Reducing tokenizer's tokens per word ratio in Financial domain with T-MuFin BERT Tokenizer

Braulio C. Blanco Lambruschini and **Patricia Becerra-Sanchez** and **Mats Brorsson**
SNT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
{braulio.blanco, patricia.becerra, mats.brorsson}@uni.lu

Maciej Zurad
Yoba S.A., Luxembourg, Luxembourg
maciej.zurad@yoba.com

Abstract

Most domain-specific BERT models are designed to work with short sentences and do not deal with the limitation of 512 tokens in the default BERT tokenizer. This limitation is further exacerbated if the tokenizer has high number of tokens per word ratio (fertility) and thus splits words into several tokens. A term-based multilingual Financial (T-MuFin) BERT tokenizer has been proposed to reduce the fertility of the default BERT tokenizer by extending the base dictionary with the most common financial terms instead of word pieces. One key factor of this proposal is to introduce multiword domain-specific terms without affecting the performance of the BERT models. T-MuFin BERT tokenizer reduces at least 40% of the fertility of long text sequences. T-MuFin BERT improves the fine-tuning of a downstream task by approximately 4% compared to a default fine-tuned model. Hence, by reducing the tokenizer's fertility, the results of explainable methods are more user-friendly.

1 Introduction

The vast amount of available textual information has allowed the development of Natural Language Processing (NLP) models to accelerate in recent years. In 2017, the *Transformer model* was proposed as a big step for NLP models (Vaswani *et al.*, 2017). The Transformer model uses an encoder-decoder architecture. The encoder extracts features from the input and the decoder interprets these features to produce the output. The input is a sequence of numerical vectors that represent the text. These numerical representations of the text are the embeddings. Both blocks take text embeddings as input and parallelize the processing using a self-attention mechanism. This mechanism replaces the sequentially of the existing Recurrent Neural Networks(RNN) processing at once each single *word-piece* and its most related text-pieces. This parallelization sped up the training of bigger models with much more data compared to the RNNs.

In 2018, google published BERT (Devlin *et al.*, 2018). BERT uses the encoder block of the Transformer Architecture for pre-training language models to perform specific eleven NLP tasks like Classification, Named Entity Recognition (NER), Sentiment Analysis (SA), and so on. The input text is divided into word-pieces or tokens and then passed to the BERT's embedding layer, limited to a maximum of 512 tokens.

The resulting pre-trained models contain a high text understanding level and can be fine-tuned for specific tasks and domains. Therefore, this fine-tuning requires less computational resources and less data. Training a BERT model with domain-specific language allows BERT to adapt the general BERT model to the target domain. Two examples of domain adaptation are FinBERT (Araci, 2019) for finance, BioBERT (Lee *et al.*, 2019) for Biomedicine, and so forth. Is important to mention that the BERT model used as the pre-trained model should be in the same language as the domain-specific training corpus. If is required to work with more than one language, a multilanguage BERT version can be used (Google-Research, 2019).

As explained before, BERT is limited to having up to 512 input tokens. The number of tokens depends on the tokenizer's capacity to divide the text into one or several sub-texts. For doing this, BERT uses WordPiece tokenization, which means that the biggest unit is *the word* and *one word can generate one or more tokens*. The measure of this capacity is called *tokenizer's fertility*. The authors in Rust *et al.* (2020) defines the tokenizer's fertility as the measure of the average number of subwords produced per tokenized word. The fertility of one means that each word produces a single token. Higher the fertility, the higher the number of tokens generated per word. Most of the current BERT-based models were trained to understand monolingual

Proceedings of the Joint Workshop of the 5th Financial Technology and Natural Language Processing (FinNLP) and 2nd Multimodal AI For Financial Forecasting (Muffin), Macao, August 20, 2023.

domains (English in our examples) and to perform NLP tasks based on short input sequences.

As previously stated, multilingual BERT models can understand several languages, but its drawback is its higher fertility in comparison with monolingual models, consequently less information can be fed into the model. For long text sequences, this can result in choosing which part of the sequence should be fed into the network and which one should be discarded, thus losing information that could be significant for the desired task.

We propose *Term-based Multilingual Financial BERT* or *T-MuFin BERT* Tokenizer. T-MuFin BERT not only fine-tunes the BERT model in the financial domain as FinBERT and similar models, but also, increases the dictionary size with multi-word financial terms, reducing the tokenizer's fertility below one.

