On-Device detection of sentence completion for voice assistants with low-memory footprint

Rahul Kumar Vijeta Gour Chandan Pandey Godawari Sudhakar Rao
Priyadarshini Pai Anmol Bhasin Ranjan Samal
Samsung R&D Institute India, Bangalore
{rahul.k4, vijeta.gour, chandan.p, g.sudhakar, priya.pai,
anmol.bhasin, ranjan.samal}@samsung.com

Abstract

Sentence completion detection (SCD) is an important task for various downstream Natural Language Processing (NLP) based applications. For NLP based applications, which use the Automatic Speech Recognition (ASR) from third parties as a service, SCD is essential to prevent unnecessary processing. Conventional approaches for SCD operate within the confines of sentence boundary detection using language models or sentence end detection using speech and text features. These have limitations in terms of relevant available data for training, performance within the memory and latency constraints, and the generalizability across voice assistant domains. In this paper, we propose a novel sentence completion detection method with low memory footprint for On-Device applications. We explore various sequence-level and sentence-level experiments using state-of-the-art Bi-LSTM and BERT based models for English language.

1 Introduction

Voice-intelligence enabled devices have tremendous potential in providing near natural human behavior based product experience to the users (Dellaert et al. 2020). Such a potential has primarily stemmed from Artificial Intelligence based mimicking of speech recognition, question answering, dialogues, conversations, and command-induced actions. In order to meet user expectations and cater towards product usage satisfaction, knowing when the question, reply or command is complete, i.e., sentence completion detection, is a crucial task.

Voice assistants like Google Assistant, Alexa, Cortana, Siri and Bixby etc. are becoming very popular in modern world. These systems rely on the text, predicted by a streaming speech recognition (ASR) system. Streaming ASR produces text continuously. It is computationally efficient to execute downstream NLP tasks only when a complete sentence is found in the text. This makes SCD a crucial need in voice assistant system. For a partial text received from ASR, the application can wait relatively longer than a complete sentence. Various downstream NLP applications such as unsupervised dependency parsing, ASR transcript readability, accurate information retrieval, and summarization can benefit from SCD.

Detection of sentence completion, has been widely attempted on speech (Hasan, 2014) and text (Azzi, 2019) using sentence boundary detection (SBD) (Sanchez, 2019), end-of-utterance detection (Treviso, 2017), sentence end detection (SED) (Hasan, 2015) models. The techniques evolved from rule-based using handcrafted heuristics (Wang, 2004), to machine learning and more recently, deep learning (Schweter, 2019) based methods. State-of-the-art deep learning architectures reported for SBD include Bi-LSTM CRF (Du, 2019) , BERT (Du, 2019) etc. techniques. On various test datasets, we found their performance highly promising. Upon exploring at further depth, we found that these models have certain limitations with regards to their size, on device platform compatibility and system coupling

Proceedings of the 17th International Conference on Natural Language Processing, pages 384–392
Patna, India, December 18 - 21, 2020.

for on-device deployment. These limitations are briefly described below:

- Size of the models were too big for deployment on memory constrained devices such as mobile phones and smart televisions. MobileBERT[1] takes 100.5Mb with 74ms latency.
- Broad-spectrum conversation data availability, which is representative of a wide range of domains and follows SCD policies as mentioned in Section 4.2
- State-of-the-art architectures reported for SBD lack in ease of modelling with modifications, within the Tensor-flow lite environment.
- Decoupling of SBD models that are a part of bigger system like ASR is challenging and not readily applicable.
- SBD makes use of punctuations and case sensitive information which are missing from immediate ASR output.

We propose that Sentence Completion Detection (SCD) can be achieved by token-level and sentence-level inferencing.

In this research, we explore both token-level and sentence-level inferencing with state-of-the-art language models within on-device deployment constraints. We delve into the tailoring of data, completion detection policies (Section 4.2, SCD Policies), embedding size optimization for achieving a light-weight SCD model that can work on a wide range of domains in memory-constrained environments.

2 Related work

Recent SCD and SBD works are primarily useful for legal text, long lectures, pdf documents etc. Consequently, the datasets used for relevant work included clean texts such as WSJ corpus and Brown Corpus (Francis, 1979), noisy unstructured texts generated from PDFs (Azzi, 2019), (Tian, 2019), lecture (Hasan, 2014) and ASR transcripts (Treviso, 2017), (Rehbein, 2020). For training a model that is suitable for the multi-domain voice assistant, we could not find broad-spectrum domain data focused on commands.

