PublishInCovid19 at the FinSBD-2 Task: Sentence and List Extraction in Noisy PDF Text Using a Hybrid Deep Learning and Rule-Based Approach

Janvijay Singh
Flipkart Private Limited
janvijay.singh@flipkart.com

Abstract

This paper describes the approach that we employed to tackle the FinSBD-2 shared task in IJCAI-2020. FinSBD-2 comprises of two subtasks: 1) extraction of boundaries of sentences, lists and items from noisy PDF (financial documents) text and 2) organisation of lists and items in a visual hierarchy. We solve these subtasks in two phases. In the first phase, we pre-process the data to embed relevant visual cues and form non-recursive and non-hierarchical tags. We then formulate the boundary prediction problem as a sequence labelling task and evaluate two neural architectures, namely BiLSTM-CRF and BERT. In the second phase, we identify the recursive relation among different items as well as their hierarchical visual layout. A rule-based method is used to achieve desired recursiveness and hierarchy in tags predicted from the first phase. The rules are based on visual cues like left-indentation and bullet-style of items. Our combined final approach achieved an F1-score of 0.937 on subtask-1 and 0.844 on subtask-2 for the English test set. We were ranked first overall in terms of MEAN F1-score.

1 Introduction

Sentence Boundary Detection (SBD) involves identification of boundaries (begin/from-end/to token indices) of sentences within a text. Sentences are the foundational units of most of the Natural Language Processing (NLP) applications including Part of Speech Tagging, Discourse Parsing [Polanyi *et al.*, 2004], Machine Translation, Sentiment Analysis and Information Retrieval [Read *et al.*, 2012]. Hence, errors introduced during the extraction of sentences can propagate further and degrade the performance of the complete NLP pipeline. Despite its fundamental importance in building NLP-based solutions, SBD has so far not received enough attention. Most of the previous research work in this area has focussed on formal texts [Agarwal *et al.*, 2005; Kiss and Strunk, 2006; Akita *et al.*, 2006; Gillick, 2009; Kreuzthaler and Schulz, 2015], such as news and European parliament proceedings, where existing rule-based and machine learning methods are highly accurate due to the perfectly clean data. SBD remains a challenging task when the input text is noisy or unstructured.

Documents encoded in machine-readable formats (such as Adobe PDF format) have the exact layout of human-readable documents. However, the text extracted from such documents loses these formatting features and is highly noisy or unstructured. Financial documents or prospectus are also encoded in such machine-readable formats. Apart from the textual paragraphs, financial documents contain tabular data, titles, sub-titles, keywords, lists, headers and footers, which further increases the complexity of SBD by making the extracted text noisier. At IJCAI-2019, the FinSBD shared task [Azzi *et al.*, 2019] was proposed to address the SBD research focussing on noisy text extracted from financial documents. In specific, FinSBD was concerned with the identification of begin and end token indices for sentences present in such noisy text.

The first step to extract information from financial documents is to transform the noisy or unstructured text into the semi-structured (with well-marked boundaries for information elements) text. Apart from the sentences, lists are the other most commonly occurring element in financial documents. Contrary to sentences, lists include multiple sentences and a visible hierarchy of information. Therefore, identifying the intra-list hierarchy and distinction in sentences and lists can make information extraction much more accurate. Building on this idea, FinSBD-2 shared task was proposed at IJCAI-2020. It comprises of following two subtasks: 1) identification of boundaries for sentences, items and lists and 2) identification of the hierarchy of items contained in the lists. We tackle these sub-tasks in two phases, using a hybrid rule-based and deep learning approach.

In the first phase, we pre-process the data to embed relevant visual cues and form non-recursive and non-hierarchical list/item tags. We then formulate the boundary prediction problem as a sequence labelling task and evaluate two neural architectures, namely BiLSTM-CRF and BERT. In the second phase, we employ a rule-based approach to identify the recursive and hierarchical relation in the items predicted from the first phase. The rules are based on visual cues like left-indentation and bullet-style of items.

