FETD^2: A Framework for Enabling Textual Data Denoising via Robust Contextual Embeddings

Govind, Céline Alec, Jean-Luc Manguin, and Marc Spaniol(✉)

Department of Computer Science, Université de Caen Normandie,
Campus Côte de Nacre, 14032 Caen Cedex, France
{govind,celine.alec,jean-luc.manguin,marc.spaniol}@unicaen.fr

Abstract. Efforts by national libraries, institutions, and (inter-) national projects have led to an increased effort in preserving textual contents - including non-digitally born data - for future generations. These activities have resulted in novel initiatives in preserving the cultural heritage by digitization. However, a systematic approach toward Textual Data Denoising (TD^2) is still in its infancy and commonly limited to a primarily dominant language (mostly English). However, digital preservation requires a universal approach. To this end, we introduce a "Framework for Enabling Textual Data Denoising via robust contextual embeddings" (FETD^2). FETD^2 improves data quality by training language-specific data denoising models based on a small number of language-specific training data. Our approach employs a bi-directional language modeling in order to produce noise-resilient deep contextualized embeddings. In experiments we show the superiority compared with the state-of-the-art.

Keywords: Textual Data Denoising · AI · Contextual representations

1 Introduction

1.1 Motivation and Problem

In recent years, natural language processing (NLP) has seen major improvements by the application of machine learning ranging from "low-level" text (pre-) processing up to "high-level" semantic enrichment. Each component is an important asset ensuring data quality along the entire value chain, e.g., when preserving the cultural heritage by digitization of textual documents. Despite all recent achievements in document digitization, the overall process is still in its infancy. In particular, studies have been primarily conducted on English language text. While the approaches are - in general - conceptually transferable to other languages, there are several drawbacks to be considered. First, models of contextual as well as non-contextual word representations have been predominantly developed for English language text. Second, these models provide dense representation for the vocabulary tokens but broadly make an assumption that tasks in NLP do not have to deal with noisy textual data, which is more prevalent in real-world or digitized documents. Last, but not least, adapting data (pre-) processing for

G. Berget et al. (Eds.): TPDL 2021, LNCS 12866, pp. 3–16, 2021.
https://doi.org/10.1007/978-3-030-86324-1_1

less commonly used languages requires a generalized approach in order to overcome performance drain caused by data scarcity.

1.2 Approach and Contribution

In this paper, we present FETD^2: "a Framework for Enabling Textual Data Denoising". FETD^2 employs a noise-resilient deep contextualized embedding model based on bi-directional language modeling. Further, language adaptability is supported by emulating language-specific patterns of spelling-errors (referred to as "Confusion Matrix++") by systematically injecting them into high-quality contents for training the model. Finally, in extensive intrinsic and extrinsic evaluations we demonstrate the superiority of FETD^2.

2 Related Work

2.1 Pre-trained Language Representation Models

Pre-trained language representation models aim at encoding the syntactic and semantic knowledge about tokens by building their continuous representations in a high dimensional space. Popular context-insensitive models such as Word2Vec [18] and GloVe [20] learn fixed embeddings of words based on their co-occurrence in large corpora. Although, these models are prone to out-of-vocabulary (OOV) problem caused by spelling errors/variations as well as fail to build unique representations of homonyms. To address this, the idea of deep contextualized representation aims at producing separate representations of a token when used in different contexts. ELMo [21] uses multiple stacked Long Short-Term Memory (LSTM) [9] layers to produce a high-level contextual representation of words with respect to their use in a sentence. BERT [6] proposes the use of masked language modeling with Transformers [26] to process the natural language text in a truly bidirectional way. ELMo and BERT work on sub-word level and thus have the ability to produce representations for misspelled/OOV tokens. Although, it has been widely reported that the model performance suffers significantly as representations of these noisy tokens degrade in quality [12,24]. Recently, there have been several advancements in LMs via transformers such as XLNet [29], ALBERT [13], CharacterBERT [3] and ELECTRA [5]. Even these approaches have not adequately focused on dealing with noisy data and the resilient language representation needs further exploration.

2.2 Handling Noise in Textual Data

Real world text often contains noise of various nature such as misspellings, optical character recognition (OCR) errors, typographical errors, etc. Models trained on clean data are prone to fail already in the presence of little noise, although being relatively easily decodeable for a human [2,23]. Multiple studies [12,24,28] have reported that pre-trained language representation models (e.g. BERT, ELMo) suffer in domains such as social media and noisy data in general. Further, [15] show the effects of OCR noise on named entity linking resulting in a considerable drop in performance. Persuaded by

the ubiquity of noisy text, there have been growing interest in building robust word representations recently. [7] propose a fastText based model to learn robust embeddings for misspellings by mapping the representation of misspelled words closer to their respective original words in the embedding space. On the other hand, [17] have introduced a context-informed embeddings model based on RNNs. In [1] embedding subspaces are exploited to learn word representation in scarce and noisy data, whereas [25] use a masked language model for denoising. Also, [16] introduce the use of noise-contrastive estimation to improve language modeling and a data augmentation framework Telephonetic to deal with misspellings is introduced by [14]. Adversarial machine learning has also received growing attention in the NLP domain recently [8,27], such as training with adversarial examples in order to build more robust machine learning models [22,24,30]. It has been noted in aforementioned works that having robust word embeddings in-turn helps in building robust models for multiple downstream NLP tasks. To the best of our knowledge, none of the prior approaches tackle data denoising by systematically building a deep contextualized bi-directional language model that is implicitly noise-resilient.