Further, we felt that, essentially, a shift of emphasis from formal, edited text towards more spontaneous language samples which represent ASR output is required Conventional language models are trained on long structured sentences leading to large memory footprints that cannot be supported for fast on-device applications. Various techniques have been reported for downsizing, such as quantization, modifications in vocabulary, truncating input etc.

We expand our work based on modifications in state-of-the-art architectures and extensive custom training with custom loss on multi-domain conversation data. We also experimented upon Tensorflow Lite post training quantization. We primarily looked at Bi-LSTM and BERT architectures as described below.

3 Model

We defined our SCD models in two categories:

- Sequence-based
- Sentence-based

For each of these categories, we explored selected state-of-the-art Bi-LSTM based model and a BERT-based model as described below. Bi-LSTM is a sequence processing model that consists of two LSTMs, one taking the input in a forward direction, and the other in a backwards direction. BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text.

3.1 Bi-LSTM and Attention based model for sequence prediction [1.a]

We convert the text input sentence to a sequence of tokens by splitting based on spaces. The model (shown in Figure 1.a) contains an embedding layer which gets trained along with the model and generates vectors for the tokens present in the sentence. For each of the generated tokens feature labelling is done as either '0' or '1' based on the method explained in 4.3. Example token features for a sentence is shown below:

Utterance: "create an event at 5"

Create	An	Event	At	number
0	0	1	0	1

[1]https://www.tensorflow.org/lite/models/bert_qa/overview

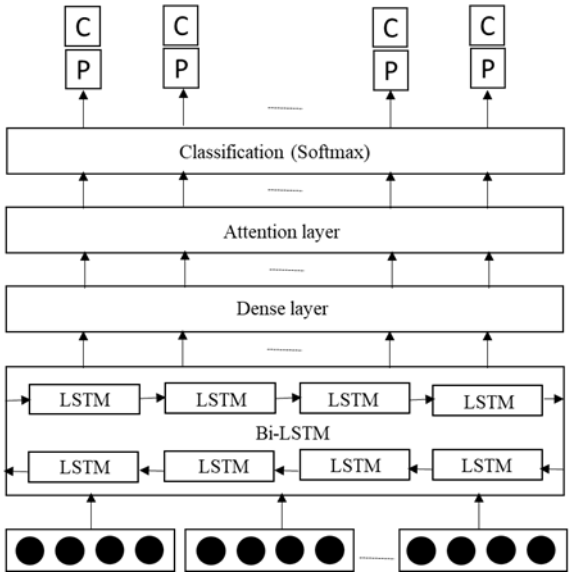

Figure 1.a Bi-LSTM and Attention based model for sequence prediction

Tokens of the input sentence are converted into token IDs. The sentence length used for prediction is kept at 25 tokens, a considerable assumption for sentences in voice assistant based systems. If the sentence is larger than 25 tokens then only the last 25 tokens are used for prediction. So, the resultant input dimensionality becomes 25x1. This resultant vector is then passed to embedding layer which converts it into 25x100 vector followed by a Bi-Directional LSTM (Hochreiter, 1997), (Graves, 2005). The output of the Bi-LSTM layer is passed to dense layer in a time distributed manner. The output of dense layer is passed through an attention layer followed by soft-max activation which predicts a label for every token as shown in Figure 1.a. The Loss is calculated using Equation (1) and the inference is done using Equation (2).