The rest of the paper is structured as follows: Section 2 states the task definition. Section 3 describes the specifics of our methodology. Section 4 explains the experimental setup and the results, and Section 5 concludes the paper.

2 Task Definition

In the literature, SBD has been attempted using several approaches. These approaches fall into three major categories: (a) rule-based approaches, which rely on hand-crafted heuristics (e.g. Stanford Core NLP[1], SpaCy[2] etc); (b) supervised machine learning based approaches, which utilise annotated training data to predict boundaries ([Reynar and Ratnaparkhi, 1997; Gillick, 2009; Du and Huang, 2019]); and (c) unsupervised machine learning approaches, where the training data is unlabelled ([Read *et al.*, 2012]). Rule-based methods are widely used for SBD since they provide ease of usage and decent performance for most of the NLP tasks. In presence of data with annotated boundaries, supervised machine learning approaches tend to provide the best performance.

Training dataset provided with FinSBD-2 shared task comprises of following:

1. String of text extracted from financial documents;
2. Bounding box coordinates corresponding to each character in the text; and
3. Set of pairwise (begin/from-end/to) character indices for some classes, *namely sentences, lists, items, item1, item2, item3 and item4*.

Set *sentences* and *lists* have non-overlapping elements; this implies that a character cannot be a part of both sentence and list segment. Each element of set *items* overlaps with exactly one element in set *lists*, this implies that a list can contain multiple items. Similar to lists, an item can contain multiple items inside it. Hence, items and lists are recursive in nature. Sets *item1, item2, item3 and item4* provide the hierarchical structure of the list. Set *item1* comprises of items which are one-level inside the containing list. Set *item2* comprises of items which are one-level inside containing *item1*, and two-level inside containing *list*. The hierarchical structure for items in set *item3* and *item4* is defined similarly. Set *items* is a union of sets *item1, item2, item3 and item4*.

Modelling this task as a sequence-labelling problem is not trivial because of a few reasons. Firstly, due to the recursiveness in lists and items, the end boundary of multiple list and item segments can share the same indices. This will require us to classify a few token indices into multiple classes. Secondly, recursiveness causes list and item segments to span over up to 1500 tokens (words). Since most of the sequence labelling models learn far smaller contextual dependencies, it becomes essential to deal with this recursiveness at pre-processing stage only. Thirdly, items at different hierarchical levels are indistinguishable from one another if the context is constrained to a small length. Therefore, determining hierarchy based on visual cues such as bullet-style and left indentation should be carried out once the boundaries of lists and items are precisely known. To formulate this task as a sequence labelling problem, we pre-process the dataset to remove the recursiveness and hierarchy among lists and items.

With non-recursive and non-hierarchical boundaries for lists and items, we formulate the boundary prediction problem as a sequence labelling task. In sequence labelling, each token in the sequence is classified to one among certain classes (classes are commonly represented using IOB tagging scheme [Evang *et al.*, 2013]). For our task, we define the following seven classes:

- S-SEN: begin and end of a sentence with a single token;
- B-SEN: begin of a sentence segment;
- E-SEN: end of a sentence segment;
- S-IT: begin and end of list/item with a single token;
- B-IT: begin of a list/item segment;
- E-IT: end of a list/item segment;
- O: other, neither of the classes mentioned above.

We utilise this sequence labelling model to predict boundaries for sentences and non-hierarchical lists/items. We then employ a rule-based method to identify the recursiveness and hierarchy in the previously predicted list/item segments. The rules for this method are based on left-indentation (determined from bounding-box coordinates) and bullet-style. Specifics of the different phases mentioned here are described in subsequent sections.

3 Methodology

Our approach is composed of two phases. In the first phase, we learn to predict the non-hierarchical and non-recursive sentence, list and item boundaries. Details of the first phase are included in sub-section 4.1 and 4.2. In the second phase, we identify the recursiveness and hierarchy in segments predicted from the first phase using a rule-based approach. Section 4.3 and 4.4 describe the details of the second phase.

