Information Extraction of Air-Traffic Control Instructions via Pre-trained Models

Xuan Wang(✉), Hui Ding, Qiucheng Xu, and Zeyuan Liu

State Key Laboratory of Air Traffic Management System and Technology, The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China
wangxuan10@cetc.com.cn

Abstract. The air traffic controllers always use air-traffic control instructions (ATC instructions) to command aircrafts. The ATC instruction consists of some situation mentions such as flight number, status, and target location etc. The deep-learning based approach can extract such information for situation awareness. In practice, it is difficult to prepare huge amount of labelled ATC instructions for training the deep-learning model due to expensive costs of handcraft annotations. The large scale pre-trained model (PTMs) can solve this problem by "pre-training" and "fine-tuning". This paper proposes: 1) pre-trained models to extract information from few scale ATC instructions; 2) the probing task to find which layer of model achieves the best performance of information extraction task.

Keywords: ATC instructions · Named entity recognition · Pre-trained model · Fine-tuning · Probing task

1 Introduction

The radiotelephony communication is a kind of voice communication mode between air-traffic controller and aircraft. The air-traffic controllers always use ATC instructions in radiotelephony communication to command aircrafts. It is important to use the correct ATC instructions during communication because incorrect or ambiguous ATC instructions would cause aviation unsafe incidents. The ATC instruction always consists of some situation mentions such as flight number, status, location etc. Analysing these mentions can help ATM systems to avoid the potential accidents. However, the practical ATC instruction occurs in text style and ATM systems cannot apply directly. Named Entity Recognition (NER) is kind of method to extract the situation information in ATC instructions. Then the ATM systems can predict some potential accidents with such situation information and other air-traffic management systems' information such as ADS-B, A-CDM, ASMGCS, etc.

NER [1] aims to recognize mentions of rigid designators from text belonging to predefined semantic types such as person, location, organization etc. The input of NER system is sequence of tokens, and its output is sequence of corresponding labels. NER

Supported by China National Key R&D Program (No. 2020YFB1600100).

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
Q. Liang et al. (Eds.): Artificial Intelligence in China, LNEE 854, pp. 82–91, 2022.
https://doi.org/10.1007/978-981-16-9423-3_11

plays an essential role in aviation text understanding. There are four main streams of techniques applied in NER: 1) rule-based approaches, 2) feature-based supervised machine learning based approaches, 3) supervised deep-learning based approaches, 4) pre-trained model based approaches [2].

Applying supervised machine learning approaches, NER is regard as sequence labelling task. Some machine learning algorithms are proposed such as hidden Markov model (HMM) [3] and conditional random field (CRF) [4]. These approaches train the model by a large annotated corpus, memorize lists of entities, and creates disambiguation rules based on transition probability. Feature engineering is critical in supervised machine learning approaches. HMM and CRF needs feature vector representation to be inputs.

Deep learning composed of multiple processing layers to learn useful representations of data automatically. Compared with shallow models of machine learning, the deep-learning models learn complex and intricate features from data. The traditional deep-learning models such as Long Short-Term Model (LSTM) is able to capture the non-linear mappings between input and output. Hammerton et al. [5] attempted NER with LSTM, which can capture long distance dependence of input text. Lample et al. [6] proposed LSTM-CRF for NER, where BiLSTM layer computes the deep contextual word embedding and CRF layer outputs tagging sequence via viterbi decoding. Although applying whole sentence information to infer the global optimum sequence of tagging, CRF layer cannot explicitly capture dependency between any two labels due to Markov assumptions. In order to break the limitation, Cui et al. [7] proposed the hierarchically-refined label attention network to replace CRF in inference layer. In Cui's model, the multi-headed self-attention [8] can help NER models learn dependency between any labels. The convolutional neural networks (CNN) are also applied in NER models due to its ability of modelling character-level information, such as LSTM-CNNs [9] and BiLSTM-CNNs-CRF [10]. Deep-learning based approach is still a kind of supervised learning approach, which needs large scale labelled data for training. However, it is difficult to satisfy such requirement of training data due to expensive costs of handcraft annotations.

The pre-trained model based pre-trained models are effective for low-resource NER via fine-tuning. The PTMs learn language knowledge from large amount of unlabeled data in pre-trained stage, then use few scales labelled data of downstream tasks to train the models in fine-tuning stage [11] and let model adapt to any downstream tasks. Radford et al. [12] proposed Generative Pre-trained Transformer (GPT) for language generating task. Devlin et al. [13] proposed Bidirectional Encoder Representations from Transformers (BERT), which encodes tokens by joint conditioning on both left and right context in all layers.

In this paper, the PTMs-CRF model is presented to extract the situation information of ATC instructions. The pre-trained model is defined to be embedding layer and CRF model to be inference layer. There are two contributions in the paper: 1) choose the pre-trained models to solve low-resource problem; 2) apply probing task to search which layer of pre-trained models can obtain the best performance. Finally, we verify PTMs-CRF on practical ATC instructions datasets and discuss the experiment results.