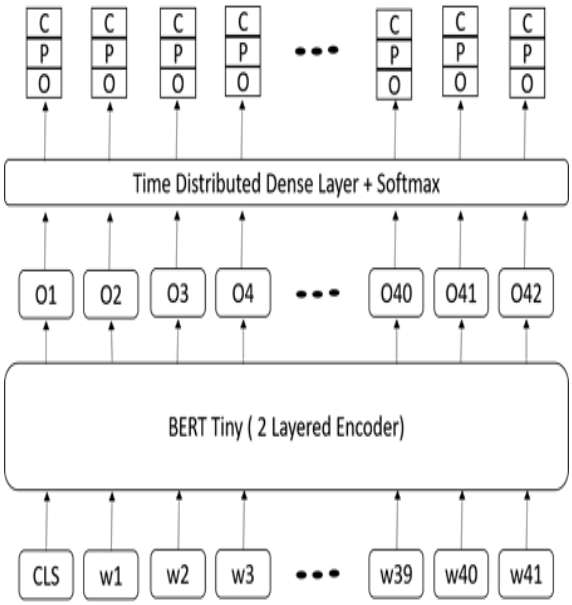

Figure 1.b BERT Tiny based model for sequence prediction

3.2 BERT Tiny based model for sequence prediction [1.b]

We convert the text input sentence to a sequence of tokens using Sentence piece tokenizer (Kudo, 2018). For each of the generated tokens feature labeling is done. The tokens generated are marked as '0' or '1' based on method explained in 4.3.
The sequence length used for these models is 42 tokens. So, a sequence of 42 tokens are passed through a pre-trained BERT (Turc, 2019) Tiny model which has got 2 encoder layers with 128 hidden states. The output of the BERT Tiny layer is passed through a dense layer with soft-max activation which predicts a label for every token. The Loss is calculated using the Equation (1) and the inference is done using Equation (2). Figure 1.b below shows the model architecture.

3.3 Bi-LSTM and Attentions based model with hybrid (word + character) embedding for sequence prediction [1.c]

The model 1.a is extended to improve the model performance on sentences containing out of vocab words. To meet that objective we introduced a hybrid embedding strategy. For every token present in the sentence we generate its embedding using char embedding in conjunction with word embedding. Every word is split into characters and then converted into IDs. The maximum length of a word is considered as 10 and first ten characters are taken if the length exceeds the maximum length. The IDs are fed into an embedding layer followed by LSTM sequence. The 50 dimension sequence output of LSTM is concatenated with 100 dimension vector of word embedding. The combined input is passed through a spatial dropout followed by Bi-LSTM. The output of the Bi-LSTM layer is passed to dense layer in a time distributed manner. The output of dense layer is passed through an attention layer followed by soft-max activation which predicts a label for every token. The Loss is calculated using Equation (1) and the

inference is done using Equation (2), which uses soft-max score of the last token.

$$L = l_{partial} + W * l_{complete} \quad (1)$$

Where L is total loss and $l_{partial}$ and $l_{complete}$ are losses on partial and complete tokens in the sentence respectively. $l_{partial}$ is calculated as the sum of categorical cross entropy losses of all the partial tokens. Similarly $l_{complete}$ is calculated as the sum of categorical cross entropy losses of all the partial tokens. W is the ratio of total count of partial tokens to the total count of complete tokens in the training data.

$$Pred = argmax(softmax(Last\_Token)) \quad (2)$$

3.4 Bi-LSTM and Attention based model for sentence classification [2.a]

The input sequence is processed in a similar way as mentioned in Section 3.1. The model contains an embedding layer which gets trained along with the model and generates embedding vectors for the tokens present in the sentence. For every sentence one label is assigned. For partial sentences label '0' is assigned and for complete sentences label '1' is assigned. Each token is converted into IDs. The sentence length used for prediction is kept at 25 tokens. If the sentence is larger than 25 tokens then only the last 25 tokens are used for prediction. So, the resultant input dimensionality becomes 25x1. This resultant vector is then passed to an embedding layer which converts it into 25x100 vector followed by a Bi-Directional LSTM (Hochreiter, 1997), (Graves, 2005). The output of the Bi-LSTM layer is passed to dense layer followed by attention layer followed by soft-max activation which predicts the label for the entire sentence. Figure 2.a shows the model architecture.