3 Noise-Resilient Contextual Representations

3.1 Bi-Directional Language Models for Contextual Representation

The task of language modeling aims at assigning probability values to future tokens given a history of previous tokens. To this end, a forward language model can be utilized to compute the probability of observing a sequence of tokens as formalized in Eq. 1 and similarly a backward language model in the backward direction as can be seen in Eq. 2. Given a sequence of tokens $t_1, t_2, \ldots, t_N$, a bi-directional language model tries to jointly maximize the log-likelihood in the forward as well as in the backward direction as formalized in Eq. 3.

$$p(t_1, t_2, \ldots, t_N) = \prod_{i=1}^{N} p(t_i | t_1, t_2, \ldots, t_{i-1}) \tag{1}$$

$$p(t_1, t_2, \ldots, t_N) = \prod_{i=1}^{N} p(t_i | t_{i+1}, t_{i+2}, \ldots, t_N) \tag{2}$$

$$\sum_{i=1}^{N} \Big(log\big(p(t_i | t_1, t_2, \ldots, t_{i-1}; \overrightarrow{\theta})\big) + log\big(p(t_i | t_{i+1}, t_{i+2}, \ldots, t_N; \overleftarrow{\theta})\big) \Big) \tag{3}$$

3.2 Noise-Resilient Bi-Directional Language Modeling

In order to make language modeling and the underlying token representation robust towards noise, we propose a novel language modeling objective where the tokens' history is perturbed via realistic noise patterns. To achieve this, we introduce a noise function Γ, which imparts noise in the sequence of history tokens in a controlled manner as seen in Eq. 4 (cf. Sect. 4.1 for details on the noise generation algorithm). Given a sequence of tokens $T = (t_1, t_2, \ldots, t_j)$, the noise function Γ produces a noisy sequence

$\widetilde{T} = (\widetilde{t_1}, \widetilde{t_2}, \ldots, \widetilde{t_j})$ with ratio of noisy tokens controlled by the parameter η. To this end, the new forward and backward language models assign probability to future tokens based on noisy histories (cf. Eqs. 5 and 6). Now, the bi-directional language model tries to optimize the joint log-likelihood in the forward and backward direction given the noise in history tokens. The intuition behind preserving the noisy tokens as a history is that the model will have to learn the representation of noisy tokens close to their original versions in order to improve the correct prediction of future tokens.

$$\widetilde{T} = \Gamma(T, \eta) \tag{4}$$

$$p(t_1, t_2, \ldots, t_N) = \prod_{i=1}^{N} p(t_i | \widetilde{t_1}, \widetilde{t_2}, \ldots, \widetilde{t_{i-1}}) \tag{5}$$

$$p(t_1, t_2, \ldots, t_N) = \prod_{i=1}^{N} p(t_i | \widetilde{t_{i+1}}, \widetilde{t_{i+2}}, \ldots, \widetilde{t_N}) \tag{6}$$

$$\sum_{i=1}^{N} \Big(log\big(p(t_i | \widetilde{t_1}, \widetilde{t_2}, \ldots, \widetilde{t_{i-1}}; \overrightarrow{\theta})\big) + log\big(p(t_i | \widetilde{t_{i+1}}, \widetilde{t_{i+2}}, \ldots, \widetilde{t_N}; \overleftarrow{\theta})\big)\Big) \tag{7}$$

Note that this language modeling objective is not simply building a LM on noisy data. Here, the model has to predict correct future token given the noise in previous tokens. In this way, the model does not intend to generate the noisy text but to recover from the noise in tokens as it aims to always predict the next correct token. In contrast, if a LM is trained on noisy text in a brute fashion then it will aim to generate the noisy text where the goal of predicting noisy future tokens will inadvertently penalize the learning of quality representations. To this end, the noise-resilient LM objective implicitly encodes the task of being robust towards noise in tokens and map them closer to their original version in embedding space (empirically evaluated in experiments, cf. Sect. 5).

3.3 Character-Aware Word Representation

Traditionally, a fixed vocabulary of tokens has been often employed for the language modeling without any sub-word level knowledge to build token representations. This might lead to vocabulary explosion because of noise and thus, a character-aware processing is required. Large-scale transformer-based language models such as BERT have popularized the Byte-Pair Encoding (BPE) tokenization of words into smaller sub-word units. On the other hand, models like ELMo use convolutional neural network based kernels in order to extract character level features of a token. Sub-word units based tokenization itself can be sensitive to the noise in tokens [3,19], which adds further complexity in building a robust LM. In the scope of this paper, we consider a simple yet effective setup to process words as a whole on char-level. We employ an "ELMo-like" character level processing on individual words aiming at capturing sub-word features.