3.1 Pre-Processing Dataset

The dataset provided with FinSBD-2 shared task cannot be used directly to train our sequence labelling model because of a couple of reasons. Firstly, the dataset contains the text extracted from financial documents as a large string of characters. Moreover, the segment labels are also provided at the character level. In contrast, our sequence labelling models operate at word level and on a smaller input sequence length. Secondly, as described in the previous section, non-hierarchical and non-recursive list/item labels are more suited to the task of sequence labelling. Therefore, we recreate the training set using the following pre-processing strategy:

1. We create a *unified set* of all the segments in set *lists* and *items*. We call a segment X as a child of segment Y, if begin index of $Y \leq$ begin index of X and end index of $X \leq$ end index of Y. For each segment X in the *unified set* if X has atleast one child segment, we change the end index of X to the minimum begin index of all its child segments. With these steps, the final *unified set* contains non-hierarchical and non-recursive list/item boundaries.
2. We tokenize the string of characters extracted from financial documents using word_tokenizer [3] from NLTK. In addition to tokenization this removes extra white-space characters (such as \n) from the text. We then

[1] https://stanfordnlp.github.io/CoreNLP/ssplit.html

[2] https://spacy.io/usage/linguistic-features/#sbd

[3] https://www.nltk.org/api/nltk.tokenize.html

assign a tag (one from S-SEN, B-SEN, E-SEN, S-IT, B-IT, E-IT and O) to each tokenized word, utilizing the character based indices for sentence and list/item (from *unified set*) segments. Hence, we achieve the word/tag sequence for each financial document.

3. The x-coordinates provided with the dataset increases from left to right on a page in PDF, whereas y-coordinates increases from top to bottom. We define a *visual line* as a contiguous sub-sequence of words which have overlapping y-coordinate bounds. Left-indentation for a *visual line* is the minimum x-coordinate of a character present in it. To embed visual cues, we embed dummy tokens $\langle tabopenX \rangle$ and $\langle tabcloseX \rangle$ at the beginning and ending of *visual line* respectively. Here X is equal to left-indentation of visual line divided (integer division) by five units. These cues help us achieve slightly better metrics at sequence labelling task.
4. We use a sliding window (parameterised by the window and hop length) upon word/tag sequence to achieve sequences of smaller length. We use a hop length of 20 words, to ensure that the sequence labelling model is provided with varied contexts.

3.2 Deep Learning Models for Sequence Labelling

Deep Learning (DL) models have achieved state-of-the-art performance in most of the NLP tasks. In the domains of sequence labelling tasks (such as Named Entity Recognition[4] and Part of Speech Tagging[5]), recurrent neural network [Peters *et al.*, 2018; Straková *et al.*, 2019] and multi-headed self-attention based DL models [Devlin *et al.*, 2019] have surpassed performance of all other methods. In our work, we evaluate two neural architectures, namely, BiLSTM-CRF and BERT, which are described below.

BiLSTM-CRF

Recurrent Neural Networks (RNNs) are suited to sequential input data since they execute the same function at each time-step and allow the model to share parameters across input sequence. In order to predict at a time-step, RNNs utilise a hidden vector which captures the useful information from past time-steps. In case of longer input sequences, RNNs suffer from the problem of vanishing gradients. Long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] was introduced to alleviate the problem of vanishing gradients. LSTM employ a gating mechanism to capture long-range dependencies in the input sequence. In contrast to uni-directional LSTM, bidirectional LSTM (BiLSTM) [Schuster and Paliwal, 1997] makes prediction by utilising hidden state vector from past as well as future time-steps.