T-MuFin BERT tokenizer is based on a multilingual BERT model fine-tuned with a dataset of annexes of Luxembourgish Annual Accounts in three languages (French, German, and English). In contrast with other financial BERT models, we extract the most frequent financial multi-word terms for extending the base dictionary. We use a n-gram terms generation and then we filter and extract the most frequent financial terms.

Besides to the self-discovered multi-word financial terms, we add to the dictionary also the financial terms from the Standardized Accounting Plan of the European Union [1]. This is because there are many financial terms that are not frequent but important in this domain.

T-MuFin BERT tokenizer was fine-tuned using a Masked Model Learning task (MML) and then tested using a classification task (CL) as the downstream task. T-MuFin BERT tokenizer reduces the average fertility of the default BERT tokenizer by 40-50%. Furthermore, it reduced the number of truncated sentences for paragraphs to almost zero in our downstream task.

Another benefit of having tokens at the term level is to have also explanations at this level. Especially in finance, models have to explain the reason for their predictions, promote transparency and adjust models to reduce any kind of bias. At the subword level, we must weigh the contributions of the subtokens that make up each term in order to facilitate the understanding for the end user. With our approach, we get a direct understandable result.

The next steps of our research are aimed at aligning numerical representations of domain-specific terms for a multilingual scenario. Our proposition can be extended to other domains where is important to give complete numerical meaning to multi-word terms and to clarify the explanations of NLP models such as medicine, law, or science.

2 Related Work

Natural Language Processing (NLP) models help machines to understand and process human language, but machines only understand numbers. In consequence, the first difficulty was to express the text in numbers that a machine can understand. As referred by Khurana *et al.* (2022), the initial models like Bag-Of-Words and One-hot-Encoding were very sparse. Later, models like Word2Vec (Mikolov *et al.*, 2013) and GloVe (Pennington *et al.*, 2014) could reduce the sparsity but still, the context was not considered to give a proper meaning for dealing with ambiguity.

BERT (Devlin *et al.*, 2018) made a big step in NLP, using the encoder of the Transformer Architecture and its self-attention mechanism, for parallelizing the processing of the input in contrast to the existing sequential models like Recurrent Neural Network (RNN) (Rumelhart *et al.*, 1986), Long-short Term Memory (LSTM / BiLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) and Gated Recurrent Units (GRU) (Cho *et al.*, 2014). BERT is a Pre-trained Language Model (PLM), which means that has a vocabulary, relations, and some good level of language understanding in its weights. The smaller BERT model contains 12 encoder layers with 110M parameters. It was trained using Masked Language Model (MLM) and Next Sentence Prediction (NSP) with a corpus with more than 3.3 billion English words in four days using 16 TPU chips. In consequence, we can use the pre-trained model to fine-tune it for specifics downstream tasks, and it will require fewer data and a shorter training time. Some of these NLP downstream tasks are Named Entity Recognition (NER), Classification Task (CL), Sentiment Analysis (SA), Next Sentence Prediction, Machine Translation, Question Answering, and Text Summarization.

A tokenizer converts text into a vector of numbers before feeding into the model. These numeri-

[1] Regulation (EC) No 1606/2002 of the European Parliament and of the Council of 19 July 2002. Source: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32002R1606

cal representations are called embeddings. BERT comes with its own tokenizer and using a dictionary of known words, replaces each word into one or several word-pieces or tokens. Is important to focus on the dictionary of words, because based on that, the meanings of the words were going to be defined by the surrounding context (Tripathy *et al.*, 2021). BERT was originally trained with English corpus, and there are many other versions trained for other languages like CamemBERT for French (Martin *et al.*, 2020), GBERT for German (Chan *et al.*, 2020), RoBERTa for Spanish (Liu *et al.*, 2019) and so on. On contrary of these monolingual models, there are also multilingual models which were trained with 104 different languages for use in multilingual scenarios.

BERT tokenizers were trained under the word-Piece paradigm. It means that one word can be composed of several entries in the dictionary and therefore, several tokens. Especially for Multilingual BERT, the average number of generated tokens per word is higher than in monolingual models. Rust *et al.* (2020) defined this ratio as the *tokenizer's fertility*. Higher fertility means more generated tokens per text input and could cause less information provided into a network in long sequences of text. Multilingual BERT models are useful for using a single model to perform a specific task for a multilanguage dataset. But the main drawback of this is the high fertility of the corresponding tokenizer. BERT models are limited to 512 input tokens, and this is the reason why only short sentences can be processed in this model (avoiding using them together with other models like LSTM).