3.4 Language Model Architecture

Concerning the underlying architecture, we employ a bi-directional language model architecture adopted and modified from the ELMo architecture proposed by

Peters et al. [21]. The ELMo model extends the ideas from [10] and [11] by introducing a residual connection between the LSTM layers and building bi-directional language model. In broad specification, the model has two bi-directional LSTM layers with 4096 units and the residual connection. A character level input processing module is utilized with 2048 character n-grams convolutional filters and 2 highway layers. It processes tokens in the input sequence on character level and aims to extract the low-level syntactical features for each of the tokens independently. We modify the model in order to adjust it to the requirements of our noise-resilient language modeling objective. To this end, we integrate a noise generation module at the char-level processing layer (Layer 0 in ELMo terminology) to impart desired level of noise in history tokens during the pre-training process. We discuss the noise generation mechanism in the subsequent section.

4 Noisy Data Generation

4.1 Noise Generation Model

Different noise occurs in texts, e.g. misspellings or erroneous OCR. We focus on noise induced by the OCR process (but is also applicable to other kind of orthographic noise). In order to synthetically generate OCR-inspired noisy text, we introduce a noise generation model in Algorithm 1 based on four parameters: input token sequence T, desired noise ratio η, the number of transformations per token K, and our transformation matrix called Confusion Matrix++ (the core of our noise generation model). In our experiments, we keep $K = 1$ constant whereas varying η over several values between 0 and 0.99.

Algorithm 1. Noise Generation Model (Γ)

Input: Token sequence T; Noise ratio η; Confusion Matrix++ $\mathcal{M}$; # transformations per token K
Output: Noisy tokens sequence $\widetilde{T}$

$seq\_len \leftarrow len(T)$
$noise\_indices \leftarrow random\_sample(T, \lceil seq\_len * \eta \rceil)$
for each index i, token t in T **do**
  if $noise\_indices$ contains i **then**
    // get character unigrams and bigrams
    $chars\_unigrams \leftarrow unigrams(t)$
    $chars\_bigrams \leftarrow bigrams(t)$
    $ngrams = chars\_unigrams + chars\_bigrams$
    // filter non-existing noise transformation
    initialize $transforms$
    for each ngram ng in $ngrams$ **do**
      if $\mathcal{M}$ contains ng **then**
        insert ng into $transforms$
    // sample K transformations randomly
    $transforms \leftarrow random\_sample(transforms, K)$
    $n\_token \leftarrow apply\_transforms(t, transforms, \mathcal{M})$
    insert n_token into $\widetilde{T}$
  else
    insert $token$ into $\widetilde{T}$

4.2 Confusion Matrix++

In order to overcome data-scarcity we introduce the Confusion Matrix++. It differs from a standard confusion matrix as it goes beyond unigram transformations and aims at generating synthetically OCR inspired noise. To this end, we construct it from manually corrected ground truth pairs by obtaining the probability values of different possible erroneous character n-gram transformations (i.e. substitution, deletion, insertion) as well as no error. We consider unigrams and bigrams (which are extremely sparse)

by extracting the statistics from the ICDAR2017 Competition on Post-OCR Text Correction [4] ground truth data in English and French (cf. Table 1 for examples). The Confusion Matrix++ is then used to inject the OCR inspired noise in any presumably clean text.

Table 1. Noisy text generated corresponding to clean sentences for English and French

Language	Noise (η)	Text
English	0	Europe had been hit by the virus shortly before the Americas , although recently some countries are beginning to announce more positive steps .
English	0.25	Europe had been hit b? the virus shorly before the Americas , altbough recently some countries ate beginning to announce more positive stes .
French	0	Antoine Meillet devait diriger la thèse de Jean Paulhan sur la sémantique de le proverbe et c' est lui qui découvrit Gustave Guillaume .
French	0.25	Antoine Meillet devail diriger la thêse de Jean Paulhan sur la semantique de le proverbe et cf est lui qut découvrit Gustave Cuillaume .

5 Experimental Evaluation

5.1 Experimental Datasets

Language-Specific Datasets

In order to demonstrate the language adaptability of the FETD^2 framework, we perform extensive experiments on English and French by employing Wikipedia dumps[1] (February 2020 for English and March 2019 for French). We train the models on around 5 million sentences from the English dataset by randomly sampling 2M paragraphs and 2.5M paragraphs for French in order to keep the same ratio. As a result, we obtain 121M tokens for English and 130M tokens for French. We put aside 10% data for model testing while keeping a separate validation set. The intrinsic evaluation is performed on 534K sentences for English and on 540K sentences for French. Further, we construct the vocabulary by discarding single occurring tokens. As a result, the English and French dataset vocabulary contain 735K and 759K tokens.

Document Classification Datasets

We perform an extrinsic evaluation on the task of document classification. For that purpose, we use the 20 Newsgroups[2] dataset for English, which contains of 18K documents categorized in 20 classes. We perform a random split of 80:10:10 (training, validation, test) by keeping the same percentage of articles from individual categories in each split. For French, we crawled the L'Express[3] newspaper and created a dataset of 2,207 news articles annotated with five different categories maintaining the same split ratio as before.

[1] Wikipedia Dumps https://dumps.wikimedia.org/.
[2] 20 Newsgroups dataset http://qwone.com/~jason/20Newsgroups/.
[3] L'Express https://www.lexpress.fr/.