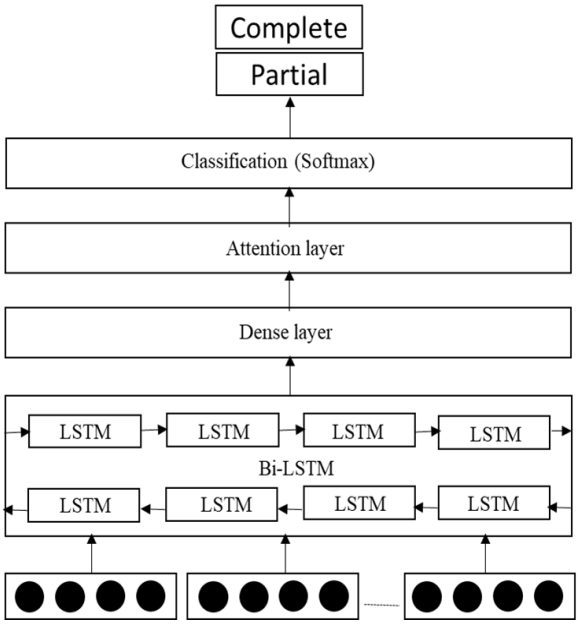

Figure 2.a Bi-LSTM and Attention based model for sentence classification

3.5 BERT based model for sentence classification [2.b]

The sentence tokenization part is similar to the model 1.b. However, this model treats this task as classification. Each sentence is labelled as '1' for complete and '0' for partial. The maximum sequence length used as input is 42 tokens. The tokenized utterance is passed through BERT Tiny model which has got 2 encoder layers with 128 hidden state size. The output of the CLS token of BERT Tiny layer is passed through a dense layer with soft-max activation which predicts a label for the sentence. Figure 2.b shows the model architecture.

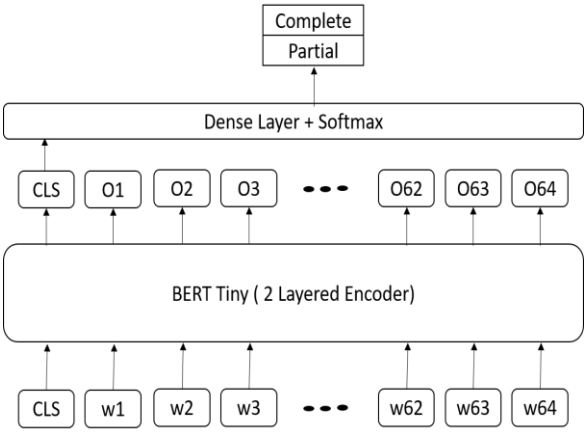

Figure 2.b BERT based model for sentence classification

4 Experimental Setup

4.1 Datasets

For exploring various SCD modelling architectures, we prepared an in-house dataset containing partial and complete sentences representing various domains such as phone call, message, contacts, reminder, maps, accessibility, calculator, clock, open domain, settings, apps, etc. This dataset comprised of sentences of varied

lengths as shown in Table 3. In addition to this, based on our analysis of various available datasets, we selected SNIPS (Coucke, 2018) for gaining insights into model generalizability. The Snips dataset on the other hand is collected from Snips Personal Voice Assistant, spanning 14484 multiple domain utterances. We split these utterances into training, validation and test datasets. From the original utterances present in the dataset we generated partial and complete utterances by generating pre-fixes. We omitted one word pre-fixes from the newly generated utterances.

Dataset	SNIPS	In-House
Train Data	13084	300000
Test Data	700	48000
Validation Data	700	20000
Vocabulary Size	11241	72001
Max Sentence Length	36	89

Table 1. Dataset details

A summary of the training and test data is provided in 3 tables. Table 1 contains the details of the original dataset. Table 2 contains the generated utterances details of SNIPS dataset and Table 3 contains the details of the generated utterances of in-house dataset.

Dataset	SNIPS		
	Total	Partial	Complete
Train	22213	9353	12860
Dev	1296	600	696
Test	1327	628	699

Table 2. Generated utterances details of SNIPS

Dataset	In-House		
	Total	Partial	Complete
Train	2702376	1136745	1565631
Dev	50000	20000	30000
Test	79899	17433	62466

Table 3. Generated utterances details of In-House dataset

4.2 SCD Policies

We aim to make this model highly suited to understand the NLU component in voice assistants for a variety of downstream applications. In each of these aspects, a comprehensive policy formation based on underlying information in relevant data is very important. This, in fact, becomes the key driving factor in determining user experience of the voice assistants. Based on our analysis and understanding, we outlined two main focus areas:

- Intent clarity
- Catchall phrases

We define all those sentences that can elicit an actionable response from the downstream target block in the voice assistant as complete. For example –

"*Create an event*"

Further, sentences that end in open titles are extremely dicey to handle. Any abrupt completion would result in unsatisfactory experience at user's end as there could be multiple complete suffixes for a given sentence. For example –

"*Create a reminder to buy milk*"

In such a scenario, it is difficult to predict if the user intends to continue after "buy milk" with "from a nearby shop". Consequently, strong allocation of sentences with catch-all phrases into partial or complete purely based on semantic understanding will not yield us desirable results. We propose to handle them separately by adopting system-specific suitable behavior.