BERT models can vary also with respect to the specific domain, to improve its performance for specific tasks. This domain adaptation is usually a fine-tuning for the desired task with a domain corpus. Most of the published and available BERT-based models work on fine-tuning the weights of the BERT model itself with the default BERT tokenizer dictionary. It means that they do not add new terms to the default dictionary. In other domains different from finance, we can find a few works on extending the BERT's dictionary with new terms like Douka *et al.* (2021), which creates JuriBERT, a fine-tuned BERT Legal french model which appends 32,000 new entries to CamemBERT dictionary. Wang *et al.* (2019) extended the multilingual dictionary of BERT to reduce the out-of-vocabulary (OOV) words. They use existing tokens to provide meaning to the new one-word terms under two approaches *Joint Mapping* and *Mixture Mapping*. This approach is equivalent to sentence embedding methods such as SciBERT (Beltagy *et al.*, 2019) for science and LegalBERT (Chalkidis *et al.*, 2020) for law. In our case, we train the embeddings of the multiword terms together with their components to try to align all the numerical representations.

In the financial domain, there are some fine-tuned BERT models like FinBERT (Araci, 2019), which is based on bert-base-uncased[2], having a total of 30,522 entries in the dictionary and trained for English. FinBERT uses the default BERT tokenizer to avoid extending the main dictionary. This model outperforms the default BERT model in the financial domain using PhraseBank, a database of financial news, for predicting the sentiment of the short text sequence (SA). Despite this improvement, its fertility is the same as the English default BERT model.

In 2022, FINER-139 by Loukas *et al.* (2022) was released, a financial dataset of 1.1 M annotated sentences. These annotations were obtained from company filings using XBRL tags. These tags are been using by many countries and over time are going to be broadly used. This format requires companies to enrich financial reports with tags that can be read easily. Most of these tokens are numeric values associated with financial concepts. They replace these numeric values with concept-based tokens. They created SEC-BERT, a fine-tuned BERT model where they test the new tokens. With SEC-BERT they showed that fragmented tokens (one word is composed of several tokens) harm BERT's performance and in comparison with FinBERT, their results outperform the default BERT and FinBERT models.

Contrasted with the initial numerical representation of words like Word2Vec (Mikolov *et al.*, 2013), where the vector itself represents the meaning of the specific word, BERT's Embedding layer is not enough to represent the meaning of a word. BERT model process the embeddings in 12 Encoder layers. Due to BERT's ability to handle term ambiguities based on its context, the numerical meaning for each word is defined in the lower Encoder layers.

Thus, BERT uses lower Encoder layers for language understanding and the remaining upper layers for performing the specific NLP downstream task. As an example, the word *"bank"*, can be inter-

[2]https://huggingface.co/bert-base-uncased

preted as *"financial institution"*, *"place to sit"*, or *"place alongside the river"*. There is no clear limit to where the language understanding finishes and the task-related understanding starts. It varies from term to term along the 512 input tokens and the 12 Encoder layers.

The current state of the art with respect to financial tokenizers is mostly limited to terms discovered from financial news datasets. Our contribution is to provide a multi-word dictionary with contextualized embeddings for being used in financial BERT-based models. Allowing to reduce the tokenizer's multi-language fertility, keeping financial multi-word terms as a whole without losing performance.

As studied in detail by Yang *et al.* (2023). Other models than BERT which are having even better results are derived from the Transformer architectures that take only the decoder part. These models like GPT-1 and its derivative works such as GPT-3, ChatGPT, Llama, Bard, and many others, are mainly closed sources and require a huge computational architecture to train and fine-tune. Some open-source GPT-derived, and not heavy, models like Alpaca or Vicuna are available only for research purposes, not for commercial, which limits their application.

3 Dataset

The data used in this study was obtained from the Luxembourg Business Registers (LBR) and is publicly available for download [3]. The LBR Annual Accounts consist of Financial Statements, which can contain only Balance Sheets or also Profit and Loss Statements, and Legal Annexes or Appendixes. These annexes use natural language to provide additional information to the Financial Statements. Although the Financial Statements must follow a specific template, there is no set layout for the legal annexes [4]. We have annexes that have a single page or even more than 15 pages.

