results is to attempt to correct the extracted text using linguistic knowledge. We work with the *ICDAR2019 Competition on Post-OCR Text Correction* [45].

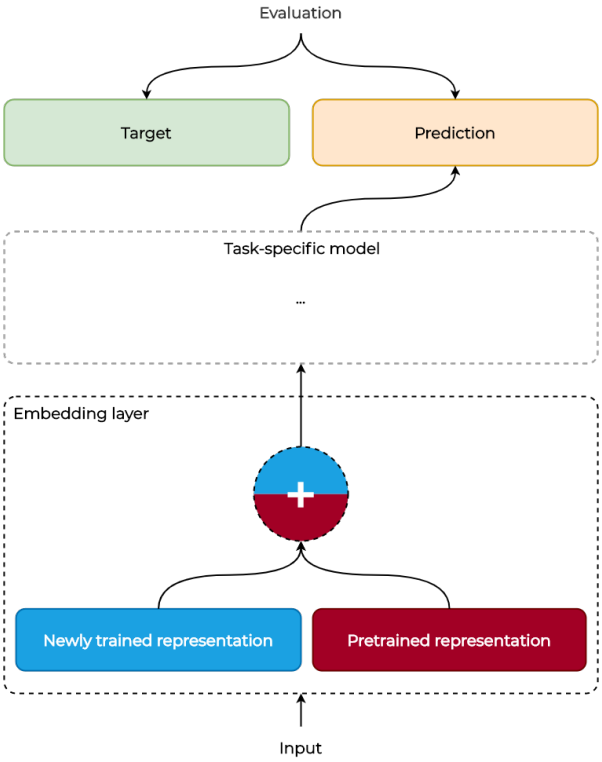


Figure 1: General model architecture for assessing transfer learning on a variety of tasks.

Two sub-tasks are proposed as part of this challenge: (i) the detection of the OCR errors and (ii) the correction the OCR errors. For this study, we focus on the latter.

2.1.1. Previous work

The quality of OCRed texts is crucial for achieving stable performance in NLP [58, 36, 11]. Noisy OCR can be the result of the acquisition process, of the document conservation state or even of some of its properties, such as the use of worn-out types [53]. This makes post-processing techniques such as post-OCR correction, potentially important as they could overcome some of the noise introduced during OCR. However, still little work has been devoted to this area. Nevertheless, previous work has found that working at narrower linguistic levels, such as focusing on characters or sub-words instead of words, leads to better results [7].

2.1.2. Data

The data which is provided includes noisy OCR of printed texts from different sources and ten languages (English, French, German, Finish, Spanish, Dutch, Czech, Bulgarian, Slovak and Polish). We focus on English, German and French which taken together total 13 628 documents for 17 884 116 characters. We keep the same 80-20% data split provided by the organisers, and use the ground truth to OCRed text versions aligned at the character level. This choice allows our models to disregard having to learn how to perform sequence alignment too.

Since we have considerably less data for English and French when compared to German, we also make use of data in these languages from a previous edition of the challenge, namely *ICDAR2017 Competition on Post-OCR Text Correction* [10]. This allows us to add 12 000 000 OCRed characters along with their corresponding ground truth, approximately half in English and half in French. We further augment the dataset by adding OverProof data[1] in English. We use 100% of ICDAR2017's and Overproof's data as training data. We split ICDAR2017 and ICDAR2019 documents into sequences of 50 characters in length. OverProof data are already split line-by-line instead.

2.1.3. Evaluation

Evaluation is performed by making use of the sum of the Levenshtein distances [35] of the documents between the corrected candidates and the corresponding ground truth, using the raw OCR text as baseline. The sum is used to calculate the improvement over the Levenshtein distance of the original OCR and the ground truth. We additionally report a normalised Jaccard similarity for each run, calculated at the character level and without taking into account their sequencing. We normalise the Jaccard similarity using the length of the sequences. We report the average for both metrics across the sequences that are evaluated. We note that our results cannot be immediately compared with the reported results from ICDAR2019 [45], for the reason that these are calculated over erroneous tokens only (i.e., they are based on the outputs of the challenge's task 1), while we consider the whole text.

2.1.4. Model

We combine the embedding layer with an encoder-decoder architecture with attention [4], typically used for translation. We thus consider the raw OCR version of a text and ground truth as two distinct languages, and train the model to 'translate' from one to the other. A key difference from a language translation setup is that we rely on the same character and sub-word vocabularies for input and output instead of having two distinct vocabularies. Our encoder-decoder architecture works at the character level, to be able to correct errors at that level.

Embedding layer We experiment with newly trained character-level embeddings, combined with BERT and in-domain sub-word embeddings. Sub-word embeddings are concatenated to character-level embeddings, as illustrated in Figure 2. Concatenation brings significant speed-up benefits while performing, in our setting, similarly to other approaches such as using extra RNNs or CNNs layers. As sub-word embeddings, we use *bert-base-cased* for English, *bert-base-german-cased* for German and *bert-base-multilingual-cased* for French. Furthermore, we use FastText [27] embeddings which have been pre-trained on historical newspaper corpora (in-domain) and were provided by the organisers of the NER task. We always use WordPiece sub-word tokenization [51, 30].

Encoder The concatenated embeddings are used in the encoder, which produces a sequence of hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_M$, one for each character embedding. The encoder in our case is a Bi-directional Gated Recurrent Unit (Bi-GRU) [12]. The Bi-GRU produces two representations, one left-to-right and one right-to-left, as follows:

[1] https://overproof.projectcomputing.com.

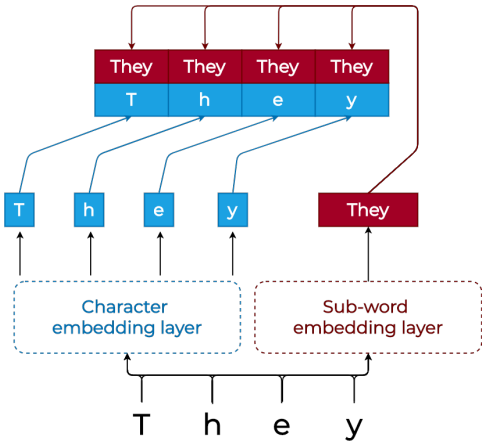


Figure 2: Sub-word and character embedding concatenation for the post-OCR correction encoder.

$$\overrightarrow{\mathbf{h}} = \left[\overrightarrow{\mathbf{h}_1}, \ldots \overrightarrow{\mathbf{h}_M}\right] \text{ and } \overleftarrow{\mathbf{h}} = \left[\overleftarrow{\mathbf{h}_1}, \ldots \overleftarrow{\mathbf{h}_M}\right]$$

Where each hidden state is calculated as:

$$\overrightarrow{\mathbf{h}_i} = \text{GRU}\left(\mathbf{x}_i, \overrightarrow{\mathbf{h}_{i-1}}\right) \text{ and } \overleftarrow{\mathbf{h}_i} = \text{GRU}\left(\mathbf{x}_i, \overleftarrow{\mathbf{h}_{i+1}}\right)$$

The final representation of the Bi-GRU is:

$$\mathbf{h} = \left[\overrightarrow{\mathbf{h}}; \overleftarrow{\mathbf{h}}\right]$$

Where ";" stands for the concatenation between the left-to-right and right-to-left passes.

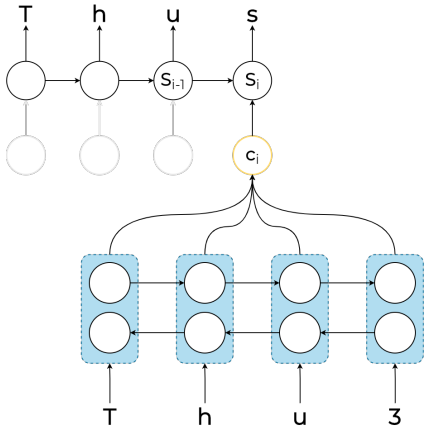


Figure 3: Post-OCR correction model, decoder pass.

Decoder This final hidden state is then forwarded to the decoder. This is similar to the encoder with two major differences. The whole pass can be seen in Figure 3. We first use an embedding layer which does not make use of any pre-trained embeddings. When we test sharing the same embedding layer in the encoder and the decoder, we skip pre-trained model embeddings for the decoder. We justify this with the fact that we decode information character by character, thus making pre-trained embeddings not applicable – since in our case they work at the sub-word level. Furthermore, we use attention [4]. The final representation of each target character can be formulated as:

$$\mathbf{s}_i = \text{GRU}\left(\mathbf{s}_{i-1}, \mathbf{y}_{i-1}, \mathbf{c}_i\right)$$

where $\mathbf{s}_i$ is the decoder hidden state, for which we use a GRU. We decode each character separately by feeding the encoded final state $\mathbf{h}$ to the attention mechanism which then outputs a context vector $\mathbf{c}_i$ (shown in yellow). This is done at each step where the attention dynamically selects that part of the encoded source sentence that is considered most relevant for the current target character. Additionally, we use the previously decoded character hidden state as input knowledge, and we label them as $\mathbf{y}_{i-1}$.

After computing the decoder state $\mathbf{s}_i$, we use a non-linear function g – a *softmax* in our case – and calculate the probability of the target character $\mathbf{y}_i$ for this step:

$$p\left(\mathbf{y}_i|\mathbf{y}_{<i}, X\right) = g\left(\mathbf{s}_i\right)$$

Here, $X = (x_1, \ldots, x_M)$ is the input sequence and g (a softmax) provides a probability vector of the same size as the character vocabulary. This is a distribution over all target characters, where at inference time we select the character with the highest probability as output. This setup ends with a cross entropy loss that is used to maximise the probability of selecting the correct character at each step.

2.2. Named Entity Recognition

Named Entity (NE) processing is increasingly applied to historical collections [31]. We participate in the *CLEF-HIPE-2020* challenge [20] which focuses on two of the most pressing NE processing tasks, namely Named Entity Recognition and Classification (NERC) and Named Entity Linking (NEL). We only focus on the former task (NERC) in what follows.