5.2 Model Configurations

Assessing the sensitivity of FETD^2, we train three different models. They differ in the strategy of importing noise and the amount of noise while performing the training. Moreover, demonstrating the applicability across different languages, we train these models for English and French. To this end, we use two models with noise ratio of $\eta = 0.10$ and $\eta = 0.50$ in all of the training sentences, denoted by FETD$^2(0.1)$ and FETD$^2(0.5)$. Further, we introduce an additional model, namely FETD$^2(0.1H)$, which is trained with noise ratio of $\eta = 0.10$, but only in a random selection of half of the training sentences while keeping the remaining half clean. For the sake of providing an extensive comparative analysis of our models, we also train an original ELMo based model, simply denoted by ELMo. All of the models are trained for 10 epochs and implemented in TensorFlow[4] by extending the open source bi-directional LM library bilm-tf[5] by AllenAI. FETD^2 pre-trained models and evaluation data are publicly available here[6].

In addition to evaluating the models performance on clean texts, we perform experiments at various level of synthetic noise (ranging from minor to extreme), specifically at 0.01, 0.05, 0.10, 0.40, 0.55, 0.70, 0.85, and 0.99. Smaller intervals near the initial boundary value have been chosen in order to analyze the sensitivity on minor perturbations already. At the same time, we analyze the models' sensitivity towards higher level of noise stretching to the extent when almost all the tokens contain noise in some form.

5.3 Noise Sensitivity Study

Embeddings Divergence

We assess the divergence of embedding vectors with respect to the noise imparted in tokens of the input sequence by using cosine similarity (cf. Eq. 8) and Euclidean distance (cf. 9). We report the results in Tables 2 and 3. The scores are aggregated by utilizing micro-averaging as well as macro-averaging. The test sets contain around 500,000 sentences for each language. The original and noisy versions of individual sentence are passed through the concerned model in order to obtain the contextual embeddings

Table 2. Results on model noise sensitivity evaluation at Layer 2 for English

Models	ELMo				FETD$^2(0.1H)$				FETD$^2(0.1)$				FETD$^2(0.5)$			
Test Noise (η)	Micro-Avg		Macro-Avg		Micro-Avg		Macro-Avg		Micro-Avg		Macro-Avg		Micro-Avg		Macro-Avg	
	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean
0.01	0.967	7.189	0.948	9.304	0.992	3.017	0.985	4.326	0.994	2.689	0.987	3.914	**0.997**	**1.795**	**0.992**	**2.692**
0.05	0.948	10.089	0.934	11.223	0.988	4.314	0.982	5.190	0.991	3.831	0.985	4.699	**0.995**	**2.571**	**0.991**	**3.228**
0.10	0.911	14.219	0.901	14.870	0.980	6.216	0.975	6.885	0.984	5.519	0.979	6.208	**0.991**	**3.731**	**0.988**	**4.265**
0.25	0.800	23.018	0.798	22.863	0.957	10.388	0.952	10.726	0.965	9.236	0.961	9.632	**0.981**	**6.314**	**0.978**	**6.640**
0.40	0.681	29.763	0.688	29.135	0.932	13.680	0.927	13.886	0.945	12.195	0.941	12.471	**0.971**	**8.381**	**0.968**	**8.628**
0.55	0.558	35.526	0.569	34.782	0.905	16.512	0.899	16.826	0.924	14.745	0.918	15.114	**0.960**	**10.163**	**0.955**	**10.499**
0.70	0.441	40.464	0.463	39.302	0.879	18.941	0.873	19.135	0.903	16.948	0.897	17.208	**0.949**	**11.691**	**0.945**	**11.966**
0.85	0.332	45.023	0.363	43.461	0.850	21.197	0.846	21.263	0.880	19.002	0.875	19.145	**0.938**	**13.096**	**0.934**	**13.294**
0.99	0.251	48.566	0.292	46.497	0.825	23.011	0.824	22.832	0.860	20.655	0.858	20.577	**0.928**	**14.211**	**0.925**	**14.268**

[4] TensorFlow https://www.tensorflow.org/.

[5] AllenAI bilm-tf https://github.com/allenai/bilm-tf.

[6] FETD^2 data https://spaniol.users.greyc.fr/research/FETD%5e2/.