For the present work, our dataset only considers the last presented Annual Account for a company that can be active or inactive. In this case, we have 74,539 annual accounts that were processed using OCR tools for scanned documents and HTML content extraction for PDF-readable documents. Most of the documents' pages in the LBR dataset are in French ($\approx 84\%$) and the rest are in German ($\approx 7\%$) and English ($\approx 9\%$), as shown in Table 1.

Language	Documents	(%)	Pages	(%)
French	66,114	88.7	426,610	84.4
German	4,924	6.6	33,966	6.7
English	3,501	4.7	45,112	8.9
Total	74,539	100	505,688	100

Table 1: Dataset distribution per language.

4 Proposition

Our main goal is to create a dictionary of financial frequent terms that will have associated vector embeddings trained with a context of words from the Annexes of the Annual Accounts. This will allow us to disambiguate the terms to a finance context and feed the models with more information without increasing the size of the input layer or reducing their performance in comparison with a base model.

4.1 Dictionary extension and embedding's training

We are using *bert-base-multilingual-uncased* [5] as the base model and tokenizer (with a corresponding dictionary D). If a word is not part of the dictionary D the tokenizer splits the word into a set of sub-words and/or characters, in consequence, tokenizing a word can result in a set of one or more tokens. For this reason, we are extending the base dictionary with domain-specific terms and then fine-tuning the model to calculate the vector embeddings. For the fine-tuning, we use Masked Language Model (MLM).

The terms to be added to the dictionary are the result of performing the following steps.

1. *Candidates extraction*: We are using multi-word terms for creating the tokenizer's dictionary. Hereby, the list of candidates is the result of the text extraction in the form of n-grams. We defined empirically n=5, which covers most of the financial terms.

2. *Candidates cleaning*: For each term, we perform a set of cleaning tasks: (A) removing enumerators (like a., note 1.3, iv., etc); (B) removing noisy characters (;-,); (C) using regular expressions identifying and replacing dates

[3] https://www.lbr.lu

[4] https://guichet.public.lu/en/entreprises/gestion-juridique-comptabilite/comptable/enregistrement/methodes-etablissement-comptes-annuels.html

[5] https://huggingface.co/bert-base-multilingual-uncased

and numbers with special *[DATE]* and *[NUMBER]* tags respectively; (D) and replace apostrophes with blank spaces.

3. *Top terms selection*: The cleaned candidates' list is sorted by frequency and then we select the top τ terms by language. This list will be our *base list*.

4. *Financial statements labeled terms*: The tree structure of a financial report like a balance sheet or profit and loss statements are full of multi-word terms that have a semantic relation with their surrounding neighbors in the three. For example *"convertible loans"* with *"non convertible loans"* (sibling) and *"creditors"* (parent). This list is appended in the *base list*.

5. *Terms decomposition*: Each term in the *base list* is decomposed and the main subterms are added to the *base list*. For instance, if the term is "subscribed capital amount", the extracted decomposed terms are *"subscribed capital"*, *"capital amount"*, *"subscribed"*, *"capital"* and *"amount"*. The main terms and their components are going to be inserted into our *base list* υ.

For example, with T-MuFin BERT tokenizer, a financial multi-word term like *"capital investment subsidies"* will be considered as a single token because it is a frequent term. Moreover, this term is added to the dictionary as a whole, also we are adding its term components. For instance, for the previous example, the tokenizer also going to generate the following tokens: *"capital investment subsidies"*, *"capital investment"*, *"investment subsidies"*, *"capital"*, *"investment"* and *"subsidies"*. During training, all those tokens are going to be numerically related to each other.

Only the terms t in υ that are not part of the dictionary D are included in the dictionary D ($t_{new} \leftarrow t \in \upsilon \quad \& \quad t \notin \upsilon$). The increment in the dictionary size causes a resizing of the embedding layer in the base BERT model before training. For the fine-tuning of the default BERT, we use as well the MLM task.

The training and testing datasets consist in sentences that have at least one t_{new}. For each sentence, the same *Candidates cleaning* step that we use for obtaining the most frequent terms are used. With the cleaned sentence, we identify t_{new} and we extract the context surrounding it. The context consists of the previous κ words and the following κ words to t_{new} (κ: context size). If κ=2, we take two words previous to the frequent term and two words posterior to the new term (if exists). Finally, we replace 20% of context's tokens with the `[MASK]` token, considering that the masked word should have more than five characters (to avoid connectors, negations and so on to be masked) and do not mask tokens that are part of a word. If a term contains n words, we also append to the training and testing dataset each component of t_{new} tokenized. This resulting dataset is shuffled and we use 70 % of the samples for training and the remaining 30% for testing. Additionally, for training the financial statements labeled terms, we add the terms and the training context are the siblings and parents until the first level, this will allow us to create a strong relationship between terms in the same financial category.