The CLEF-HIPE-2020 NERC task is further split into two sub-tasks. Sub-task 1 considers *coarse-grained* NE types, that is to say the most general entity tags, for example a grouping such as `loc` combines all location entities and sub-entities. Sub-task 2 considers *fine-grained* entity types. Following previous examples, we now have detailed sub-entities such as `loc.adm.town` which corresponds to an administrative town or `loc.adm.nat` which corresponds to an administrative unit at the national level. Both sub-tasks make a distinction between the literal and the metonymic senses of an entity. *Metonymy* stands for a figure of speech in which a specific thing is referred to by the name of something closely related to it. Additionally, detection and classification of nested entities of depth one and of entity mention components (such as title, function, etc.) is required. Table 1 shows a comparison of the two sub-tasks and the differences in expected predictions.

2.2.1. Previous work

Recently, the task of named entity recognition has seen major improvements thanks to the inclusion of novel deep learning techniques and the usage of learned representations (em-

	Sub-task 1	Sub-task 2
NE mentions with coarse types	yes	yes
NE mentions with fine types	no	yes
Metonymic sense	yes	yes
NE components	no	yes
Nested entities of depth one	no	yes

Table 1
NERC sub-task comparison

beddings) [1, 34, 31]. Named entity recognition and classification (NERC) is also of great importance in the digital humanities and cultural heritage. However, applying existing NERC techniques is also made challenging by the complexities of historical corpora [55, 26]. Crucially, transferring NE models from one domain to another is not straightforward [59] and in many cases performance is consequently greatly impacted [56].

2.2.2. Data

The data consists of Swiss, Luxembourgish and American historical newspapers written in French, German and English respectively, collected in the context of the IMPRESSO project [19]. Newspaper articles were sampled over a period spanning 1790 to 2010. We have in total 569 articles and 1 894 741 characters with a vocabulary of 151 unique characters. It is also important to note that, as it is often the case with NE tasks, most of the tokens are labelled as "Other" or `O`. In total, we have that 94.92%, 95.95%, and 96.5% of all English, French and German tokens are labelled as "Other", respectively. Furthermore, some tags are particularly sparse. For nested and the two metonymic tags, we have more than 99.5% non-entities across languages. More information about the data distribution can be found in the Appendix.

The data are released in `IOB` format (Inside-Outside-Beginning format), which we also use during training and evaluating. Some pre-processing was done by the organisers, applying tokenization using white space splitting and flagging some tokens as `NoSpaceAfter` in order to allow full words to be reconstructed. The organisers also provided in-domain pre-trained FastText embeddings for all languages, which we used for both post-OCR correction and NER [27]. The organisers also provide Flair embeddings [2] which we do not use here.

2.2.3. Evaluation

The task is evaluated in terms of Precision, Recall and F1-measure. Two evaluation scenarios are considered: (i) strict (exact boundary matching) and (ii) relaxed (fuzzy boundary matching). Fuzzy scoring works in a relaxed way, allowing fuzzy boundary matching of entities. That is if an entity is only partially recognised, e.g., if 4 out of total of 6 tokens are recognised correctly, this is still considered a successful recognition. Conversely, strict matching requires all tokens to match with exact boundary matching – in previous example this would require 6 out of 6 total tokens to be predicted correctly. Each entity type is evaluated independently, by using the entity-level micro average. It must be noted that all evaluations are performed on entities (inside or at the beginning) only and do not consider predictions of outside tokens or tokens originally labelled as outside (`O`). The organisers provide a simple CRF [32] baseline

model based on the `sklearn_crfsuite` library [1] and using hand-crafted features.

2.2.4. Model

We combine our modular embedding layer with a Bidirectional LSTM-CRF (Bi-LSTM-CRF) for Named Entity Recognition [34], with the only simplification of removing the `tanh` non-linearity after the LSTM. We consider embeddings at varying input granularities, including: (i) *character-*, (ii) *sub-word-* and (iii) *word-level*. As a sanity test, we applied our model on the modern-day CoNLL-2003 dataset [49], achieving results comparable to current state of the art. Due to the challenge consisting of six prediction tasks (one for each tag type), we consider a multi-task approach where we repeat our final fully connected and CRF layers for each task and share the remaining ones. This is illustrated in Figure 4. When testing for single-task, we use the same architecture with a single final layer for a single entity type.

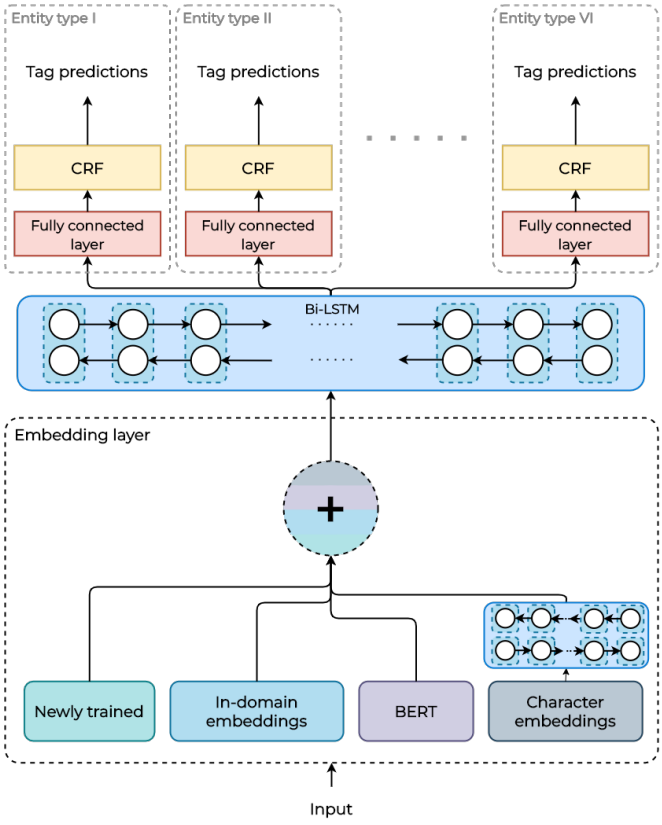


Figure 4: NERC multi-task model architecture. Our single-task architecture is identical and only contains a final layer for one entity type.

Embedding layer We consider four different embeddings. (i) *Character embeddings* with an embedding layer followed by a bidirectional LSTM. We use a character-level custom vocabularies for each language built from the training and validation data sets. (ii) *BERT embeddings* at sub-word level. We use *bert-base-multilingual-cased* for French, *bert-base-german-cased* for

[1] https://sklearn-crfsuite.readthedocs.io/en/latest [version used: 0.3.6, last accessed: 2020-07-24].

German, and *bert-base-cased* for English, from HuggingFace Transformers library [61]. This brings the specific limitation of only working with sequences of 512 character in maximum length. As our text sequences are usually longer, we implement a sliding-window splitting of input sequences before passing them through BERT. While splitting, we keep the first and last 5 characters of each chunk as overlap among sequential chunks. After embedding each chunk, we then reconstruct the full input sequences by averaging the embeddings of the overlapping characters. (iii) *Newly trained* embeddings at sub-word level with randomly initialised weights. We include these newly-trained embeddings to test whether parameters learned from scratch at the sub-word level, instead of just pre-trained, can help. We use the same vocabulary as with BERT. (iv) *In-domain pre-trained embeddings* provided by the task organisers are used for feature extraction only (frozen). These embeddings have size of 300 and work at the sub-word level, using the FastText library [27].

After testing different alternatives, we found that the simplest and fastest way to combine these embeddings is by concatenating them, resulting in concatenated sub-word embeddings of size 1836 when the largest configuration and all embedding options are used.

Task-specific layer based on a Bi-LSTM-CRF [34]. *The Bi-LSTM-CRF* works on the concatenated sub-word embeddings, which are then merged at word level by taking their average before feeding the representation through a fully connected layer which then outputs tag probabilities for each token. We tested concatenating embeddings before or after the Bi-LSTM, or not merging at all, and found that our approach performs best, also in accordance with previous findings [48]. A Conditional Random Field (CRF) [33] is used over the produced tag probabilities to decode the final tag predictions.

Multi-task We introduce additional output heads, one for each of the different entity types that the task aims to predict. The final two layers of the model, namely the fully connected layer and CRF, are specific to each entity type, while the rest of the architecture is shared. The individual losses for each task are summed during backpropagation. We compare using single vs multi-task approach in what follows.

Additional resources We use the Annotated Corpus for Named Entity Recognition built on top of Groningen Meaning Bank (GMB) [9][1]. This dataset is annotated specifically for training NER classifiers, and contains most of the coarse-grained tag types which occur in the English dataset provided by organisers. We consolidate some tags with the same meaning but different labels ourselves. The dataset contains in total 1,354,149 tokens of which 85% are labelled as `O` originally. We convert the tag types that are not part of this challenge to `O` as well, resulting in total of 94.92% tokens having `O` literal tags.

3. Results

3.1. Post-OCR Correction

We start discussing results for post-OCR correction reporting the average Levenshtein distance and normalised Jaccard similarity. We always provide, for reference, both measures calculated on the raw OCRed text (No correction), and the % of improvement. We report results with

[1] https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus [accessed 2020-07-16].

a baseline model without pre-trained embeddings (Base), adding FastText (+ FT), BERT (+ BERT) and both. We further assess fine-tuning BERT models by unfreezing BERT embeddings from the start or after convergence. More details on model fitting and the hyper-parameters we used are given in the Appendix. We also show, in Figure 5 (in Appendix A), the normalised histogram of the Levenshtein distances for all documents and languages, comparing the raw OCRed text, the Base model and the best model we found for each language.