Table 3. Results on model noise sensitivity evaluation at Layer 2 for French

Models	ELMo				FETD$^2(0.1H)$				FETD$^2(0.1)$				FETD$^2(0.5)$			
Test	Micro-Avg		Macro-Avg		Micro-Avg		Macro-Avg		Micro-Avg		Macro-Avg		Micro-Avg		Macro-Avg	
Noise (η)	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean	CosSim	Euclidean
0.01	0.969	7.409	0.950	9.688	0.993	3.151	0.985	4.614	0.994	2.766	0.987	4.096	**0.997**	**1.915**	**0.992**	**2.940**
0.05	0.948	10.740	0.935	11.916	0.988	4.676	0.982	5.638	0.991	4.097	0.985	4.993	**0.995**	**2.818**	**0.991**	**3.554**
0.10	0.911	15.121	0.902	15.730	0.980	6.772	0.974	7.480	0.984	5.939	0.979	6.610	**0.991**	**4.068**	**0.988**	**4.648**
0.25	0.799	24.343	0.799	24.024	0.955	11.301	0.951	11.623	0.965	9.945	0.961	10.266	**0.981**	**6.875**	**0.978**	**7.205**
0.40	0.681	31.334	0.690	30.502	0.930	14.828	0.926	14.994	0.944	13.084	0.941	13.265	**0.970**	**9.087**	**0.968**	**9.327**
0.55	0.559	37.297	0.574	36.288	0.902	17.865	0.897	18.126	0.923	15.794	0.918	16.057	**0.959**	**10.994**	**0.955**	**11.303**
0.70	0.443	42.404	0.469	40.953	0.875	20.462	0.871	20.581	0.901	18.121	0.897	18.257	**0.948**	**12.625**	**0.945**	**12.855**
0.85	0.336	47.052	0.370	45.202	0.846	22.884	0.843	22.866	0.878	20.298	0.875	20.306	**0.937**	**14.123**	**0.934**	**14.272**
0.99	0.256	50.619	0.302	48.250	0.820	24.850	0.820	24.548	0.858	22.060	0.857	21.812	**0.927**	**15.320**	**0.925**	**15.307**

of tokens before and after adding noise in the sentence. Micro-averaging computes an average over similarity values for individual token versions (i.e. clean and noisy). Macro-averaging first averages the similarity scores for tokens within a sentence and subsequently averaging over all the sentences. We perform macro-averaging in order to capture the influence of sentence lengths stemming from different languages.

$$CosSim(\overrightarrow{emb_t}, \overrightarrow{emb_{\tilde{t}}}) = \frac{\overrightarrow{emb_t} \cdot \overrightarrow{emb_{\tilde{t}}}}{||\overrightarrow{emb_t}|| \cdot ||\overrightarrow{emb_{\tilde{t}}}||} \tag{8}$$

$$Euclidean(\overrightarrow{emb_t}, \overrightarrow{emb_{\tilde{t}}}) = \sqrt{\sum_{i=1}^{n} (emb_t^i - emb_{\tilde{t}}^i)^2} \tag{9}$$

Cosine Similarity: Table 2 and 3 highlight similarity of embedding vectors where a higher value quantifies the lesser divergence and demonstrates a higher robustness towards noise. In Table 2 values vary heavily for the baseline ELMo model. It is worth noting, that even with as little as 1% (i.e. $\eta = 0.01$) of noise, the baseline model embeddings diverge quickly. In contrast, FETD^2 models perform fairly robust as there is comparatively low variance in similarity values. Among the FETD^2 models $\eta = 0.5$ shows the highest resiliency. Thus, FETD^2 models recover from noise at diverse scale by mapping the contextual embeddings of noisy tokens closer to their original versions, while keeping contextual embeddings of clean tokens fairly unaffected.
Euclidean Distance: Table 2 and 3 show the divergence of contextual embeddings where the higher the distance value is, the lower the robustness of the concerned model is. As before, we observe a similar pattern of performance degeneration while FETD^2 remain stable mostly on noisy input. In addition, findings are consistent across languages.

Perplexity
We assess Perplexity (PP) by quantifying "how well a language model predicts the next token if the history tokens contains noise". Equation 10 defines the modified robust perplexity (RPP) measure for token sequence T of length N. We perform the perplexity evaluation at different levels of noise for ELMo and FETD^2 models (cf. Table 4). ELMo performs best without noise in the token history and decreases rapidly with increasing noise-levels. Perplexity values for the English ELMo model differ by over 4,000 as well

Table 4. Evolution of perplexity values with respect to noise in test set sequences

	Test Noise (η)	ELMo	FETD$^2(0.1H)$	FETD$^2(0.1)$	FETD$^2(0.5)$
English	0.00	**65.217**	66.846	66.186	69.941
	0.01	75.383	68.445	**67.347**	70.606
	0.05	81.933	69.387	**68.063**	70.974
	0.10	96.598	71.241	**69.426**	71.685
	0.25	161.221	77.223	**73.833**	73.892
	0.40	285.483	84.330	79.001	**76.326**
	0.55	416.297	89.250	82.522	**77.914**
	0.70	1303.862	105.626	93.967	**82.686**
	0.85	2177.934	113.727	99.462	**84.795**
	0.99	4174.887	125.227	107.231	**87.519**

	Test Noise (η)	ELMo	FETD$^2(0.1H)$	FETD$^2(0.1)$	FETD$^2(0.5)$
French	0.00	**42.073**	43.272	43.636	45.189
	0.01	48.544	**44.257**	44.374	45.606
	0.05	53.526	44.945	**44.907**	45.886
	0.10	64.027	46.205	**45.863**	46.380
	0.25	112.207	50.330	48.982	**47.929**
	0.40	207.603	55.276	52.629	**49.625**
	0.55	312.278	58.780	55.179	**50.748**
	0.70	1043.002	70.460	63.416	**54.076**
	0.85	1769.731	76.375	67.492	**55.566**
	0.99	3473.314	85.047	73.293	**57.546**

as over 3,000 for French. Even at low noise levels such as 10% the perplexity of ELMo increases rapidly. In contrast, FETD^2 models remain considerably stable with increasing noise while achieving comparable performance on the clean test set. In addition, we observe that models trained with lower η perform better at lower noise level whereas the models with higher η remain more consistent towards higher noise level.