Hyper-parameter	**Values**
σ: dataset size	10, 25, 50, 75 and 100%
ϕ: frozen layers	2, 4, 6, 8 and 10
λ: learning rate	1e-5, 2e-5, 3e-5 and 4e-5
δ: dropout	10s, 15 and 20%
κ: context size	2,3,4 and 5

Table 2: Testing values for each hyper-parameter for training T-MuFin embeddings.

We test different hyper-parameters that lead us to the best-performed model. These hyper-parameters are *κ: context size*; *λ: optimizer's learning rate*; *δ: BERT's dropout percentage*; *σ: dataset size* for (a) extracting the frequent terms and (b) for training the MLM model; and *ϕ: number of BERT encoder layers to freeze*. Table 2 shows the different test case values per each hyperparameter. We use AdamW as the optimizer.

Freezing from 2 to 12 encoder BERT layers of the BERT model during fine-tuning allows us to reduce the memory used in the GPU and make the model put the effort into the embeddings' training and also the upper encoder BERT layers (remaining layers and the final dense layer). We are going to try different numbers of layers to freeze in order to get the best results and take advantage of the memory in the GPU, increasing the batch size. Figure 1 shows the layers to freeze and the corresponding layers to train in the MLM task.

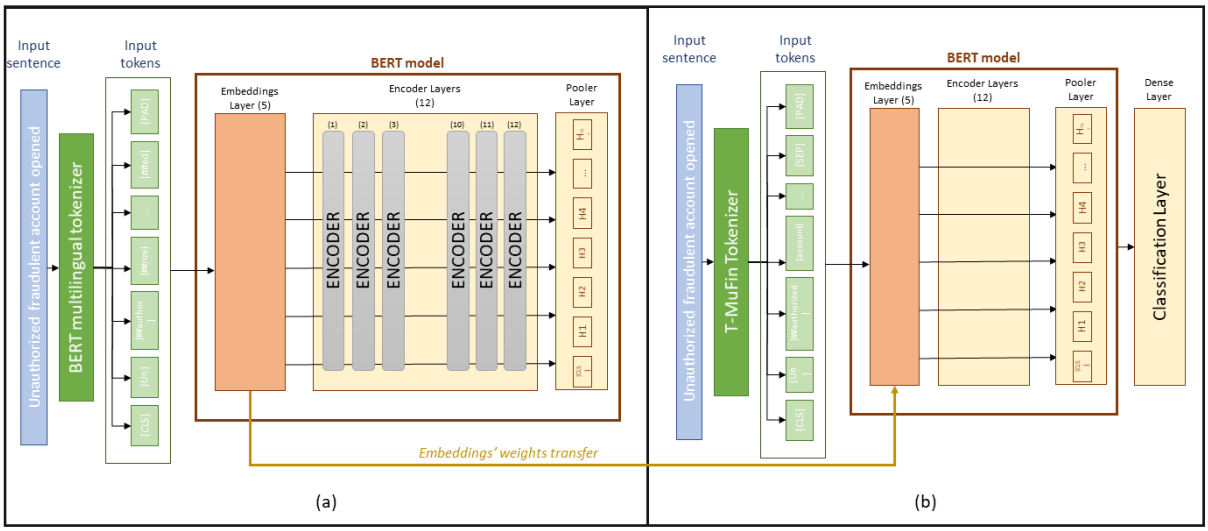

Figure 1: (a) shows the BERT model and tokenizer. The layers to freeze are up to 10 first encoder layers, for each test case, the remaining layers will be training layers; (b) shows the model to test the performance of T-MuFin BERT tokenizer which includes a final dense layer for the specific classification task, transferring the first unfreezed layers from the MLM model

4.2 Classification as Downstream task

To test the performance of T-MuFin BERT tokenizer, we use a Classification Task (CL) as the downstream task. The model will predict nine financial categories based on a text paragraph. For doing this, from each Annual Account Annex, we extracted the last-level subtitles and their corresponding paragraphs. Then we selected the first most frequent subtitles and manually assigned them one of nine financial categories. These categories are (1) Asset, (2) Capital, (3) Deposit, (4) Expenses, (5) Investment, (6) Obligations, (7) Personal, (8) Receivables, and (9) Taxes. The training and testing datasets are obtained from subtitles that have an assigned financial category and also have paragraphs with at least 20 words. The evaluation dataset is composed of 10,000 documents.