Configuration	Levenshtein distance		Normalised Jaccard similarity	
	Average	% improvement	Average	% improvement
No correction	3.568	–	0.926	–
Base	**3.369**	**5.579**	0.934	0.824
Base + FT	3.442	3.522	0.934	0.855
Base + BERT	3.393	4.896	0.934	0.881
Base + FT + BERT	3.389	5.020	0.934	0.850
+ Fine-tuning (unfreezing, from start) BERT				
Base + BERT	3.441	3.565	0.933	0.722
Base + FT + BERT	3.397	4.784	**0.936**	**1.008**
+ Fine-tuning (unfreezing, after initial convergence) BERT				
Base + BERT	3.401	4.668	0.935	0.923
Base + FT + BERT	3.448	3.347	0.935	0.900

Table 2
Post-OCR correction, French. Best result per table and column is given in bold, second best is underlined.

Starting with French (Table 2), we find that the raw OCRed texts for this language are already of very high quality, thus we are able to get but minor improvements. Furthermore, using pre-trained embeddings does not seem to help in any significant way. When considered using the Levenshtein distance, the best model is one without pre-trained embeddings (Base), while fine-tuning BERT leads to slightly higher performance under the Jaccard similarity. Nevertheless, these gains are very marginal. German raw OCRed texts are of lower quality than French, we are thus able to get substantial improvements with correction (up to 64% with Levenshtein distance, Table 3). While our post-OCR correction gains are substantial, the impact of pre-trained embeddings remains negligible or non-existent.

Lastly, with English we face another challenge namely the bad quality of the ground truth (Table 4). As a consequence, results are largely inconclusive and our model fails to learn anything significant in order to improve upon the original sequence pairs. Furthermore, in many instances, the ground truth proved to contain errors while our models were suggesting valid corrections. For example:

- input: “any glimpse, or sign *f Eight trom the Earth, it”
  ground truth: “any glimpse, or ffgn of Light from the Earth, it”
  prediction: “any glimpse, or sign of Eight from the Earth, it”

- input: “• Henry K Concert—AU fIddledidee—Triumph* ot”
  ground truth: “.... Henry A Concert All ddledidee Triumphs of”
  prediction: “Henry a Concert All dreest the”

Configuration	Levenshtein distance		Normalised Jaccard similarity	
	Average	% improvement	Average	% improvement
No correction	12.008	–	0.656	–
Base	4.302	64.172	**0.900**	37.290
Base + FT	4.439	63.034	0.896	36.584
Base + BERT	4.464	62.827	0.896	36.630
Base + FT + BERT	5.393	55.088	0.872	32.938
+ Fine-tuning (unfreezing, from start) BERT				
Base + BERT	**4.283**	**64.334**	**0.900**	**37.324**
Base + FT + BERT	4.340	63.863	0.899	37.131
+ Fine-tuning (unfreezing, after initial convergence) BERT				
Base + BERT	4.344	63.828	0.898	37.039
Base + FT + BERT	4.411	63.271	0.898	36.908

Table 3
Post-OCR correction, German. Best result per table and column is given in bold, second best is underlined.

Configuration	Levenshtein distance		Normalised Jaccard similarity	
	Average	% improvement	Average	% improvement
No correction	**9.397**	–	0.825	–
Base	9.955	-5.944	0.822	-0.310
Base + FT	9.864	-4.971	0.825	-0.005
Base + BERT	10.228	-8.840	0.822	-0.364
Base + FT + BERT	9.992	-6.338	0.825	-0.006
+ Fine-tuning (unfreezing, from start) BERT				
Base + BERT	9.835	-4.665	0.825	0.062
Base + FT + BERT	9.787	-4.151	0.825	0.078
+ Fine-tuning (unfreezing, after initial convergence) BERT				
Base + BERT	9.724	**-3.483**	**0.829**	**0.510**
Base + FT + BERT	9.927	-5.639	0.826	0.108

Table 4
Post-OCR correction, English. Best result per table and column is given in bold, second best is underlined.

In conclusion, we find that for post-OCR correction pre-trained embeddings do not provide any significant gain over a baseline with newly trained embeddings. Considering that the convergence speed (and hence compute cost) is higher when using pre-trained embeddings, in particular when fine-tuning them (Appendix, Table 12), transfer learning does not appear to help with post-OCR correction.

We underline that the data provided for the ICDAR2019 challenge is far from uniform across languages, and this has a major impact on our results. While data for German contains bad raw OCRed texts and good ground truth (the ideal setting for post-OCR correction), data for French contains high-quality raw OCRed texts and data for English contains a low-quality ground truth. We still see how the inclusion of pre-trained embeddings makes our model able to correct words that have incorrect ground truth, thus confusing the learning process even more.

Furthermore, we observe how we can get better results with a single-origin dataset (German), than with similarly sized datasets (French and English) originating from a combination of different sources.

3.2. Named Entity Recognition

We report results for the three languages part of the task, namely French, German and English, using the official test set v1.3 [20] We base our reporting from our submission in the challenge [57]. In addition to that, we report using *multi-segment* and *document-level* split types for French and German and *segment* split type for English, since our English training data lacks the document level. All results are reported in the two scoring approaches used in the challenge – *fuzzy* and *strict*. For each scoring approach, we provide *precision* (P), *recall* (R) and *F-score* (F). We report the baseline model provided by organisers for reference, reminding the reader that the baseline model always uses a document-level split. We also report the baseline model results on our English data. More details on model fitting and the parameters are provided in the Appendix.

Configuration	Literal coarse					
	Fuzzy			Strict		
	P	R	F	P	R	F
Baseline (organisers)	**.736**	.454	**.562**	**.531**	.327	**.405**
Baseline (ours)	.377	.612	.466	.190	.310	.236
Base	.307	.576	.401	.139	.261	.181
Base + CE	.300	**.640**	.409	.139	.296	.189
Base + CE + FT	.309	.627	.414	.140	.285	.188
Base + CE + BERT	.457	.538	.494	.261	.307	.282
Base + CE + BERT – newly	.475	.535	.503	.265	.298	.281
Base + CE + FT + BERT	.408	.590	.482	.229	**.332**	.271
Base + CE + FT + BERT – newly	.415	.528	.465	.179	.227	.200
+ Fine-tuning (unfreezing) BERT						
Base + CE + BERT	.421	.622	.502	.203	.301	.243
Base + CE + BERT – newly	.493	.530	.511	.292	.314	.303
Base + CE + FT + BERT	.404	.582	.477	.205	.296	.242
Base + CE + FT + BERT – newly	.462	.508	.484	.261	.287	.274

Table 5
NERC, English, segment split. Best result per table and column is given in bold, second best is underlined

We order the different configurations for all languages in order to assess the impact of transfer learning. We start with the simplest Base model which is only using newly trained sub-word embeddings and no pre-trained information of any type. Then we continue by adding character embeddings which use RNN (+ CE). Due to the significant improvements observed by adding character embeddings, we keep them enabled in all of our next reported setups. We further report results that were achieved by adding firstly the (frozen) FastText embeddings provided by organisers (+ FT), then (frozen) BERT embeddings (+ BERT), and finally both. Whenever BERT is enabled, we also report runs where we disable newly trained embeddings (– newly). Eventually, we report three different setups where we unfreeze BERT and fine-tune them on the task at hand. Due to the long sequence lengths when working on document level, we are unable to perform fine-tuning of BERT at the document level. We therefore report the results

of fine-tuning BERT only using multi-segment split. All models use the multi-task approach, except for one single-task run using all available embeddings (single).

Results for French (Table 6) and German (Table 7) are aligned. Firstly, adding character-level embeddings and BERT consistently improves results. Better results overall are obtained with a single-task approach and using all available embeddings, including newly trained ones. A document-level split, following this configuration, perform best across the board. We also see that most of our configurations struggle on tasks with sparser annotations such as Metonymic and Nested, in particular for German where we are not predictive at all on nested tags. Furthermore, fine-tuning BERT does not seem to improve results.

For completeness, we report results for English in Table 5, limited to the Literal coarse task. For a better comparison, we provide results from two baseline models: i) the baseline from the organisers and ii) the baseline model trained on the English dataset we use. Our models are mostly not able to perform beyond the provided baseline. This is likely in part due to the training data that we use originating from a different source than the evaluation data.

We see that in a scenario where data is small or lacking, transfer learning proves to be highly beneficial. The inclusion of BERT for NER brings noticeable improvements, especially when compared to the post-OCR correction challenge where improvements are negligible while data is more abundant. Furthermore, multi-task learning proves to be beneficial, often bringing better results than single-task and requiring less time for convergence. This challenge further underlines the importance of having high-quality annotated data.