Our BiLSTM-CRF model is composed of: 1) a character-level BiLSTM layer; 2) a dropout layer [Srivastava *et al.*, 2014]; 3) a word-level BiLSTM layer; and 4) a linear-chain Conditional Random Field (CRF) [Sutton and McCallum, 2012]. The character-level BiLSTM operates on words and is employed to learn morphological features from them. We concatenate the output vectors of character-level BiLSTM

[4]http://nlpprogress.com/english/named_entity_recognition.html
[5]http://nlpprogress.com/english/part-of-speech_tagging.html

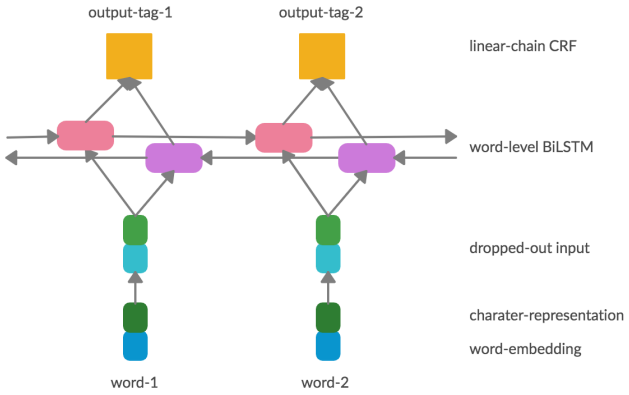

Figure 1: The architechture of our BiLSTM-CRF model.

(character representation) with pretrained word embeddings (GloVe [Pennington *et al.*, 2014]) to provide our model with more powerful word representations. In order to prevent the model from depending on one representation or the other too strongly, we pass this concatenated vector through a dropout layer. The output of the dropout layer is then passed to the word-level BiLSTM layer, which outputs a vector corresponding to each word in the input sequence. For our task, output labels share dependencies among themselves, such as an end-tag is followed by a begin-tag. In order to model these dependencies, we use a linear-chain CRF at the end, instead of the commonly used softmax layer. A linear-chain CRF is parameterised by a transition matrix (transitions within output labels), and consequently is capable of learning dependencies in the output sequence. The complete architecture of our BiLSTM-CRF model for this task is shown in Fig. 1.

BERT

Transformer [Vaswani *et al.*, 2017] based neural models have shown promising results in most of the NLP tasks. Its architecture is composed of feed-forward layers and self-attention blocks. The fundamental difference in RNN based models and transformer is that transformer does not rely on recurrence mechanism to learn the dependencies in the input sequence. Instead, on each input time step, they employ self-attention. Attention can be thought of as a mechanism to map a query and a set of key-value pair to an output, where query, keys, values and output are all vectors. In the case of self-attention, for each vector in the input sequence, a separate feed-forward layer is used to compute query, key and value vectors. Attention-score for a input vector, is determined as the output of a compatibility function, which operates on input's key and the some query vector. The output of self attention mechanism is weighted sum of value vectors, where weight is determined by the attention-score. In case of multi-headed attention, multiple blocks of such self-attention modules operate on the input sequence.

Transformer's encoder is composed of 6 identical layers, where each layer is composed of two sublayers. These two layers are multi-head self-attention and a position-wise fully

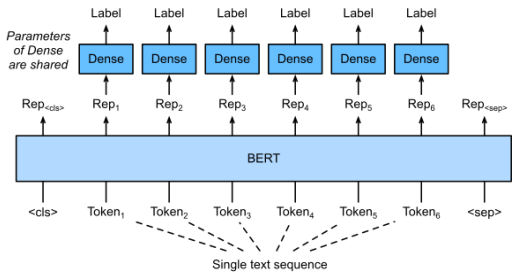

Figure 2: The token-tagging architecture for fine-tuning BERT.

connected feed-forward network. A residual connection is used around each sublayer, followed by layer normalisation. BERT [Devlin *et al.*, 2019] utilises a multi-layer Transformer encoder to pre-train deep bidirectional representations by jointly conditioning on both left and right context across all layers. As a result, pre-trained BERT representations can be fine-tuned conveniently using only one additional output layer.

For a given token, BERT's input representation is constructed by summing the corresponding token, segment, and position embeddings. BERT is trained using two unsupervised prediction tasks, Masked Language Model and Next Sentence Prediction. In order to fine-tune BERT on a sequence labelling task, BERT representation of every token of the input text is fed into the same extra fully-connected layers to output the label of the token. The predictions are not conditioned on the surrounding predictions. Since we view our task as a sequence labelling problem, we configure BERT to instantiate the token tagging architecture which is shown in Fig 2.