4.3 Data preprocessing

Before passing a sentence to the model, we preprocessed it .Firstly, we removed punctuations to make the input sentence similar to ASR output. Secondly, we added acronym expansion, and replaced integers with the term "number in order to reduce Out Of Vocabulary (OOV) words. Lastly we removed polite phrases to reduce sentence length. We selected 25 token length for Bilstm models and 42 for BERT models as 98% of the tokenized sentences lengths are covered (Figure 3). This reduces the inference time.

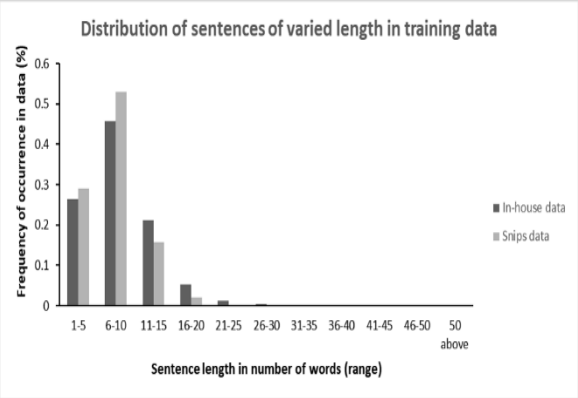

Figure 3 Occurrences of sentences of varied lengths in training data

1. For Bi-LSTM Sequence Models

The tokens present in the sentence were assigned a tag of 0 or 1 depending on whether the intermediate sentence forming up to the current token was already present in the dataset as complete. This was done so as the model sees the sentence as complete even if it is a part of another sentence. If this is not taken care of then the model gets confused for sub-sentences which are complete but also part of a longer sentence. Every intermediate substring of the sentence was checked if it was already present as a complete sentence and the end of the substring was marked as 1.

Utterance: "create a reminder"

create	a	reminder
0	0	1

Utterance: "create a reminder to buy"

create	a	reminder	to	buy
0	0	1	0	0

2. For BERT Sequence Model

The tokens generated by the Sentence Piece Tokenizer were tagged according to strategy mentioned in section 4.3(1). The root token and the subsequent token for the word are both given the same tag.

Utterance: "dont be so judgmental"

don	##t	be	so	judgment	##al
0	0	0	0	1	1

4.4 Training and inferencing

We developed all the models described in section 3 using Tensorflow2.0 as it has a wide collection of workflows, multiple language support and deployment flexibility. We limited the size of the vocabulary of the models- 1.a, 1.c and 2.a to 40000 for in-house dataset and 11000 for SNIPS dataset. BERT tiny based models are initialized with pre trained weights and fine-tuned using recommended hyperparameters (batch size: 32, learning rate: 3e-5, epochs: 3-5) during training.

We selected the best model based on best average F1 score for both the classes of prediction (Partial and Complete) as well as memory foot print.

$$F_1 = \frac{2*Precision*Recall}{Precision+Recall} \quad (3)$$

Once the models were ready, we converted them to TensorFlow Lite with post training quantization for on-device deployment. TensorFlow Lite is designed for efficient model execution on memory constrained devices such as mobile phones. Some of this efficiency reportedly comes from the use of a special storage format that reduces model file size and relevant optimizations that have very less impact on the accuracy.

4.5 Testing

We tested our models on-device on three test datasets as described in 4.1 and report model performances at various inferencing levels, architecture levels and sentence complete detection levels. Further, we also checked the latency and memory footprint to evaluate the feasibility of using such a model on mobile devices.

5 Results and discussion

In the following, we present the results of our performance assessment on various models, *i.e.*, both token-level and sentence-level inferencing using Bi-LSTM and BERT techniques on the three test datasets.

5.1 Performance assessment of various models

Among the BERT and Bi-LSTM models, as shown in Table 4. The best performance is achieved by BERT-Tiny sequence based SCD model. On an average, on both the datasets, it is able to achieve an overall F1-score of 90.95%. The next best

performing model was with Bi-LSTM, attention and word embedding for sentence classification.