$$RPP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{p(t_i|\widetilde{t_1}, \widetilde{t_2}, \ldots, \widetilde{t_{i-1}})}} \tag{10}$$

5.4 Document Classification

Classification Model

We employ a plain document classification model to study the effect of noise on the classification performance. The intuition is to observe and demonstrate the robustness of the contextual embeddings produced by different bi-directional language models on a higher level NLP task. To this end, we represent a document by averaging the embedding vectors of the tokens it contains. Each of the tokens in a document is represented by concatenating the token context-insensitive embedding from Layer 0 along with the contextualized representations from Layer 1 and 2 of LSTMs. We first run the concerned bi-directional language model on individual sentences of the document. Then, we average over the embeddings of tokens in individual sentences. This way, we first compute the sentence representations and then average over the representation vectors of sentences to produce the document representation. By means of macro-averaging the influence of sentence lengths in the overall representation of the document is incorporated.

Subsequently, the document representations are fed into the perceptron-based classification model, which basically consists of a single hidden layer of size 512 with ReLU activation function. The hidden layer is connected to the softmax output layer. The number of units in softmax are equal to the class labels in datasets. The English dataset has 20 output class labels whereas the French dataset has 5 classes. Further, we do not fine-tune the pre-trained contextual embedding models as part of our document classification model training process. The model is trained using the Adam optimizer

Table 5. English documents classification results with model noise sensitivity

Models	ELMo			FETD$^2(0.1H)$			FETD$^2(0.1)$			FETD$^2(0.5)$		
Test Noise (η)	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
0	0.769	0.765	0.766	**0.777**	**0.772**	**0.773**	0.767	0.765	0.763	0.772	0.770	0.769
0.01	0.738	0.734	0.735	0.758	0.752	0.753	**0.760**	**0.756**	**0.755**	0.758	0.754	0.753
0.05	0.743	0.736	0.737	**0.766**	**0.760**	**0.761**	0.759	0.756	0.754	0.761	0.756	0.756
0.10	0.729	0.721	0.722	0.756	0.749	0.750	**0.757**	**0.752**	**0.752**	0.751	0.746	0.746
0.25	0.680	0.654	0.652	0.744	0.735	0.736	0.745	0.737	0.736	**0.754**	**0.750**	**0.748**
0.40	0.621	0.562	0.555	0.716	0.703	0.704	0.723	0.715	0.714	**0.753**	**0.747**	**0.747**
0.55	0.577	0.465	0.442	0.684	0.665	0.664	0.702	0.686	0.686	**0.723**	**0.716**	**0.714**
0.70	0.491	0.342	0.312	0.661	0.628	0.627	0.684	0.656	0.655	**0.718**	**0.706**	**0.705**
0.85	0.299	0.220	0.192	0.623	0.575	0.571	0.667	0.613	0.615	**0.705**	**0.689**	**0.687**
0.99	0.221	0.150	0.116	0.575	0.521	0.515	0.618	0.570	0.568	**0.700**	**0.680**	**0.678**

Table 6. French documents classification results with model noise sensitivity

Models	ELMo			FETD$^2(0.1H)$			FETD$^2(0.1)$			FETD$^2(0.5)$		
Test Noise (η)	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
0	0.906	0.841	0.855	**0.926**	**0.843**	**0.868**	0.897	0.839	0.858	0.884	0.785	0.817
0.01	0.909	0.844	0.858	**0.938**	**0.860**	**0.884**	0.900	0.843	0.861	0.907	0.793	0.833
0.05	0.910	0.843	0.855	**0.935**	**0.858**	**0.882**	0.893	0.833	0.851	0.885	0.778	0.811
0.10	0.903	0.840	0.854	**0.911**	**0.853**	**0.872**	0.894	0.837	0.855	0.882	0.782	0.814
0.25	0.870	0.826	0.827	0.904	0.842	0.864	**0.921**	**0.858**	**0.872**	0.885	0.786	0.818
0.40	0.795	0.736	0.718	**0.914**	**0.852**	**0.874**	0.904	0.833	0.852	0.889	0.789	0.821
0.55	0.703	0.643	0.542	**0.912**	**0.855**	**0.875**	0.910	0.828	0.851	0.883	0.787	0.818
0.70	0.659	0.517	0.375	**0.913**	**0.846**	**0.875**	0.890	0.820	0.837	0.887	0.788	0.820
0.85	0.289	0.495	0.316	0.881	0.820	**0.842**	**0.889**	**0.821**	0.838	0.883	0.784	0.816
0.99	0.261	0.352	0.223	**0.905**	**0.816**	**0.852**	0.874	0.793	0.809	0.884	0.789	0.819

with a learning rate of 0.0001 and betas of (0.99, 0.999) for 50 epochs. We use a step learning rate scheduler with step size of 1 and decay coefficient gamma of 0.95. As we do not train the pre-trained contextual embeddings model as part of classification model training, the document representations are pre-computed.