Configuration	Literal coarse						Metonymic coarse					
	Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.825	.721	.769	.693	.606	.646	.541	.179	.268	.541	.179	.268
Base	.776	.690	.730	.618	.550	.582	.500	.424	.459	.495	.420	.454
Base + CE	.806	.739	.771	.649	.594	.620	.552	.379	.45	.545	.375	.444
Base + CE + FT	.789	.780	.784	.650	.642	.646	.481	.339	.398	.468	.330	.387
Base + CE + BERT	**.886**	.801	.841	**.782**	.707	.743	.424	.397	.410	.410	.384	.396
Base + CE + BERT – newly	.859	.818	.838	.719	.685	.702	.417	.384	.400	.417	.384	.400
Base + CE + FT + BERT	.866	.836	.851	.767	.739	.753	.664	.362	.468	.656	.357	.462
Base + CE + FT + BERT – newly	.864	**.848**	**.856**	.765	**.751**	**.758**	**.766**	.321	.453	**.766**	.321	.453
Base + CE + FT + BERT (single)	.872	.835	.853	.769	.737	.753	.036	.069	.000	.036	.069	.000
+ Fine-tuning (unfreezing) BERT												
Base + CE + BERT	.876	.824	.849	.775	.729	.751	.442	.375	.406	.432	.366	.396
Base + CE + BERT – newly	.877	.804	.839	.775	.711	.742	.754	.384	.509	.754	.384	.509
Base + CE + FT + BERT	.857	.836	.846	.759	.741	.750	.551	.482	.514	.541	.473	.505
Base + CE + FT + BERT – newly	.845	.838	.842	.742	.737	.740	.659	**.500**	**.569**	.659	**.500**	**.569**

(a) coarse grained entity type, multi-segment split

Configuration	Literal fine						Metonymic fine						Component						Nested					
	Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.838	.693	.758	.644	.533	.583	.564	.196	.291	.538	.187	.278	.799	.531	.638	.733	.487	.585	.267	.049	.082	.267	.049	.082
Base	.800	.670	.729	.548	.459	.499	.476	**.451**	.463	.472	**.446**	.459	.774	.531	.630	.692	.475	.563	.383	.140	.205	.333	.122	.179
Base + CE	.825	.708	.762	.562	.482	.519	.594	.366	.453	.594	.366	.453	.779	.556	.649	.720	.514	.600	**.500**	.067	.118	.364	.049	.086
Base + CE + FT	.801	.763	.781	.568	.541	.554	.567	.228	.325	.533	.214	.306	.762	.598	.670	.682	.535	.600	.425	**.207**	**.279**	**.375**	**.183**	**.246**
Base + CE + BERT	**.889**	.781	.831	.658	.578	.616	.532	.366	.434	.519	.357	.423	.803	.579	.673	.715	.515	.599	.000	.000	.000	.000	.000	.000
Base + CE + BERT – newly	.865	.748	.802	.613	.530	.568	.54	.241	.333	.540	.241	.333	.821	.504	.625	.732	.449	.557	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.866	.818	.842	.672	.634	.653	.702	.263	.383	.643	.241	.351	.804	.563	.662	.712	.499	.587	.357	.03	.056	.143	.012	.022
Base + CE + FT + BERT – newly	.873	.820	**.846**	.672	.631	.651	**.771**	.241	.367	**.743**	.232	.354	**.842**	.546	.663	**.774**	.503	.61	.393	.067	.115	.286	.049	.083
Base + CE + FT + BERT (single)	.868	.818	.842	.676	.636	.655	.538	.442	.485	.533	.438	**.48**	.752	**.677**	**.713**	.659	**.594**	**.625**	.000	.000	.000	.000	.000	.000
+ Fine-tuning (unfreezing) BERT																								
Base + CE + BERT	.877	.806	.840	.654	.600	.626	.434	.379	.405	.429	.375	.400	.770	.598	.673	.673	.523	.588	.267	.049	.082	.133	.024	.041
Base + CE + BERT – newly	.885	.782	.830	.672	.593	.630	.739	.290	.417	.705	.277	.397	.818	.524	.639	.745	.477	.582	.107	.018	.031	.071	.012	.021
Base + CE + FT + BERT	.871	.814	.842	**.687**	.642	**.664**	.568	.411	.477	.543	.393	.456	.741	.672	.705	.648	.587	.616	.232	.159	.188	.179	.122	.145
Base + CE + FT + BERT – newly	.852	**.837**	.845	.663	**.652**	.658	.681	.420	**.519**	.609	.375	.464	.785	.626	.697	.701	.559	.622	.333	.183	.236	.244	.134	.173

(b) fine grained entity type, multi-segment split

Configuration	Literal coarse						Metonymic coarse					
	Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.825	.721	.769	.693	.606	.646	.541	.179	.268	.541	.179	.268
Base	.812	.686	.743	.671	.566	.614	.444	.536	.486	.444	.536	.486
Base + CE	.802	.762	.782	.658	.625	.641	.575	.272	.370	.566	.268	.364
Base + CE + FT	.815	.737	.774	.673	.608	.639	.510	.469	.488	.505	.464	.484
Base + CE + BERT	.871	.831	.851	.779	.743	.760	.684	.232	.347	.684	.232	.347
Base + CE + BERT – newly	**.890**	.828	.858	.788	.733	.759	.564	.277	.371	.545	.268	.359
Base + CE + FT + BERT	.872	.828	.849	.772	.733	.752	.433	**.696**	**.534**	.428	**.688**	**.527**
Base + CE + FT + BERT – newly	.869	**.872**	.871	.780	**.782**	.781	**.755**	.357	.485	**.755**	.357	.485
Base + CE + FT + BERT (single)	**.890**	.856	**.873**	**.807**	.776	**.791**	.699	.424	.528	.691	.420	.522

(c) coarse grained entity type, document split

Configuration	Literal fine						Metonymic fine						Component						Nested					
	Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.838	.693	.758	.644	.533	.583	.564	.196	.291	.538	.187	.278	.799	.531	.638	**.733**	.487	.585	.267	.049	.082	.267	.049	.082
Base	.822	.672	.739	.594	.486	.534	.446	**.513**	.477	.419	**.482**	.448	.738	.600	.662	.657	.534	.589	**.512**	.250	.336	.350	.171	.230
Base + CE	.809	.752	.780	.586	.546	.565	.521	.223	.313	.521	.223	.313	.743	.618	.675	.650	.541	.590	.350	.171	.230	.275	.134	.180
Base + CE + FT	.811	.722	.764	.599	.534	.565	.540	.362	.433	.507	.339	.406	.759	.603	.672	.684	.544	.606	.453	.177	.254	**.406**	.159	.228
Base + CE + BERT	.885	.799	.840	.696	.629	.661	.654	.304	.415	.654	.304	.415	.719	.686	.702	.625	.596	.610	.304	.104	.155	.250	.085	.127
Base + CE + BERT – newly	**.896**	.790	.840	.675	.595	.633	.568	.223	.321	.568	.223	.321	**.808**	.603	.690	.696	.520	.595	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.883	.800	.839	.717	.649	.682	**.741**	.371	.494	.679	.339	.452	.794	.631	.703	.715	.568	.633	.341	.183	.238	.318	.171	.222
Base + CE + FT + BERT – newly	.881	.841	.861	.703	.671	.687	.705	.384	**.497**	.689	.375	**.486**	.792	.644	.710	.704	.572	.631	.233	.043	.072	.067	.012	.021
Base + CE + FT + BERT (single)	.882	**.853**	**.867**	**.729**	**.704**	**.716**	**.741**	.357	.482	**.741**	.357	.482	.734	**.726**	**.73**	.650	**.642**	**.646**	.438	**.299**	**.355**	.393	**.268**	**.319**

(d) fine grained entity type, document split

Table 6
NERC, French. The best result per table and column is given in bold, second best result is underlined

Configuration	Literal coarse						Metonymic coarse					
	Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.790	.464	.585	.643	.378	.476	**.814**	.297	.435	**.814**	.297	.435
Base	.698	.526	.600	.535	.404	.46	.559	**.602**	**.580**	.551	.593	.571
Base + CE	.685	.605	.642	.535	.473	.502	.588	.568	.578	.588	.568	**.578**
Base + CE + FT	.691	.554	.615	.528	.424	.470	.534	**.602**	.566	.534	**.602**	.566
Base + CE + BERT	.801	.675	.733	.596	.502	.545	.000	.000	.000	.000	.000	.000
Base + CE + BERT – newly	.759	.706	.732	.582	.541	.561	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.784	.724	.753	.639	.589	.613	.598	.542	.569	.598	.542	.569
Base + CE + FT + BERT – newly	**.840**	.640	.726	.696	.530	.602	.696	.466	.558	.696	.466	.558
Base + CE + FT + BERT (single)	.827	**.731**	**.776**	**.708**	**.625**	**.664**	.492	.530	.510	.472	.508	.490
+ Fine-tuning (unfreezing) BERT												
Base + CE + BERT	.756	.718	.737	.546	.519	.532	.000	.000	.000	.000	.000	.000
Base + CE + BERT – newly	.752	.718	.734	.56	.534	.547	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.738	.678	.707	.575	.528	.551	.562	.500	.529	.543	.483	.511
Base + CE + FT + BERT – newly	.802	.689	.741	.658	.565	.608	.621	.521	.567	.616	.517	.562

(a) coarse grained entity type, multi-segment split

Configuration	Literal fine						Metonymic fine						Component						Nested					
	Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.792	.419	.548	**.641**	.339	.444	**.805**	.28	.415	**.805**	.28	.415	.783	.34	.474	.727	.316	.44	**.333**	**.014**	**.026**	**.333**	**.014**	**.026**
Base	.723	.517	.602	.479	.343	.400	.593	.593	.593	.585	.585	.585	.589	.298	.396	.486	.246	.327	.000	.000	.000	.000	.000	.000
Base + CE	.704	.585	.639	.466	.388	.424	.667	.559	**.608**	.667	.559	**.608**	.589	.432	.498	.506	.371	.428	.250	**.014**	**.026**	.000	.000	.000
Base + CE + FT	.706	.521	.600	.478	.353	.406	.538	**.602**	.568	.538	**.602**	.568	.654	.266	.378	.571	.232	.33	.000	.000	.000	.000	.000	.000
Base + CE + BERT	.773	.693	.731	.348	.312	.329	.000	.000	.000	.000	.000	.000	.562	.222	.318	.382	.151	.216	.000	.000	.000	.000	.000	.000
Base + CE + BERT – newly	.800	.647	.716	.358	.289	.32	.000	.000	.000	.000	.000	.000	.455	.480	.467	.31	.327	.318	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.800	.626	.703	.515	.403	.452	.581	.547	.563	.568	.534	.550	.670	.471	.553	.525	.369	.433	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT – newly	**.816**	.639	.717	.551	.432	.484	.627	.542	.582	.627	.542	.582	.533	.227	.319	.397	.169	.237	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT (single)	.776	.569	.656	.477	.35	.403	.000	.000	.000	.000	.000	.000	**.841**	.423	**.563**	**.751**	.378	**.503**	.000	.000	.000	.000	.000	.000
+ Fine-tuning (unfreezing) BERT																								
Base + CE + BERT	.759	**.703**	.73	.311	.288	.299	.000	.000	.000	.000	.000	.000	.418	.295	.346	.276	.195	.229	.000	.000	.000	.000	.000	.000
Base + CE + BERT – newly	.758	.696	.726	.290	.267	.278	.000	.000	.000	.000	.000	.000	.399	.328	.360	.239	.197	.216	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.736	.687	.711	.433	.405	.418	.524	.551	.537	.508	.534	.521	.474	.508	.490	.338	.362	.349	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT – newly	.801	.685	**.738**	.548	**.469**	**.506**	.691	.475	.563	.691	.475	.563	.58	**.51**	.543	.472	**.415**	.442	.000	.000	.000	.000	.000	.000