Model	SNIPS		In-House Data	
	C	P	C	P
1.a	84.3	77.3	95.5	90.0
1.b	88.0	84.3	96.8	92.3
1.c	82.9	75.2	87.7	88.4
2.a	87.5	83.8	95.69	86.66
2.b	87.8	84.5	93.05	90.19

Table 4. Comparison of F1 scores for partial and complete utterances

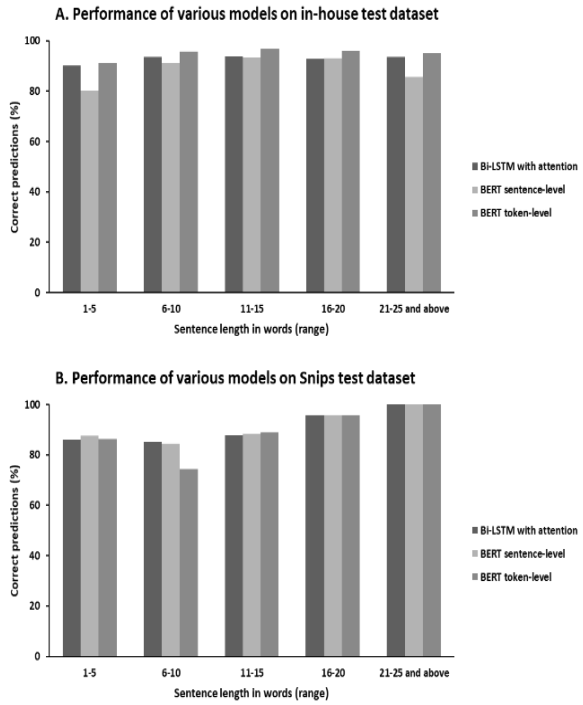

Figure 4. Performance assessment of various models on sentences of different lengths. (A) Results on in-house test data. (B) Results on Snips test data.

Analysis on the results suggests that the sentence completion detection is relatively challenging on very short length (1-5 word) sentences (Figure 4). This might be the reason behind slightly decreased prediction performance (86.87% correct predictions on an average across in-house and Snips test datasets) as compared to sentences with lengths greater than 5 words. We saw most consistent performance on sentence length of 16-20 words at 94.82% average correct predictions across datasets and above 92.62% correct predictions across all model architectures. Possibly a clearer understanding of partial and complete can be achieved by the model in this sentence length range. Further, although there were very few sentences with length beyond 21 words, the models were able to learn completion detection and predict 95.71 % of test data correctly.

5.2 Analytical insights into partial complete sentences prediction

Among the partial and complete sentences tested uisng BERT sequence-based model, we observed that the F1 score for complete sentences was better than partial sentences. We also noticed the same trend in majority of the cases. This is a promising scenario for user experience, where if sentence completion prediction is better on complete sentences, the wait time can be drastically shortened. Consequently, the user experience is also likely to improve.

5.3 Analytical insights into predictions on various test datasets

We observed that in general, the models trained and tested on in-house data performed better than the models trained and tested on SNIPS. The OOV failures were observed less in Word+Char Bi-lstm sequence model and Bert Tiny sequence model.
The sequence models were able to generalize the data better than the sentence models due to the subsequence learning mentioned in Section 4.3.

5.4 Memory footprint of various models

The memory footprint of each of the models developed is given in Table 5. Low memory footprint enables it to be used in memory constrained environment.

Model	Memory (in MB)	
	SNIPS	In-House Data
1.a	6.1	6.8
1.b	4.5	4.5
1.c	4	4.8
2.a	6.1	6.8
2.b	4.5	4.5

Table 5. Comparison of model memory footprint

5.5 On Device Latency of various models

The On Device latency for each of the developed models is mentioned in Table 6. The devices used for testing were Android devices with SDK version 10. The solution works in real time due to low latency.

Model	Latency(in ms)
1.a	22-34
1.b	20-30
1.c	25-37
2.a	15-25
2.b	15-25

Table 6. Comparison of On-device run time latency

6 Conclusion

Sentence completion detection is important for various NLP applications on voice assistant enabled devices. The existing solutions do not cater to the challenges present in conversational ASR output data and are not optimized to work on memory and latency constrained devices. In this paper, we tailored state-of-the-art Bi-LSTM and BERT models for on-device solutions. Fine-tuned BERT Tiny sequence model [1.b] outperforms all other models on both the datasets. Our experimental results show that the mentioned solutions are highly promising for various real-time on-device applications.