3.3 Post-Processing Predicted Tags

To extract a sentence or list/item segment, both begin and end tags need to be predicted accurately. From predictions on the validation dataset, we realise that many unretrieved segments have a single missing begin or end tag. In order to recover as many as possible missing/erroneous tags, we employ (in the order as described) the following post-processing strategy on the predicted tags:

1. If E-IT tag is missing for a B-IT tag, then E-IT occurs at the end of a *visual line* (one with B-IT tag or the following ones) if:
   - first tag in next *visual line* is B-IT or B-SEN.
   - last tag in the current *visual line* is E-SEN.
   - vertical-spacing between current *visual line* and next *visual line* is greater than most frequent inter *visual line* spacing (specific to a document).
2. If B-IT tag is missing for a E-IT tag, then B-IT occurs at:
   - the word next to the just (all the tags in between are O) previously occurring E-IT or E-SEN tag.
   - the just previously occurring B-SEN.
3. If B-SEN tag is missing for a E-SEN tag, then B-SEN occurs at the word next to the just previously occurring E-SEN tag.
4. If E-SEN tag is missing for a B-SEN tag, then E-SEN occurs at the word previous to the just next occurring B-SEN tag.

3.4 Identification of Recursiveness and Hierarchy

After the prediction of non-hierarchical items, we identify the recursiveness and hierarchy among them using a rule-based method. The rules of this method rely on two pieces of information, namely, left-indentation and bullet of the item segment. Bullet of an item segment can be a roman number, an English letter or a special symbol present at its start. Left-indentation for an item segment is the minimum x-coordinate of its first word (excluding bullet). We define a bullet's *predecessor* as the bullet that will occur just before it in the list of ordered bullets of corresponding bullet-style. e.g. predecessor of bullet (c) will be (b), predecessor for bullet 5. will be 4., predecessor for • will be •. We call a bullet to be of *start type* if it occurs first in the list of ordered bullets of corresponding bullet-style. e.g. (a), 1. and • are of *start type*. With these pieces of information we employ the *algorithm* described below. We maintain a set called *candidate lists* which stores the final *lists* and recursiveness/hierarchy among its item segments.

1. Sort all the items extracted from a financial document on the basis of their occurrence in the original text string. Jump to 2.
2. If *last-item* has been assigned call *last-item* as *first-item*, else choose the first item from the list of sorted item segments and call it *first-item*. Create a list with just *first-item* and call it *candidate list*. Jump to 3.
3. If no new items are left in sorted list of item segments exit the algorithm. Call the next new item in sorted list of item segments as *current-item*. If *candidate list* has just one element then jump to 4, else jump to 5.
4. If *current-item* has a bullet of *start type* mark it as child of *first-item* and jump to 3, else store *candidate list* in *candidate lists* and jump to 2. Before jumping, assign *current-item* to *last-item*.
5. If left-indentation of *current-item* and *last-item* are equal, jump to 6. If left-indentation of *current-item* is greater than that of *last-item* jump to 7. If left-indentation of *current-item* is less than that of *last-item* jump to 8.
6. If *last-item*'s bullet is *predecessor* of *current-item*'s bullet then mark *current-item* as child of *last-item*'s parent; store *current-item* in *candidate list* and jump to 3, else store *candidate list* in *candidate lists* and jump to 2. Before jumping, assign *current-item* to *last-item*.
7. If *current-item* has a bullet of *start type* mark it as child of *last-item*; store *current-item* in *candidate list* and jump to 3, else store *candidate list* in *candidate lists* and jump to 2. Before jumping, assign *current-item* to *last-item*.
8. Assign parent of *last-item* to *candidate-sibling*. Jump to 9.