The base model is a *bert-base-multilingual-uncased* BertModel with its own default tokenizer. For comparing our tokenizers with respect to the base model we copy the weights from T-MuFin BERT tokenizer (Embedding and unfreezed encoder layers) and train in the Classification task for 5 epochs like the base model. All the hyperparameters are the same for the base model and T-MuFin-based models.

For these models, we only add a classification layer after BERT model. We take only the first hidden state (`[CLS]`) of the BERT model as input for our dropout and classification layer (Dense Layer). The dropout percentage of the classification layer

Hyper-parameter	Value
σ: dataset size	34K working samples%
ϕ: frozen layers	embeddings and first 10
λ: learning rate	1e-5
δ: dropout	10%

Table 3: Hyper-parameter for downstream task (CL) evaluation.

and the BERT dropout is the same (δ). We are using CrossEntropyLoss as our loss function, using the class weights to deal with unbalanced datasets.

Table 3 shows the hyperparameters used for this model. Using AdamW as the optimizer and a batch size of 30.

Equation 1 evaluates the tokenizers' fertility ψ, which is the average number of tokens generated per word. Additionally, as shown in Equation 2, we are going to measure the proportion of samples that were truncated because of the 510 tokens limitation (Π), considering reducing two special tokens for `[CLS]` and `[SEP]`.

$$\psi = \sum_{i=0}^{n} \frac{(Number_{tokens}/Number_{words})}{n} \quad (1)$$

$$\Pi = \sum_{i=0}^{n} \begin{cases} 1 & , Number_{tokens} > 510 \\ 0 & , \text{otherwise.} \end{cases} \quad (2)$$

Moreover, in the results section, we show other

Dataset size (σ)	Time (hr.)	F1 Score Training	F1 Score Testing
10	**6.78**	**85.62%**	**77.65%**
25	11.90	85.64%	76.03%
50	18.37	86.30%	78.19%
75	22.52	87.24%	79.59%

Table 4: Impact of dataset size in tokenizer's training.

Frozen Layer (ϕ)	F1 Score Training	F1 Score Testing
2	85.63%	84.79 %
4	85.78%	84.72 %
6	86.59%	85.07 %
8	87.37%	84.95 %
10	**86.41%**	**85.55%**

Table 5: Evaluation of the number of frozen layers.

Learning rate Layer (λ)	F1 Score Training	F1 Score Testing
1e-5	**84.86 %**	**83.72%**
2e-5	84.52%	83.16%
3e-5	83.66%	82.40%
4e-5	83.53%	82.17%
5e-5	83.54%	82.25%

Table 6: Evaluation of learning rate λ.

Dropout Layer (ϕ)	F1 Score Training	F1 Score Testing
10	**85.37%**	**80.80%**
15	85.22%	80.72%
20	85.40%	80.40%

Table 7: Evaluation of dropout percentage.

statistical measures like the average number of tokens per document, the average number of words per document, and the average number of tokens per word.

To evaluate the explainability results with the default BERT tokenizer and T-MuFin BERT tokenizer, we use the fine-tuned models with Captum.ai [6], which analyze the most important input features that the model takes into consideration for making a single prediction.

5 Experiment and Results

The subsection 5.1 shows the results for training the tokenizer. The best-performed tokenizer has been selected to be evaluated in the CL task (subsection 5.1).

5.1 Tokenizer: Embedding's training

In this section, we train the tokenizer with different hyper-parameters such as κ, λ, δ, and ϕ.

5.1.1 Dataset size selection (σ)

Table 4 shows the impact of the different dataset sizes in the overall F1 Score, having: $\kappa = 2$, $\lambda = 5e-5$ and $\delta = 10\%$.

We decided to train with only 10% of the data based on the results in Table 4. This is because the impact on the performance is not high and the time required for tokenizer's training is significantly lower.

Once defined the dataset size σ for the tokenizer's training, we execute several test cases with different hyper-parameters: ϕ, λ, δ, and κ. When a hyper-parameter is being evaluated the default values for the others are $\phi = 0$, $\lambda = 5e-5$, $\delta = 10\%$, $\kappa = 2$.

Table 6 shows that the best learning rate λ is 1e-5, with a F1 Score of 83.72%. We also tested others lowers and biggers learning rates whose performance were lower.