(b) fine grained entity type, multi-segment split

Configuration	Literal coarse						Metonymic coarse					
	Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.790	.464	.585	.643	.378	.476	**.814**	.297	.435	**.814**	.297	.435
Base	.678	.552	.609	.519	.422	.465	.571	.581	.576	.567	.576	.571
Base + CE	.688	.573	.626	.548	.456	.498	.618	.576	.596	.618	.576	.596
Base + CE + FT	.706	.548	.617	.549	.426	.480	.725	.492	.586	.725	.492	.586
Base + CE + BERT	.763	.752	.758	.642	.632	.637	.714	.508	.594	.714	.508	.594
Base + CE + BERT – newly	.805	.654	.722	.641	.520	.574	.433	.517	.471	.426	.508	.463
Base + CE + FT + BERT	.767	**.765**	.766	.647	.645	.646	.622	**.627**	**.624**	.622	**.627**	**.624**
Base + CE + FT + BERT – newly	.799	.726	.761	.671	.609	.639	.696	.542	.610	.696	.542	.610
Base + CE + FT + BERT (single)	**.860**	.738	**.795**	**.753**	**.647**	**.696**	.709	.517	.598	.709	.517	.598

(c) coarse grained entity type, document split

Configuration	Literal fine						Metonymic fine						Component						Nested					
	Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict			Fuzzy			Strict		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Baseline	.792	.419	.548	.641	.339	.444	**.805**	.280	.415	**.805**	.280	.415	.783	.340	.474	**.727**	.316	.440	.333	.014	.026	.333	.014	.026
Base	.690	.530	.599	.448	.344	.389	.586	.606	.596	.582	.602	.592	.592	.394	.474	.491	.327	.393	.312	.068	.112	.250	.055	.090
Base + CE	.706	.555	.622	.483	.380	.426	.670	.534	.594	.670	.534	.594	.683	.447	.54	.589	.385	.466	.154	.027	.047	.077	.014	.023
Base + CE + FT	.726	.530	.613	.527	.384	.445	.766	.500	.605	.766	.500	.605	.722	.332	.455	.636	.292	.401	**.500**	**.082**	**.141**	**.500**	**.082**	**.141**
Base + CE + BERT	.782	.734	.757	.571	.536	.553	.750	.508	.606	.750	.508	.606	.700	.500	.583	.623	.445	.520	.333	.027	.051	.333	.027	.051
Base + CE + BERT – newly	.806	.594	.684	.496	.365	.421	.500	.508	.504	.500	.508	.504	.565	.090	.156	.420	.067	.116	.000	.000	.000	.000	.000	.000
Base + CE + FT + BERT	.791	**.763**	.777	.594	.574	.584	.649	**.610**	**.629**	.649	**.610**	**.629**	.703	.582	.637	.585	.485	.53	.250	.014	.026	.250	.014	.026
Base + CE + FT + BERT – newly	**.840**	.679	.751	.615	.497	.550	.744	.517	.610	.744	.517	.610	**.792**	.397	.529	.699	.350	.467	.250	.007	.013	.000	.000	.000
Base + CE + FT + BERT (single)	.839	.743	**.788**	**.669**	**.593**	**.629**	.667	.525	.588	.645	.508	.569	.718	**.588**	**.647**	.632	**.517**	**.569**	.000	.000	.000	.000	.000	.000

(d) fine grained entity type, document split

Table 7
NERC, German. The best result per table and column is given in bold, second best result is underlined

4. Conclusion

We have used a modular architecture to approach two distinct shared tasks relevant for OCRed historical corpora: post-OCR correction (ICDAR2019) and Named Entity Recognition (CLEF-HIPE-2020), using data in English, French and German. Our architecture combines a modular embedding layer with a task-specific layer, allowing us to test the impact of pre-trained language models, such as BERT, and compare them against other set-ups. We find that using pre-trained models has a limited or absent impact on post-OCR correction, while it greatly helps for NER. These results underline how, despite the recent successes of pre-trained language models, their use on historical collections should still be assessed critically according to the data and the task at hand.

The quality and quantity of the available data is a significant limitation of our study, for both tasks and especially for English. Curating and releasing high-quality shared tasks for the community is a particularly important present and future contribution to make, as demonstrated by recent, successful examples including SemEval2020 [50] and CLEF-HIPE-2020 [20]. Another related limitation is given by the choice to focus on only two tasks and three languages. Future work should encompass a broader set of tasks and languages. Lastly, the current offering of language models pre-trained on historical texts is still very limited, its expansion in the future should prove useful.

Limitations notwithstanding, our work might serve to dispel the illusion that plug-and-play transfer learning with pre-trained language models will overcome all the challenges posed by historical corpora. While practically useful in a variety of settings, pre-trained language models also have their limitations. When considering the use of pre-trained language models, several elements should be considered, including: a) does the complexity of the task at hand require them, or would a simpler approach suffice? b) Does the task at hand benefits from modelling a broader linguistic context (e.g., the sentence, the preceding and following sentences) or not? c) How many data are available? Pre-trained language models are particularly helpful in tackling complex tasks, benefiting from modelling a wider linguistic context, and when little or no high-quality annotated data are available.

Data and code availability

Our code base is publicly available and described at https://doi.org/10.5281/zenodo.4033104.

References

[1] A. Akbik, D. Blythe, and R. Vollgraf. "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Aug. 2018, pp. 1638–1649.

[2] A. Akbik et al. "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)*. Association for Computational Linguistics, June 2019, pp. 54–59.

[3] C. Artaud et al. "Find it! Fraud Detection Contest Report". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. Aug. 2018, pp. 13–18. DOI: 10.1109/ICPR.2018.8545428.

[4] D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473 (2015).

[5] J. Baxter. “A Model of Inductive Bias Learning”. In: *Journal of Artificial Intelligence Research* 12 (Mar. 2000), pp. 149–198. ISSN: 1076-9757. DOI: 10.1613/jair.731.

[6] T. Bluche et al. “Preparatory KWS Experiments for Large-Scale Indexing of a Vast Medieval Manuscript Collection in the HIMANIS Project”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. Nov. 2017, pp. 311–316. DOI: 10.1109/ICDAR.2017.59.

[7] P. Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5 (Dec. 2017), pp. 135–146. DOI: 10.1162/tacl_a_00051.

[8] M. Bollmann. “A Large-Scale Comparison of Historical Text Normalization Systems”. In: *Proceedings of the 2019 Conference of the North* (2019). arXiv: 1904.02036, pp. 3885–3898. DOI: 10.18653/v1/N19-1389.

[9] J. Bos et al. “The Groningen Meaning Bank”. In: *Handbook of linguistic annotation*. Springer, 2017, pp. 463–496.

[10] G. Chiron et al. “ICDAR2017 Competition on Post-OCR Text Correction”. In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. Nov. 2017, pp. 1423–1428. DOI: 10.1109/ICDAR.2017.232.

[11] G. Chiron et al. “Impact of OCR Errors on the Use of Digital Libraries: Towards a Better Access to Information”. In: *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. June 2017, pp. 1–4. DOI: 10.1109/JCDL.2017.7991582.

[12] K. Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734.

[13] A. Chronopoulou, C. Baziotis, and A. Potamianos. “An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2089–2095.

[14] R. Collobert and J. Weston. “A unified architecture for natural language processing: deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. ICML ’08. Association for Computing Machinery, July 2008, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177. URL: https://doi.org/10.1145/1390156.1390177.

[15] W. Dai et al. “Self-taught clustering”. In: *Proceedings of the 25th international conference on Machine learning*. ICML ’08. Association for Computing Machinery, July 2008, pp. 200–207. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390182. URL: https://doi.org/10.1145/1390156.1390182.

[16] L. Deng, G. Hinton, and B. Kingsbury. “New types of deep neural network learning for speech recognition and related applications: an overview”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. May 2013, pp. 8599–8603. DOI: 10.1109/ICASSP.2013.6639344.

[17] J. Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[18] M. Ehrmann et al. *Diachronic Evaluation of NER Systems on Old Newspapers*. 2016. URL: https://infoscience.epfl.ch/record/221391.

[19] M. Ehrmann et al. “Language Resources for Historical Newspapers: the Impresso Collection”. In: *Proceedings of The 12th Language Resources and Evaluation Conference*. European Language Resources Association, May 2020, pp. 958–968. ISBN: 979-10-95546-34-4.

[20] M. Ehrmann et al. “Overview of CLEF HIPE 2020: Named Entity Recognition and Linking on Historical Newspapers”. In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Ed. by A. Arampatzis et al. Cham: Springer International Publishing, 2020, pp. 288–310. ISBN: 978-3-030-58219-7.

[21] Ehrmann et al. *HIPE - Shared Task Participation Guidelines*. Jan. 2020. DOI: 10.5281/zenodo.3677171.

[22] A. Ghaddar and P. Langlais. “Robust Lexical Features for Improved Neural Network Named-Entity Recognition”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Aug. 2018, pp. 1896–1907.

[23] R. Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[24] M. Giulianelli, M. Del Tredici, and R. Fernández. “Analysing Lexical Semantic Change with Contextualised Word Representations”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 3960–3973.

[25] S. Gururangan et al. “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks Tasks”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740.

[26] S. van Hooland et al. “Exploring entity recognition and disambiguation for cultural heritage collections”. In: *Literary and Linguistic Computing* 30.2 (June 2015), pp. 262–279. ISSN: 0268-1145. DOI: 10.1093/llc/fqt067.