9. If *candidate-sibling* has greater left-indentation than the *current-item*, assign parent of *candidate-sibling* to *candidate-sibling* and jump to 9, else jump to 10.
10. If *candidate-sibling*'s left-indentation is equal to that of *current-item* and *candidate-sibling*'s bullet is *predecessor* of *current-item*'s bullet then mark the parent of *current-item* with parent of *candidate-sibling*; store *current-item* in *candidate list* and jump to 3, otherwise store *candidate list* in *candidate lists* and jump to 2. Before jumping, assign *current-item* to *last-item*.

With above mentioned algorithm we achieve a set called *candidate lists* which captures parent-child relationships in initial item segments. If an item in the *candidate lists* has atleast one child, we change its end boundary to maximum of end boundaries of its children. The items at highest level (with no parents) correspond to *lists*. Items at one level lower correspond to *item1* and so on.

4 Experiments

We evaluated two neural architectures followed by rule-based post-processing. In this section, we describe the dataset, system settings, evaluation metrics, results and a brief error-analysis for our system.

4.1 Dataset

The dataset for FinSBD-2 shared task (English track) was provided in the form of JSON files. Each of the JSON files contained text and character-based coordinates extracted from a different financial document. The train and test set contained six and two files, respectively. Segment boundaries were provided in the form of character-based index pairs. Segment boundaries for the test dataset were provided after submission of our system's predictions. Table 1 summarises the statistics for the official FinSBD-2 dataset.

In Table 1, columns *Min.*, *Max.* and *Avg* correspond to minimum, maximum and average length (in number of words) of segments of a particular type. The column *#Count* denotes the number of occurrences of a certain segment-type in the dataset. The row *items (modified)* corresponds to the non-hierarchical and non-recursive list/item segments (corresponding to tags S-IT, B-IT and E-IT). Since the average length of any segment-type lies far away from the mean of its range, we can deduce that the length distribution of all segment-types is highly unbalanced. Additionally, the distribution of segment length for train and test dataset is quite different. On average, segments in the test dataset are longer as compared to the train dataset. This difference implies that the test set may be more complicated (more recursive lists/items and more complex sentences).

We define *coverage* as the percentage of unique words from the test set, which appear in the training set. *Coverage* gives us a fair idea of the number of unseen words/tokens, which the model sees at the testing stage. For FinSBD-2 shared task, the training dataset contains 7173 unique words in total. Whereas, the test dataset contains 4894 unique words in total. The vocabulary *coverage* for test set turns out to be 70.55%, implying that around 30% of the words in test set didn't appear in train set.

Train Dataset				
Segment-Type	Min.	Max.	Avg.	#Count
sentences	1	270	24.4	8070
lists	21	1520	149.88	249
items	1	456	32.9	1111
items (modified)	1	236	26.75	1360
Test Dataset				
Segment-Type	Min.	Max.	Avg.	#Count
sentences	1	391	29	2450
lists	15	1150	213.144	69
items	2	622	45.68	322
items (modified)	2	249	36.10	401

Table 1: Dataset (English) statistics for FinSBD-2020 shared task.

Hyper-parameter	BiLSTM-CRF	BERT
Max sequence length	300 (words)	500 (sub-words)
Lower case	False	False
Epochs	25 (max)	5
Batch-size	20	32
Learning rate	0.001	5e-5
Optimizer	Adam	Adam
Pre-trained model	-	bert-base-cased
Char-LSTM size	50	-
Word-LSTM size	200	-
Embedding dropout	0.3	-
Pre-trained embedding	GloVe	-

Table 2: Hyper-parameters employed in training neural models.

4.2 System Settings

In the first phase of our approach, we train two deep neural models, namely, BiLSTM-CRF and BERT. We train the BiLSTM-CRF model to a maximum of 25 epochs, along with an early-stopping strategy. With this strategy, we stop the training if the model does not show any improvements in F1-score on validation split for 500 continuous iterations. In addition to this, we also employ exponential moving averages of the trained parameters to achieve slightly better F1-scores on the validation split. For the pretrained word embeddings, we use GloVE[6] which are trained on large Common Crawl dataset and can effectively represent 84 billion cased tokens.