Table 7 shows that the best dropout percentage δ is 10%, with a F1 Score of 80.80%. Also, we test bigger values like 20% and 30% but the performance drops drastically.

Table 8 shows that the best context size is $\kappa = 5$, with a F1 Score of 85.05%.

As shown in Table 9 with the best combination of these hyper-parameters, we got the following results in 5 epochs. We use this model for fine-tuning the BertModel for the selected downstream task.

5.2 Tokenizer's Performance evaluation

5.2.1 Establishing CL baseline

For the evaluation of T-MuFin BERT tokenizer, first, we evaluate the performance of BERT baseline model and the parameters specified in Table 3. In Table 10 is shown the performance F1 score for training, testing and also the default fertility and the proportion of samples that did not fit in the model.

As our Classification task is working only with small text to determine if the dictionary extension of the tokenizer affects the downstream tasks, the effect on the tokenizer's fertility is not easy to appreciate. Hence, we use T-MuFin tokenizer to process 1,000 Annual Accounts' Annexes. As we can

[6] https://captum.ai/

Context Size Layer (κ)	F1 Score Training	F1 Score Testing
2	83.54%	82.25%
3	85.10%	84.05%
4	84.85%	83.83%
5	**85.99%**	**85.05%**

Table 8: Evaluation of context size κ.

F1 Score Training	F1 Score Testing
89.87%	89.08%

Table 9: Results for training with the best hyper-parameters

see in Table 11, on average for feeding a BERT model with complete annexes of Luxembourgish Annual Accounts, we require on average 867 tokens.

5.2.2 T-MuFin BERT Tokenizer results

With the best performed T-MuFin BERT tokenizer from Table 9 and the same hyper-parameters as the baseline in our downstream task, we got an increment of the F1 score for testing from 94.97% to 98.80% as shown in Table 12.

As shown in Table 12, T-MuFin BERT tokenizer could reduce from 1.2592 to 0.8906 ($\approx 41\%$) the fertility ψ with respect to the default BERT tokenizer and reduces almost to zero (0.2%) the truncated sentences Π.

As shown in Table 13, we can see that the number of tokens per word was reduced on average at ≈ 1.0, this is mainly because a big group of multi-term tokens is reducing the average of the words that produce more than one token. The average fertility of the document was improved $\approx 50\%$.

Figure 2 shows the difference in terms of explainable NLP with T-MuFin tokenizer, which makes more easy to understand for the final user and also allows the NLP model to focus on the term as a whole.

6 Discussion

Most of the current NLP projects using BERT-based models are related to domain adaptation while keeping the same default dictionary. However, researchers who are increasing the dictionary are only considering single-word terms. T-MuFin increases its dictionary with multiword terms in finance without affecting the performance

F1 Score Training	F1 Score Testing	Fertility (ψ)	% Truncated samples (Π)
95.08%	94.97%	1.2592	0.79

Table 10: Performance of BERT BaseLine

Metric	Value
Avg. Number of tokens per document	1,182
Avg. Number of words per document	867
Avg. Number of tokens per word	1.37
Avg. fertility per document (ψ)	0.126

Table 11: BaseLine for fertility ratios

F1 Score Training	F1 Score Testing	Fertility (ψ)	% Truncated samples (Π)
98.20%	98.80%	0.8906	0.2

Table 12: Results for training with T-MuFin BERT tokenizer

of downstream tasks and even increasing them due to the self-nature of the fine-tuning. Adding these domain-specific terms always reduces the fertility of the tokenizer. For T-MuFin, this fertility goes below one, which means that we can feed more information into the BERT-based models.

In our proposed terms discovery method, the discovery of new terms is not only based on including frequent multi-word terms in the new dictionary but also on including their corresponding components and training them together.

Using T-MuFin BERT tokenizer can help explainable models produce more understandable results. This is because financial terms are no longer divided into word-pieces, which can make it more difficult for the user to process at first sight.

This proposed method for terms discovery and training can be applied to any other domain like Medicine, Law, or Science; where most of the multiword terms tend to be understood as a whole.

Is important to mention that in the lasts weeks, the GPT-based model is reaching very good performance for NLP tasks, but unlike BERT-based models, those models can not be used for commercial purposes unless the service is purchased.