[27] A. Joulin et al. “Bag of Tricks for Efficient Text Classification”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017, pp. 427–431.

[28] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[29] J.-C. Klie et al. “The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation”. In: *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*. Santa Fe, New Mexico: Association for Computational Linguistics, Aug. 2018, pp. 5–9. (Visited on 05/23/2020).

[30] T. Kudo. “Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 66–75.

[31] K. Labusch, C. Neudecker, and D. Zellhofer. “BERT for Named Entity Recognition in Contemporary and Historic German”. In: *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*. Erlangen, Germany: German Society for Computational Linguistics & Language Technology, 2019, pp. 1–9.

[32] J. Lafferty, A. McCallum, and F. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Departmental Papers (CIS)* (June 2001). URL: https://repository.upenn.edu/cis_papers/159.

[33] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML ’01. Morgan Kaufmann Publishers Inc., June 2001, pp. 282–289. ISBN: 978-1-55860-778-1.

[34] G. Lample et al. “Neural Architectures for Named Entity Recognition”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, June 2016, pp. 260–270. DOI: 10.18653/v1/N16-1030.

[35] V. I. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions and Reversals*. 1966.

[36] E. Linhares Pontes et al. “Impact of OCR Quality on Named Entity Linking”. In: *Digital Libraries at the Crossroads of Digital Information for the Future*. Ed. by A. Jatowt, A. Maeda, and S. Y. Syn. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 102–115. ISBN: 978-3-030-34058-2. DOI: 10.1007/978-3-030-34058-2_11.

[37] I. Loshchilov and F. Hutter. *Fixing Weight Decay Regularization in Adam*. 2018. URL: https://openreview.net/forum?id=rk6qdGgCZ.

[38] S. J. Pan and Q. Yang. “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (Oct. 2010), pp. 1345–1359. ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[39] C. Papadopoulos et al. “The IMPACT dataset of historical document images”. In: *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*. HIP ’13. Association for Computing Machinery, Aug. 2013, pp. 123–130. ISBN: 978-1-4503-2115-0. DOI: 10.1145/2501115.2501130.

[40] S. Pinker. *Learnability and Cognition, new edition: The Acquisition of Argument Structure*. Google-Books-ID: adivAAAAQBAJ. MIT Press, May 2013. ISBN: 978-0-262-31428-2.

[41] M. Piotrowski. “Natural Language Processing for Historical Texts”. In: *Synthesis Lectures on Human Language Technologies* 5.2 (Sept. 2012), pp. 1–157. ISSN: 1947-4040. DOI: 10.2200/S00436ED1V01Y201207HLT017.

[42] R. Raina et al. “Self-taught learning: transfer learning from unlabeled data”. In: *Proceedings of the 24th international conference on Machine learning*. ICML ’07. Association for Computing Machinery, June 2007, pp. 759–766. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273592. URL: https://doi.org/10.1145/1273496.1273592.

[43] B. Ramsundar et al. “Massively Multitask Networks for Drug Discovery”. In: *CoRR* abs/1502.02072 (2015).

[44] L. Rauscher, V.-J. Vos, and H. Verwayen. *Access to Digital Resources of European Heritage*. 2017.

[45] C. Rigaud et al. *ICDAR 2019 Competition on Post-OCR Text Correction*. eng. Sept. 2019. URL: https://zenodo.org/record/3459116#.XkmXfChKiUk (visited on 02/16/2020).

[46] R. Rikowski. *Digitisation Perspectives*. Google-Books-ID: IUNg7dj3Ue0C. Springer Science & Business Media, July 2011. ISBN: 978-94-6091-299-3.

[47] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv:1609.04747 [cs]* (June 2017). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[48] S. Ruder. “Neural Transfer Learning for Natural Language Processing”. en. PhD thesis. National University of Ireland, Galway, Feb. 2019, p. 329.

[49] E. T. K. Sang and F. De Meulder. “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”. In: *Proceedings of CoNLL-2003, Edmonton, Canada*. Morgan Kaufman Publishers. 2003, pp. 142–145.

[50] D. Schlechtweg et al. *SemEval-2020 Task 1: Unsupervised Lexical Semantic Change Detection*. 2020.

[51] M. Schuster and K. Nakajima. “Japanese and Korean voice search”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2012, pp. 5149–5152. DOI: 10.1109/ICASSP.2012.6289079.

[52] P. Shoemark et al. “Room to Glo: A Systematic Comparison of Semantic Change Detection Approaches with Word Embeddings”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 66–76. DOI: 10.18653/v1/D19-1007.

[53] D. A. Smith and R. Cordell. “A Research Agenda for Historical and Multilingual Optical Character Recognition”. In: *NUlab, Northeastern University.@ https://ocr. northeastern. edu/report* (2018), p. 36.

[54] A. Søgaard and Y. Goldberg. “Deep multi-task learning with low level tasks supervised at lower layers”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Aug. 2016, pp. 231–235. DOI: 10.18653/v1/P16-2038.

[55] C. Sporleder. “Natural Language Processing for Cultural Heritage Domains”. In: *Language and Linguistics Compass* 4.9 (2010), pp. 750–768. ISSN: 1749-818X. DOI: 10.1111/j.1749-818X.2010.00230.x.

[56] D. van Strien et al. “Assessing the Impact of OCR Quality on Downstream NLP Tasks:” in: *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*. SCITEPRESS - Science and Technology Publications, 2020, pp. 484–496. ISBN: 978-989-758-395-7. DOI: 10.5220/0009169004840496.

[57] K. Todorov and G. Colavizza. “Transfer Learning for Named Entity Recognition in Historical Corpora”. In: *CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*. Ed. by L. Cappellato et al. CEUR-WS, 2020, to appear.

[58] M. C. Traub, J. van Ossenbruggen, and L. Hardman. “Impact Analysis of OCR Quality on Research Tasks in Digital Archives”. In: *Research and Advanced Technology for Digital Libraries*. Ed. by S. Kapidakis, C. Mazurek, and M. Werla. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 252–263. ISBN: 978-3-319-24592-8. DOI: 10.1007/978-3-319-24592-8_19.

[59] M. Vilain, J. Su, and S. Lubar. “Entity Extraction is a Boring Solved Problem – Or is it?” In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, Apr. 2007, pp. 181–184.

[60] D. Wang and T. F. Zheng. “Transfer learning for speech and language processing”. In: *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. Dec. 2015, pp. 1225–1237. DOI: 10.1109/APSIPA.2015.7415532.

[61] T. Wolf et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *arXiv:1910.03771 [cs]* (Feb. 2020). arXiv: 1910.03771. URL: http://arxiv.org/abs/1910.03771.

Appendix A Post-OCR Correction

A.1 Data

More details on the data split are given in Table 8. German and French are further split into different sources depending on where the data is from. The corresponding ground truth for these languages comes from initiatives such as HIMANIS [6], IMPACT [39], IMPRESSO [19] and RECEIPT [3].

Language	Subset	File count	Characters count		
			Total	Trainset	Testset
German	DE1	102	575 416	460 333	115 083
	DE2	200	494 328	309 973	106 862
	DE3	7623	10 018 258	8 014 606	2 003 652
	DE4	321	509 757	407 806	101 951
	DE5	654	818 711	654 969	163 742
	DE6	773	935 014	748 011	187 003
	DE7	415	527 845	422 276	105 569
English	EN1	200	243 107	189 085	52 112
French	FR1	1172	2 792 067	2 233 654	558 413
	FR2	200	227039	170 569	52 989
	FR3	1968	742 574	594 059	148 515

Table 8
ICDAR 2019 data split.

For each document, there are three versions available: (i) `OCR_toInput` which is the OCRed text, (ii) `OCR_aligned` which is the OCRed text aligned to the ground truth character by character. Finally, (iii) `GS_aligned` contains the ground truth. An example from such a structure is provided in Table 9.

Version	Text
`OCR_toInput`	`sail sliort of that than they who`
`OCR_aligned`	`sail sliort of that than they who@@@@@@@`
`GS_aligned`	`fall s@hort of that than they who aspire`

Table 9
ICDAR 2019 data sample.

ICDAR2017's data come from several digital collections, including the National Library of France (BnF) and the British Library (BL). The ground truth tokens come from BnF's internal projects and other initiatives such as Project Gutenberg[1], Europoeana Newspapers, IMPACT and Wikisource. OverProof data[2] combines publications of historical newspapers coming from the National Library of Australia's Trove newspaper archive[3] with randomly selected articles

[1] https://www.gutenberg.org.
[2] https://overproof.projectcomputing.com.
[3] https://trove.nla.gov.au.

in the Library of Congress Chronicling America[1]. The service uses ABBYY's FineReader OCR texts published from 1842 to 1954.

A.2 Model fitting

In order to fine-tune the architecture and assess the impact of pre-trained embeddings, we consider an extensive list of hyper-parameters for testing. The full list is shown in Table 10. The configuration that we use to report results is summarised in Table 11.

	Parameter name	Value options
Model hyper-parameters	use of newly trained encoder character embeddings - newly trained encoder character embedding layer size - newly trained encoder character embedding layer dropout	yes/no 32/64/128 0/0.2/0.5/0.8
	newly trained decoder character embedding layer size newly trained decoder character embedding layer dropout	32/64/128 0/0.2/0.5/0.8
	share encoder and decoder embedding layers	yes/no
	use of pre-trained (FastText) embeddings	yes/no
	use of pre-trained (BERT) embeddings - weights usage type - fine tune type - fine tune BERT learning rate	yes/no fine tune/freeze from beginning/after initial convergence same as global learning rate/1e−3/1e−4
	Encoder GRU options - hidden size - dropout - directionality - number of layers	 128/256/512/1024 0/0.2/0.5/0.8 bi-directional/uni-directional 1/2/3
	Decoder GRU options - hidden size - dropout - number of layers	 128/256/512/1024 0/0.2/0.5/0.8 1/2/3
Training	Optimizer - learning rate	SGD/Adam/AdamW 1e−2/1e−3/1e−4/1e−5

Table 10
Post-OCR correction hyper-parameters.