To train the BERT for our task, we fine-tune a pre-trained model, namely *bert-base-cased*. We have utilised *huggingface*'s BERT APIs[7] to train our model. Since the pre-trained BERT model was trained of a maximum input sequence length of 512 (including special tokens), we could not experiment with larger context window. We ran our experiments on single NVIDIA V100 GPU. It took around 20 and 30 minutes to train BERT and BiLSTM-CRF model, respectively. Table 2 summarises the hyper-parameters which we employed to

[6] http://nlp.stanford.edu/data/glove.840B.300d.zip
[7] https://huggingface.co/transformers/model_doc/bert.html

Class	BiLSTM–CRF			BERT		
	P	R	F1	P	R	F1
S-SEN	0.308	0.500	0.381	0.143	0.5	0.222
B-SEN	0.923	0.942	0.932	0.921	0.952	0.936
E-SEN	0.946	0.954	0.950	0.922	0.962	0.941
S-IT	-	-	-	-	-	-
B-IT	0.919	0.875	0.897	0.882	0.88	0.881
E-IT	0.877	0.873	0.875	0.88	0.875	0.878

Table 3: Scores on predictions from deep neural models on test set.

train both the models.

4.3 Evaluation Metrics

To extract a segment from the text, both begin and end boundaries should be predicted accurately. Hence, the evaluation metric should penalise the predictions in which either of the boundaries is incorrect. Consider that *P* and *T* represent the set of predicted and ground-truth boundary pairs (begin and end index pairs) for certain segment-type. Then, pairwise precision, recall and F1-score for the boundary prediction of the considered segment-type is defined as follows:

$$Precision = \frac{|P \bigcap T|}{|P|}$$

$$Recall = \frac{|P \bigcap T|}{|T|}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

FinSBD-2 is composed of two subtasks. Subtask-1 aims at evaluating the system's ability to predict and differentiate between sentences, lists and items accurately. Therefore, its official evaluation metric is mean of pairwise F1-score of sets *sentences*, *items* and *lists*. Whereas, subtask-2 aims at the prediction of the hierarchical layout of items, and hence its official evaluation metric is mean of pairwise F1-score of sets *item1*, *item2*, *item3* and *item4*.

4.4 Results and Error Analysis

FinSBD-2 shared task dataset had no separate validation split. In order to tune the hyper-parameters of our models, we chose one among the six financial documents in the train set for the validation purpose. Utilising this validation data, we tuned parameters such as input sequence length and hop-length for our final models. Table 3 and Table 4 summarise the results of our final models on the official test set.

In the first phase, we defined the problem as a sequence labelling task with seven output clases. Table 3 states the precision, recall and F1-score of our BiLSTM-CRF and BERT models in the first phase on our approach. Using the results table 3, we can conclude following points:

- Both the models predicted SEN tags more accurately than IT tags, presumably because sentences are more homogeneous and have little intra-class variations as compared to items.

Segment-Type	P	R	F1
sentences	0.923	0.938	0.931
lists	0.968	0.895	0.929
items	0.991	0.916	0.951
item1	0.907	0.905	0.906
item2	1.000	0.697	0.783
item3	-	-	-
item4	-	-	-
Subtask-1 (Macro F1) :			0.937
Subtask-2 (Macro F1) :			0.844

Table 4: Scores after final rule-based approach on test set.

- Both the models achieve better F1-score for the B-IT tag than the E-IT tag. This can be attributed to the fact that the beginning of item has more distinctive features, such as bullets/numbers, as compared to the ending of items.
- Similarly, models achieve better F1-score for E-SEN compared to B-SEN due to more reliable punctuation characters at the ending of the sentence.
- Class S-SEN has too few examples both in train and test set, and thus the numbers for this class do not convey much about models' performance.

For these two models, we also computed the weighted mean of F1-score of all the classes. BiLSTM-CRF and BERT achieved the final F1-score of 0.959 and 0.956. Thus, we conclude that both the models gave an almost similar performance on the task.