7 Conclusion

By training the BERT embeddings with financial data and extending the dictionary with the most common multi-word financial terms, T-MuFin BERT tokenizer can increase the information feed

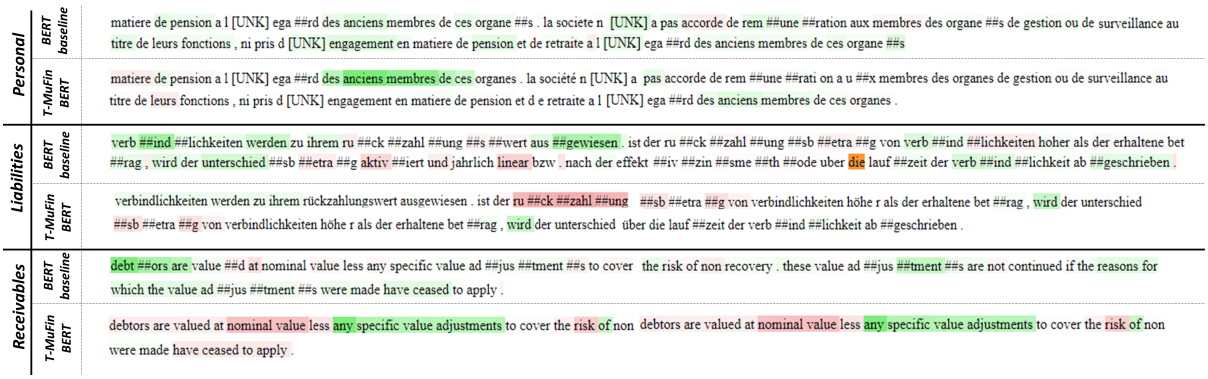

Figure 2: Comparison between NLP explainability using the default BERT multilingual tokenizer for French, German, and English using the Classification fine-tuned model (left), and the fine-tuned model using T-MuFin BERT tokenizer (right).

Table 13: BaseLine for fertility ratios

Metric	Value	Improvement
Avg. number of tokens per document	868	26%
Avg. number of words per document	867	-
Avg. number of tokens per word	1.02	34%
Avg. fertility per document (ψ)	0.065	48.41%

into a BERT model. When we freeze the first 10 layers of BERT to calculate the weights of the embeddings, we force the model to disambiguate the terms at the beginning of the model, in the Embeddings layers. Later these weights are transferred to a default BERT and used for any downstream task. With T-MuFin tokenizer, we could increase the F1 score from 94.97% to 98.80% with respect to the baseline of the downstream task. This means that we are not losing performance with the newly trained multiword terms; on the contrary, we increase it.

On the other hand, we reduced between $\approx 40\%$ and $\approx 50\%$ of the fertility of the default BERT tokenizer for short and long text sentences respectively. We could also reduce almost to zero the truncation of long paragraphs and facilitate explainable AI.

For future steps, we plan to align the numerical representations of the terms along different languages. This means that the same term in different languages should have very similar numerical representation, making it easier to include more languages in the NLP models.

Acknowledgements

This work has been partly funded by the Luxembourg National Research Fund (FNR) under contract number 15403349.

References

Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models, 2019.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. 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)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. Legal-bert: The muppets straight out of law school, 2020.

Branden Chan, Stefan Schweter, and Timo Möller. German's next language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Stella Douka, Hadi Abdine, Michalis Vazirgiannis, Rajaa El Hamdani, and David Restrepo Amariles. Juribert: A masked-language model adaptation for french legal text. *CoRR*, abs/2110.01485, 2021.

Google-Research. Bert: Multilingual models, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: state of the art, current trends and challenges. In Springer, editor, *Multimedia Tools and Applications*, 2022.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

Lefteris Loukas, Manos Fergadiotis, Ilias Chalkidis, Eirini Spyropoulou, Prodromos Malakasiotis, Ion Androutsopoulos, and Georgios Paliouras. FiNER: Financial numeric entity recognition for XBRL tagging. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4419–4431, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suá rez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. 2014.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How good is your tokenizer? on the monolingual performance of multilingual language models, 2020.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

Jatin Karthik Tripathy, Sibi Chakkaravarthy Sethuraman, Meenalosini Vimal Cruz, Anupama Namburu, Mangalraj P., Nandha Kumar R., Sudhakar Ilango S, and Vaidehi Vijayakumar. Comprehensive analysis of embeddings and pre-training in nlp. *Computer Science Review*, 42:100433, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, and Dong Yu. Improving pre-trained multilingual model with vocabulary expansion. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 316–327, Hong Kong, China, November 2019. Association for Computational Linguistics.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond, 2023.