We have two configuration sets related to *newly trained embeddings* due to having two embedding layers – for the encoder and the decoder respectively. There is a special case in which we only use one and we *share it* between the encoder and decoder. When used in the decoder, the embedding layer does not consider pre-trained information since it works at the character level.

If *BERT embeddings* are included in our configuration, we face similar decisions as mentioned before, and we assess whether freezing the weights hinders performance compared to fine-tuning them further. If we choose the latter, we additionally test two ways of fine-tuning: from the beginning or, alternatively, after the encoder-decoder model has converged. Our global learning rate is generally similar to the one used for fine-tuning BERT, but we keep the option to configure those two separately and use a different one for the pre-trained partition.

[1] https://chroniclingamerica.loc.gov.

Hyper-parameters	Value
GRU encoder hidden size	512
GRU directionality	bi-directional
GRU encoder dropout	0.5
GRU encoder number of layers	2
GRU decoder hidden size	512
GRU decoder dropout	0.5
GRU decoder number of layers	2
Share embedding layer	yes
Newly trained embeddings size	128
Newly trained dropout	0.5
Optimizer	AdamW
Learning rate	1e−4
Fine-tune learning rate	1e−4

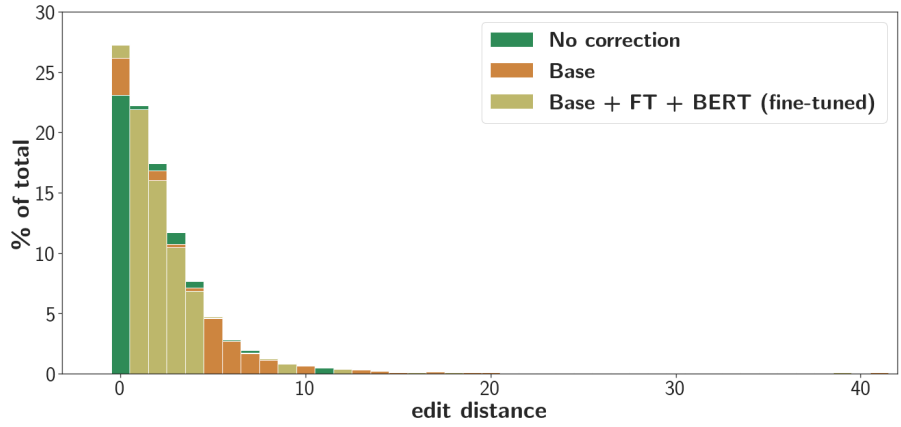
Table 11
Post-OCR correction hyper-parameter configuration.

Due to using an encoder-decoder architecture, we have two distinct configuration sets for the RNNs taking part in those. We assess a variety of hyper-parameters including: *hidden size*, *dropout*, *directionality* and *number of layers* for each GRU. For the decoder, we keep a uni-directional setup due to the specifics of working with a single character and therefore not having any benefits from using two directions.

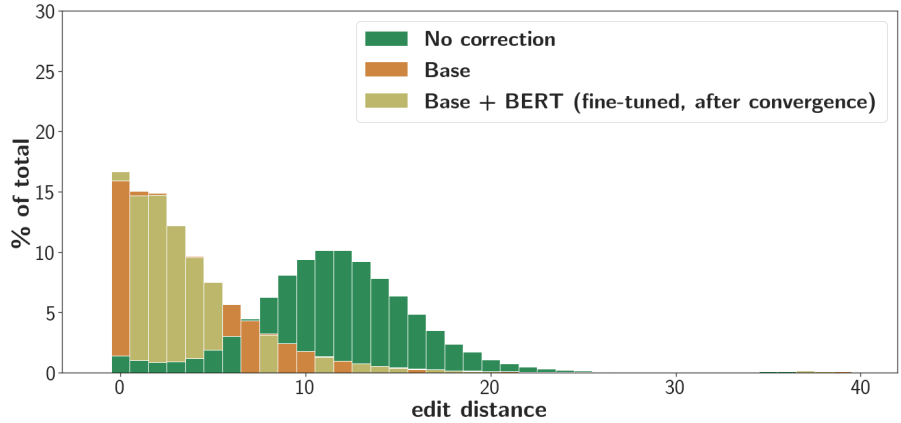
Finally, we assess a variety of optimisers including SGD, Adam and AdamW [47, 28, 37], using AdamW in our reported results. We test out different learning rate values, keep the momentum of SGD optimiser to its default value of 0 and set a weight decay of 1e−8 for all optimisers. We also report the average convergence time of different models during training in Table 12. As it can be seen, adding BERT entails a slower convergence speed, in particular if BERT if fine-tuned.

Configuration	English	French	German
Base	475.882	662.735	1560.595
Base + FT	475.494	735.93	1481.484
Base + BERT	710.549	1070.933	1593.157
Base + FT + BERT	659.277	1141.16	1873.405
+ Fine-tuning (unfreezing, from start) BERT			
Base + BERT	1218.266	1292.461	3153.515
Base + FT + BERT	1300.52	1529.498	3064.282
+ Fine-tuning (unfreezing, after initial convergence) BERT			
Base + BERT	990.149	1200.414	2639.135
Base + FT + BERT	1087.442	1570.694	2393.629

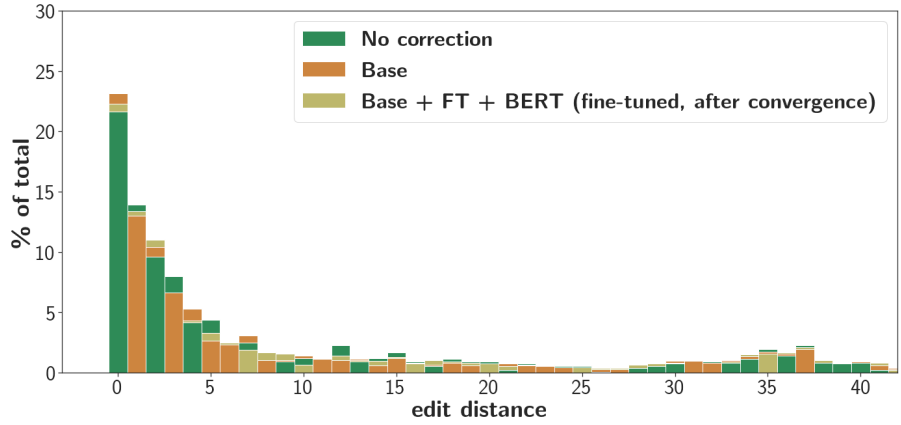
Table 12
Post-OCR convergence speed (averaged, in minutes).



(a) French.



(b) German.



(c) English.

Figure 5: Levenshtein edit distance distribution comparing the raw OCR texts, the Base model and the best model for each language. The bars are ordered, for each bin, from the smallest to the largest value, in order to show them all.

Appendix B Named Entity Recognition

B.1 Data

For each newspaper, articles were randomly sampled in order to (i) belong to the first years of a set of predefined decades covering the life-span of the newspaper, and (ii) have a title, have more than 50 characters, and belong to any page (no restriction to front pages only). For each decade, the set of selected articles was additionally manually triaged in order to filter-out non-journalistic contents such as ads. The time span of the whole dataset goes from 1790 until 2010 decades and the OCR quality varies according to time and archival material. Information about the amount of tokens per decade and per language is shown in Figure 6. Additionally, the amount of mentions, i.e., entity occurrences per decade are also shown in Figure 7. They are broken down per coarse-grained type.

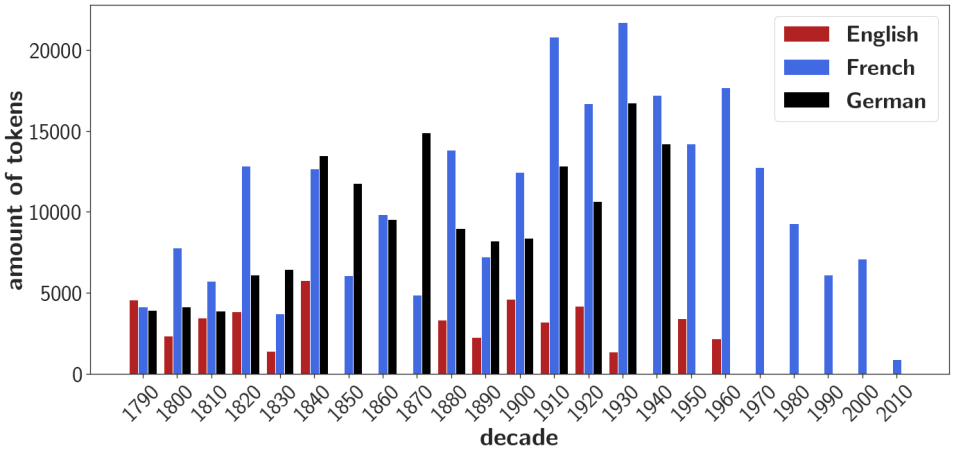


Figure 6: Amount of tokens per decade and language.