Classification Results

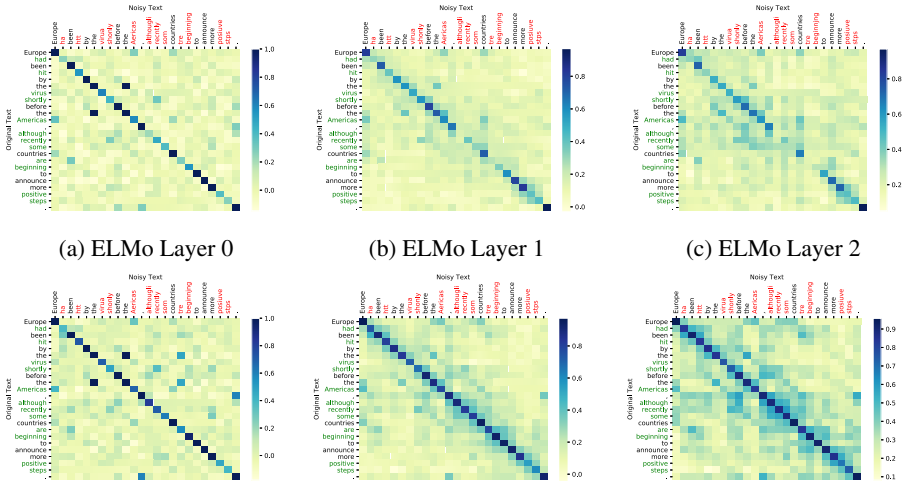
We report the results on the document classification task in Table 5 and 6. Precision, recall, and F1 measures are macro-averaged in order to give equal weight to each of the output class labels and avoid a biasing towards more populated classes. Findings are in-line with the observations from the intrinsic evaluation. From experiments in English we notice that all of the models have almost similar performance on the clean text (i.e. $\eta = 0$) with FETD$^2(0.1H)$ being slightly better than others. The performance of the baseline ELMo classification model is fragile in nature and, thus, the F1 score drops from 0.766 for clean text to 0.116 at $\eta = 0.99$ noise level. However, the F1 score of the FETD$^2(0.5)$ model performance remains quite robust as it only drops by less than 10%. Similar to the perplexity, the FETD^2 models trained with lower noise level perform better on test sets with low(er) noise-level, while models trained with higher η are more robust towards high noise without compromising too much performance

for low(er) noise levels. In experiments on the French dataset a clear dominance of the FETD^2 models can be observed. Scores are overall higher for French than for English. This can be attributed to the fact that there are fewer classes in the French dataset, which makes the problem less complex. In addition, the FETD$^2(0.1H)$ model performs best, although performance differences with other FETD^2 models are less compared to the English dataset.

In summary, extensive experiments on intrinsic evaluation (language-specific data denoising) along with extrinsic evaluation (document classification in different languages) confirm our hypothesis that noise robustness can be added to the bi-directional contextual embedding models without compromising their performance on clean data.

5.5 Success and Error Analysis

Figure 1 depicts the effect of noise on the contextual embeddings of tokens at different layers in an example English sequence by comparing the cosine similarity between embeddings of token pairs from clean and noisy versions of an English sentence with 50% noise. The heatmap color-encodes the scores (yellow $\triangleq$ lowest and blue $\triangleq$ highest). At the context-insensitive Layer 0, it can be seen that the effects of noisy tokens are isolated. Further, Fig. 1a and 1d show that Layer 0 embeddings of noisy tokens diverge for both models but the effect is more visible in ELMo than FETD^2. On Layer 1, noisy tokens degrade their own embeddings and neighboring tokens in case of ELMo (cf. Fig. 1b). In contrast, a dark blue colored diagonal in Fig. 1e for FETD$^2(0.1)$ can be observed, while some of the light colored diagonal boxes from layer 0 have already become darker. This means that FETD$^2(0.1)$ recovers from noise in contrast to ELMo, which becomes even more evident at Layer 2 (cf. Fig. 1f and 1c).



(a) ELMo Layer 0 (b) ELMo Layer 1 (c) ELMo Layer 2

(d) FETD$^2(0.1)$ Layer 0 (e) FETD$^2(0.1)$ Layer 1 (f) FETD$^2(0.1)$ Layer 2

Fig. 1. Cosine similarity of token pairs between a clean and noisy (50%) sentence

6 Conclusion and Outlook

We presented FETD^2, mitigating noisy input data (e.g., from digitization) by utilizing robust contextual embeddings. FETD^2 tackles two aspects of digital preservation at the same time: improving the data quality of digitally and non-digitally born data as well as by providing a language-adaptable framework. While deep neural networks, suffer from performance drop for languages with less ample resources, our Confusion Matrix++ overcomes sparsity issues. In our extensive experiments on English and French datasets, we prove the superiority of FETD^2 compared with the state-of-the-art implementations.

In future, we intend to apply FETD^2 as part of a content semantification pipeline for digitized documents by employing it on OCRed data of historical texts with a subsequent step of named entity recognition and disambiguation utilized for semantic retrieval afterwards. In addition, we consider further adaptations of our noise-resilient bi-directional deep contextualized embeddings framework in the context of other language modeling objectives such as masked or autoregressive language modeling.

Acknowledgements. This work was supported by the RIN RECHERCHE Normandie Digitale research project ASTURIAS contract no. 18E01661. We thank our colleagues for the inspiring discussions.