In the second phase, we utilised the outputs of first phase to identify the recursiveness and hierarchy. Table 4 states the precision, recall and F1-score of our rule-based approach. Our rule-based approach is susceptible to prediction errors, if the phase-1 fails to recall even a single item in some list. The falling F1-scores with the increased hierarchy (*lists* $>$ *item1* $>$ *item2*) further reflect on this fact. The Macro-F1 scores for subtask-1 and subtask-2 were ranked first among all the submissions.

5 Conclusion and Future Work

In this paper, we described our approach to tackle the FinSBD-2 shared task. Our approach was composed of two phases. In the first phase, we formulated the modified version of the task as a sequence labelling problem. We experimented with two neural models, namely, BiLSTM-CRF and BERT. In the second phase, we employed a rule-based approach to identify recursiveness and hierarchy among item segments from the first phase. We experimented with different hyper-parameter settings to tune our model. We submitted a system based on BiLSTM-CRF with an input length 300 as our final entry to the shared task. Our final system achieved the highest MEAN F1-score in the shared task. Our approach in this shared task should motivate research into the usage of visual information for sentence/list extraction from noisy PDF documents. In the future, we wish to explore the idea of multi-modality and end-to-end trainable deep neural models for this task.

References

[Agarwal *et al.*, 2005] Nishant Agarwal, K. Ford, and Mickhail N Shneider. Sentence boundary detection using a maxent classifier. 2005.

[Akita *et al.*, 2006] Yuya Akita, Masahiro Saikou, Hiroaki Nanjo, and Tatsuya Kawahara. Sentence boundary detection of spontaneous japanese using statistical language model and support vector machines. 01 2006.

[Azzi *et al.*, 2019] Abderrahim Ait Azzi, Houda Bouamor, and Sira Ferradans. The FinSBD-2019 shared task: Sentence boundary detection in PDF noisy text in the financial domain. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, pages 74–80, Macao, China, August 2019.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Du and Huang, 2019] Jinhua Du and Yan Huang. Aig investments.ai at the finsbd task: Sentence boundary detection through sequence labelling and bert fine-tuning. 2019.

[Evang *et al.*, 2013] Kilian Evang, Valerio Basile, G. Chrupała, and Johan Bos. Elephant: Sequence labeling for word and sentence segmentation. pages 1422–1426, 01 2013.

[Gillick, 2009] Dan Gillick. Sentence boundary detection and the problem with the U.S. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 241–244, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. volume 9, page 1735–1780, Cambridge, MA, USA, November 1997. MIT Press.

[Kiss and Strunk, 2006] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. volume 32, page 485–525, Cambridge, MA, USA, December 2006. MIT Press.

[Kreuzthaler and Schulz, 2015] Markus Kreuzthaler and Stefan Schulz. Detection of sentence boundaries and abbreviations in clinical narratives. volume 15 Suppl 2, page S4, 06 2015.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Peters *et al.*, 2018] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Polanyi *et al.*, 2004] Livia Polanyi, Chris Culy, Martin van den Berg, Gian Lorenzo Thione, and David Ahn. A rule based approach to discourse parsing. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*, pages 108–117, Cambridge, Massachusetts, USA, April 30 - May 1 2004. Association for Computational Linguistics.

[Read *et al.*, 2012] Jonathon Read, Rebecca Dridan, Stephan Oepen, and Lars Jørgen Solberg. Sentence boundary detection: A long solved problem? In *Proceedings of COLING 2012: Posters*, pages 985–994, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

[Reynar and Ratnaparkhi, 1997] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, page 16–19, USA, 1997. Association for Computational Linguistics.

[Schuster and Paliwal, 1997] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. volume 45, pages 2673–2681, 1997.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. volume 15, page 1929–1958. JMLR.org, January 2014.

[Straková *et al.*, 2019] Jana Straková, Milan Straka, and Jan Hajic. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy, July 2019. Association for Computational Linguistics.

[Sutton and McCallum, 2012] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373, April 2012.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.