References

Du, J., Huang, Y. and Moilanen, K., 2019. IG Investments. AI at the FinSBD Task: Sentence Boundary Detection through Sequence Labelling and BERT Fine-tuning. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 81-87).

Azzi, A.A., Bouamor, H. and Ferradans, S., 2019. 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* (pp. 74-80).

Sanchez, G., 2019, June. Sentence boundary detection in legal text. In *Proceedings of the Natural Legal Language Processing Workshop 2019* (pp. 31-38).

Xu, C., Xie, L. and Xiao, X., 2018. A bidirectional lstm approach with word embeddings for sentence boundary detection. *Journal of Signal Processing Systems*, *90*(7), pp.1063-1075.

Treviso, M.V., Shulby, C.D. and Aluisio, S.M., 2017. Evaluating word embeddings for sentence boundary detection in speech transcripts. *arXiv preprint arXiv:1708.04704*.

Che, X., Luo, S., Yang, H. and Meinel, C., 2016. Sentence Boundary Detection Based on Parallel Lexical and Acoustic Models. In *Interspeech* (pp. 2528-2532).

Ho, T.N., Chong, T.Y. and Chng, E.S., 2016, March. Improving efficiency of sentence boundary detection by feature selection. In *Asian Conference on Intelligent Information and Database Systems* (pp. 594-603). Springer, Berlin, Heidelberg.

Schweter, S. and Ahmed, S., 2019. Deep-EOS: General-Purpose Neural Networks for Sentence Boundary Detection. In *KONVENS*.

Fatima, M. and Mueller, M.C., 2019. HITS-SBD at the FinSBD Task: Machine Learning vs. Rule-based Sentence Boundary Detection. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 115-121).

Au, W., Chong, B., Azzi, A.A. and Valsamou-Stanislawski, D., 2020, July. FinSBD-2020: The 2nd Shared Task on Sentence Boundary Detection in Unstructured Text in the Financial Domain. In *Proceedings of the Second Workshop on Financial Technology and Natural Language Processing* (pp. 47-54).

Mathew, D. and Guggilla, C., 2019. Ai_blues at finsbd shared task: Crf-based sentence boundary detection in pdf noisy text in the financial domain. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 130-136).

Tian, K. and Peng, Z.J., 2019. aiai at finsbd task: Sentence boundary detection in noisy texts from financial documents using deep attention model. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 88-92).

Hirano, M., Sakaji, H., Izumi, K. and Matsushima, H., 2019. mhirano at the finsbd task: Pointwise prediction based on multi-layer perceptron for sentence boundary detection. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing* (pp. 102-107).

Zhang, R. and Zhang, C., 2020, July. Dynamic Sentence Boundary Detection for Simultaneous Translation. In *Proceedings of the First Workshop on Automatic Simultaneous Translation* (pp. 1-9).

Rehbein, I., Ruppenhofer, J. and Schmidt, T., 2020. Improving sentence boundary detection for spoken language transcripts.

Le, T.A., 2020, January. Sequence Labeling Approach to the Task of Sentence Boundary Detection. In *Proceedings of the 4th International Conference on Machine Learning and Soft Computing* (pp. 144-148).

Wang, D. and Narayanan, S.S., 2004, May. A multi-pass linear fold algorithm for sentence boundary detection using prosodic cues. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 1, pp. I-525). IEEE.

Oba, T., Hori, T. and Nakamura, A., 2006. Sentence boundary detection using sequential dependency analysis combined with crf-based chunking. In *Ninth International Conference on Spoken Language Processing*.

Hasan, M., Doddipatla, R. and Hain, T., 2014. Multi-pass sentence-end detection of lecture speech. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Hasan, M., Doddipatla, R. and Hain, T., 2015. Noise-matched training of CRF based sentence end detection models. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Dellaert, B.G., Shu, S.B., Arentze, T.A., Baker, T., Diehl, K., Donkers, B., Fast, N.J., Häubl, G., Johnson, H., Karmarkar, U.R. and Oppewal, H., 2020. Consumer decisions with artificially intelligent voice assistants. Marketing Letters, pp.1-13.

Kudo, T. and Richardson, J., 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv: 1808.06226.*

Turc, Iulia and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2*.

Francis, W.N. and Kucera, H., 1979. Brown corpus manual. *Letters to the Editor*, *5*(2), p.7.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.

Graves, A. and Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, *18*(5-6), pp.602-610.

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T. and Primet, M., 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.