References

1. Astudillo, R., Amir, S., Ling, W., Silva, M., Trancoso, I.: Learning word Representations from scarce and noisy data with embedding subspaces. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1074–1084. Association for Computational Linguistics, Beijing, China, July 2015. https://www.aclweb.org/anthology/P15-1104
2. Belinkov, Y., Bisk, Y.: Synthetic and natural noise both break neural machine translation. In: International Conference on Learning Representations (2018)
3. Boukkouri, H.E., Ferret, O., Lavergne, T., Noji, H., Zweigenbaum, P., Tsujii, J.: CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters (2020)
4. Chiron, G., Doucet, A., Coustaty, M., Moreux, J.: ICDAR2017 competition on post-OCR text correction. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 01, pp. 1423–1428, November 2017. https://doi.org/10.1109/ICDAR.2017.232
5. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: ELECTRA: pre-training text encoders as discriminators rather than generators. In: ICLR (2020). https://openreview.net/pdf?id=r1xMH1BtvB
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805 (2018)
7. Edizel, B., Piktus, A., Bojanowski, P., Ferreira, R., Grave, E., Silvestri, F.: Misspelling oblivious word embeddings. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), Minneapolis, MN, USA, June 2–7 2019, Vol. 1 (Long and Short Papers), pp. 3226–3234 (2019). https://aclweb.org/anthology/papers/N/N19/N19-1326/

8. Eger, S., et al.: Text processing like humans do: visually attacking and shielding NLP systems. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long and Short Papers), pp. 1634–1647. Association for Computational Linguistics, Minneapolis, Minnesota, June 2019. https://www.aclweb.org/anthology/N19-1165
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997), http://dx.doi.org/10.1162/neco.1997.9.8.1735
10. Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the limits of language modeling. CoRR abs/1602.02410 (2016). http://arxiv.org/abs/1602.02410
11. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016), pp. 2741–2749. AAAI Press (2016)
12. Kumar, A., Makhija, P., Gupta, A.: noisy text data: achilles' heel of BERT. In: Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), pp. 16–21. Association for Computational Linguistics, November 2020. https://doi.org/10.18653/v1/2020.wnut-1.3, https://www.aclweb.org/anthology/2020.wnut-1.3
13. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: a lite BERT for self-supervised learning of language representations. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=H1eA7AEtvS
14. Larson, C., Lahlou, T., Mingels, D., Kulis, Z., Mueller, E.: Telephonetic: making neural language models robust to ASR and semantic noise. ArXiv abs/1906.05678 (2019)
15. Linhares Pontes, E., Hamdi, A., Sidere, N., Doucet, A.: Impact of OCR quality on named entity linking. In: Proceedings of 21st International Conference on Asia-Pacific Digital Libraries (ICADL 2019) (2019)
16. Liza, F.F., Grzes, M.: Improving language modelling with noise-contrastive estimation. In: AAAI (2018)
17. Malykh, V., Logacheva, V., Khakhulin, T.: Robust word vectors: context-informed embeddings for noisy texts. In: Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, pp. 54–63. Association for Computational Linguistics, Brussels, Belgium, November 2018. https://www.aclweb.org/anthology/W18-6108
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
19. Nayak, A., Timmapathini, H., Ponnalagu, K., Venkoparao, V.G.: Domain adaptation challenges of BERT in tokenization and sub-word representations of out-of-vocabulary words. In: Rogers, A., Sedoc, J., Rumshisky, A. (eds.) Proceedings of the 1st Workshop on Insights from Negative Results in NLP, Insights 2020, pp. 1–5. ACL (2020)
20. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
21. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of NAACL (2018)
22. Ren, S., Deng, Y., He, K., Che, W.: Generating natural language adversarial examples through probability weighted word saliency. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 1085–1097. Association for Computational Linguistics, Florence, Italy, July 2019. https://www.aclweb.org/anthology/P19-1103
23. Subramaniam, L., Roy, S., Faruquie, T., Negi, S.: A survey of types of text noise and techniques to handle noisy text. In: ACM International Conference Proceeding Serie, pp. 115–122, January 2009. https://doi.org/10.1145/1568296.1568315
24. Sun, L., et al.: Adv-BERT: BERT is not robust on misspellings! Generating nature adversarial samples on BERT. arXiv preprint arXiv:2003.04985 (2020)

25. Sun, Y., Jiang, H.: Contextual text denoising with masked language model. In: Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019), pp. 286–290. Association for Computational Linguistics, Hong Kong, China, November 2019
26. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc., Red Hook (2017)
27. Wang, W., Tang, B., Wang, R., Wang, L., Ye, A.: A survey on adversarial attacks and defenses in text. arXiv preprint arXiv:1902.07285 (2019)
28. Xiong, W., et al.: TweetQA: a social media focused question answering dataset. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (2019)
29. Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V.: XLNet: generalized autoregressive pretraining for language understanding. CoRR abs/1906.08237 (2019). http://arxiv.org/abs/1906.08237
30. Zhang, W.E., Sheng, Q.Z., Alhazmi, A.A.F.: Generating textual adversarial examples for deep learning models: a survey. arXiv preprint arXiv:1901.06796 (2019)