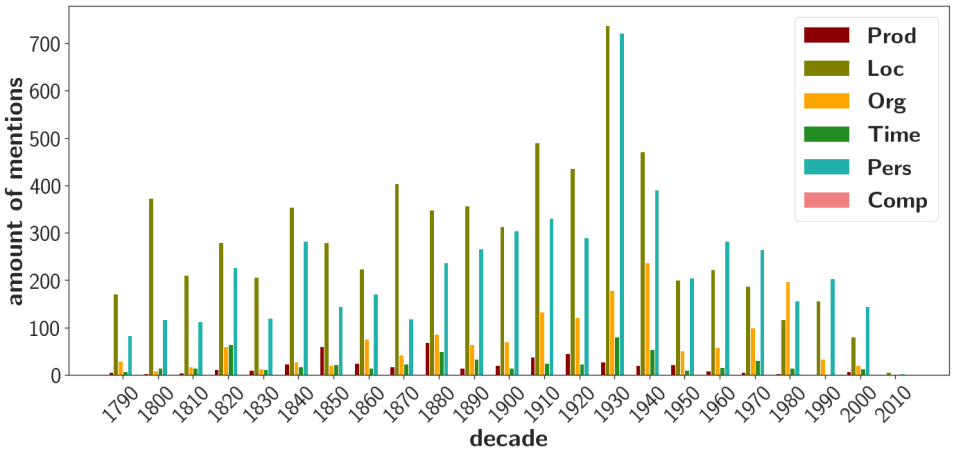


Figure 7: Amount of tag mentions per decade and tag.

Another important characteristic of the data is the abundance of non-entity tokens. More specifically, percentage-wise tokens which are labelled as "Other" or `O` make up a considerable

amount of the labels. We show the proportions of non-entity tokens for the training datasets in Figure 8.

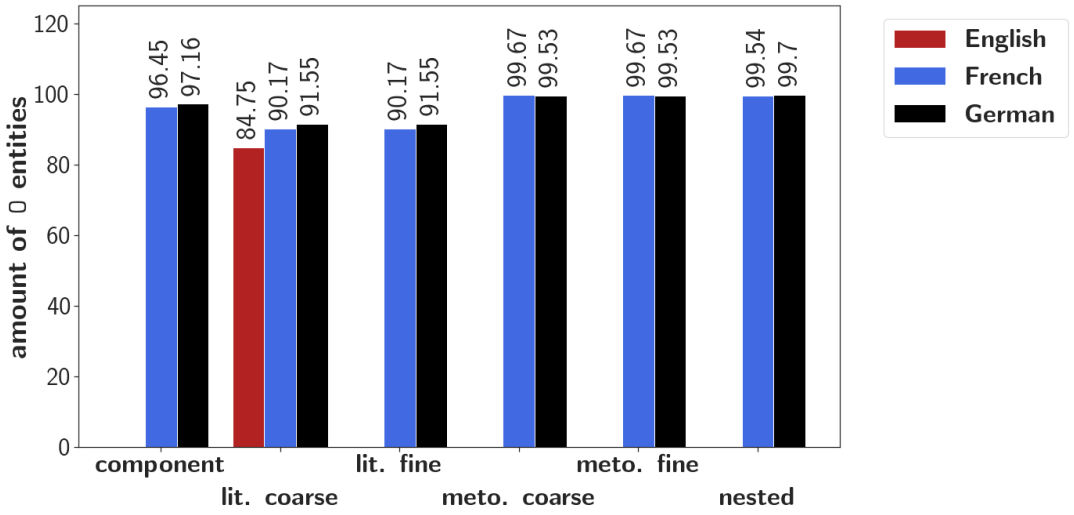


Figure 8: Non-entity tokens per language in the training datasets.

The annotation was made by native speakers using the INCEpTION annotation platform [29]. Annotators were first trained on a 'mini-reference' corpus consisting of 10 articles per language, in order to ensure their understanding of the guidelines. Additionally, some items of the test set, as well as randomly sampled items among the training and development sets, were double-annotated and adjudicated.

To showcase all entity types and compare their properties, we visualise them in Table 13. The data is released in IOB format (inside-outside-beginning format)[1] which we also use during training and evaluating. This format is derived in a similar fashion, following the CoNLL-U format[2]

Full details and specifications about the dataset can be found in the original CLEF-HIPE-2020 challenge participation guidelines [21].

B.2 Model fitting

We investigate how different configurations, embedding combinations and single- to multi-task transition affects the performance of our model and of its components. To this end, we consider an extensive set of hyper-parameters as listed in Table 14. The configurations we use to report results are summarised in Table 15.

Pre-processing The input data is organised into *documents*, and each document is split into multiple *segments* where usually one segment corresponds to one line in the original historical source. The input can thus be split into segments or into documents. Using segments leads to much faster convergence, while document splitting usually yields better results in our experiments. We further analyse the importance of splitting by introducing a *multi-segment* option which combines more than one consecutive segment. We pick the maximum length of one

[1] https://en.wikipedia.org/wiki/Inside–outside–beginning_(tagging)
[2] https://universaldependencies.org/format.html.

Coarse-grained tag set	Fine-grained tag set	Metonymy applies	Entity nesting applies
pers	pers.ind pers.coll pers.ind.articleauthor	yes	yes
org	org.adm org.ent org.ent.pressagency	yes	yes
prod	prod.media prod.doctr	yes	no
time	time.date.abs	no	no
loc	loc.add.elec loc.add.phys loc.adm.town loc.adm.reg loc.adm.nat loc.adm.sup loc.phys.geo loc.phys.hydro loc.phys.astro loc.oro loc.fac	yes	yes
	loc.unk	no	no

Table 13
NERC entity types comparison

multi-segment sequence to be the maximum length allowed by the HuggingFace Transformers library. We do this purely for simplicity reasons and to avoid any unwanted noise. At document level we overcome this limitation by splitting documents using a sliding window approach where the first and last 5 tokens for each split are overlapping with the previous and next splits respectively. We perform the cutting before extracting features through BERT after which we concatenate the representations back. We take the average values for the overlapping tokens. Finally, we replace all numbers with zeros, including such that contain more than one digit. Besides, we do not lowercase, nor do we remove any punctuation or other characters.

Fine-tuning vs. freezing There are two possibilities when using pre-trained embeddings: keep them frozen or fine-tuning. Fine-tuning the model lets us introduce two additional configuration options. The first one is related to when to start fine-tuning. This is most often performed at the beginning and until convergence. However, previous studies have shown [25] that fine-tuning from the start might lead the model away from its main objective and thus that the full model should converge first, with frozen pre-trained weights. After convergence, the pre-trained weights are fine-tuned. This is something that we also investigate but find no difference between the two approaches. We therefore fine-tune from the start in the reported experiments with fine-tuning enabled.

Manually crafted features Following previous work [22], we assess the importance of manually crafted features. We use `AllLower`, `AllUpper`, `IsTitle`, `IsNumeric`, `FirstLetterUpper`, `FirstLetterNotUpper` and `NoAlphaNumeric` as extra morphological features. When including these features in the model, we do not get significant improvements.

	Parameter name	Value options

Model hyper-parameters	amount of tag types simultaneously trained	one of 6 tags;
		coarse(literal and metonymic);
		fine(literal, metonymic, nested and component);
		all 6 tags;

	replace numbers during pre-processing	yes/no
	sequences split type	segment/multi-segment/document
	use of character embeddings - characters embedding layer size - characters RNN hidden size	yes/no 16/32/64/128 16/32/64/128/256
	use of newly trained sub-word embeddings - newly trained sub-word embedding layer size - newly trained sub-word embedding layer dropout	yes/no 16/32/64/128 0/0.2/0.5/0.8
	use of pre-trained (FastText) embeddings	yes/no

	use of pre-trained (BERT) embeddings
	- weights usage type
	  - fine tune type
	  - fine tune BERT learning rate
		yes/no
		fine tune/freeze
		from beginning/after initial convergence
		same as global learning rate/1e−3/1e−4

	LSTM options - hidden size - dropout - directionality - number of layers	 128/256/512 0/0.2/0.5/0.8 bi-directional/uni-directional 1/2
	use of manually crafted features	yes/no
	use of weighted loss	yes/no
Training	Optimizer - learning rate	SGD/Adam/AdamW 1e−2/1e−3/1e−4

Table 14
NER hyper-parameters

Weighting As it is common with NERC tasks, most of the ground truth is composed of *outside* or `O` tags. In our case, these make up for approximately 94.92%, 95.95%, and 96.5% of the total tokens for English, French and German languages respectively. To counteract tag imbalance, we test a *weighted loss* which we plug into the CRF layer, giving more weight to tags predicted as outside ones but are in fact part of entities, and less on tokens which are predicted as inside an entity but are actually outside. This weighted loss does not prove to be beneficial.

Other hyper-parameters We assess a variety of optimizers including SGD, Adam and AdamW [47, 28, 37]. For the *learning rate* we see that higher values benefit the model more. When using SGD, lower values tend to produce better results. We use a default value of 0 for *momentum* – when using SGD – and pick a similar default value of 1e−8 for *weight decay* for all optimizers.

We observe a similar trend as for post-OCR correction in terms of converge speed when using pre-trained information. Additionally, we investigate the speed when comparing a multi-task versus a single-task setting. As expected, multi-task training is slower compared to a single-task approach. Even more so, when using multi-segment split type, running six single-task

Hyper-parameters	Configuration I	Configuration II	Configuration III
RNN hidden size	512	256	512
RNN directionality	bi-directional	bi-directional	bi-directional
RNN dropout	0.5/0.8	0.5/0.8	0.5/0.8
Newly trained embeddings size	64	64	64
Character embeddings size	–	16	16
Character embeddings RNN hidden size	–	32	32
Replace numbers during pre-processing	yes	yes	yes
Weighted loss usage	no	no	no
Optimizer	AdamW	AdamW	AdamW
Learning rate	1e−2	1e−2	1e−2
Fine-tune learning rate	1e−4	1e−4	1e−4

Table 15
NER hyper-parameter configurations.
Configuration I is used for `Base`.
Configuration II is used for `Base + CE + BERT` *and* `Base + CE + BERT - newly.`
Configuration III is used for all remaining setups.

runs for each tag type takes about the same time as one multi-task but yields better results. Exact numbers are reported in Table 16.

Configuration	Time(minutes)	Time(hours)
Multi-task (document)	347.1	5.78
Single-task (document)	144.15	2.4
Multi-task (multi-segment)	162.96	2.72
Single-task (multi-segment)	26.86	0.45

Table 16
NER convergence speed (averaged).