2 Related Work

Pre-trained Models. In the past decade, the technology of pre-trained models has developed rapidly. The technology development is divided into two generations. First generation is to learn pre-trained word embedding, PTMs learn words representation from large scale unlabeled data. Second generation is pre-trained contextual encoder, PTMs learn contextual word embedding by building language model.

In order to capture syntactic and semantic information of words, word embedding via neural network language model is proposed [14]. Mikolov et al. [15] proposed Continuous Bags-of-Words (CBOW) and Ship-Gram (SG) models to represent words. Word2vec can learn high-quality word embedding to capture the latent sematic similarities among words. Glove et al. [16] obtains the global vectors for word representation by word-word cooccurrence matrix. Although improving the performance in different NLP tasks, the pre-trained word embeddings are context-independent and do not apply information of the entire input sentence for representing words.

The pre-trained contextual encoder can represent the word semantic depending on its context. Peters et al. [17] proposed a type of deep contextualized word representation by using stacked BiLSTM and pre-trained on a large text corpus. ULMFiT [18] attempted to fine-tune pre-trained language model for text classification. After Transformer was proposed, the vast majority of PTMs are made up of its encoder or decoder, such as GPT and BERT. Moreover, some advanced models were proposed to improve performance such as Roberta [19], ALBERT [20], XLNET [21]. Compared with pre-trained word embedding, these PTMs can learn more knowledge from large scale text corpora by self-supervised training.

Probing Task. The PTMs are hierarchical structure, which are stacked by multi-layer transformer. The probing task is an approach to find what knowledge do each layer of PTMs learn. Jawahar et al. [22] proposed that BERT's lower layers capture the phrase-level information, intermediate layers learn the linguistic information and higher layers encode the sematic information of. Wallace et al. [23] apply probing task to search BERT's representing ability for numbers, which shows that BERT only uses sub-word units and is less exact.

3 PTMs-CRF Model

This section will describe the PTMs-CRF model. The architecture of PTMs-CRF model includes embedding layer and inference layer, which is similar to the one presented by Lample et al. [6]. In this paper, BERT and its advanced versions are applied in model's embedding layer due to their excellent natural language encoding performance.

3.1 BERT Family

BERT has some advanced versions: RoBERTa trains longer with bigger batches and over more data; and ALBERT, a lite version of BERT.

3.1.1 Original BERT

BERT is a stack of Transformer encoder layers which consist of multi-headed self-attention, fully-connected, layer normalization and residual connection. Every token of input is defined as query, key and value vectors, each head self-attention creates the weighted representation by them. Then fully-connected layer is used to combine the outputs of all heads in the same layer. The residual connection makes model deeper, and layer normalization can improve the performance by reducing gradient disappearance and explosion (Fig. 1).

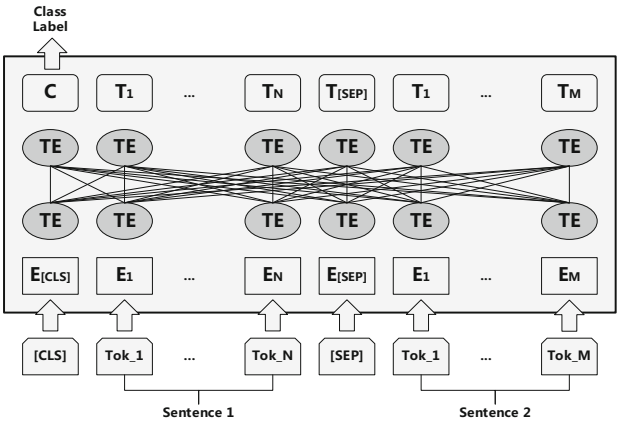

Fig. 1. The architecture of BERT

The traditional workflow for BERT includes two stages, first stage is pre-training by large scale text corpora with two semi-supervised tasks: masked language modeling (MLM) and next sentence prediction (NSP); the second stage is fine-tuning for downstream applications by supervised training via few scales labelled data.

For every word of input sequence, BERT first tokenize it into word-pieces. The final input embeddings are combined by token embedding, position embedding and segment embedding. The special token [CLS] represents the whole sentence and [SEP] is used for separating input segments.

Define the input sequence of ATC instruction as $x_1, x_2, \ldots, x_n$ and the output sequence as $y_1, y_2, \ldots, y_n$, where N is the instruction's length, x_i means ith token of input sequence and y_i means ith label. Denote L to be number of layers, and H to be hidden size of model. Then input embeddings of BERT is computed as:

$$E_i = W_e x_i + W_p + W_s \tag{3.1}$$

where $E_i \in R^{H \times 1}$ represents the ith token, equals to the sum of token embedding $W_e x_i$, position embedding W_p and segment embedding W_s.

The output of BERT can be computed by following equations:

$$h_i^{l+1} = transformer\_encoder(h_i^l), \quad \text{where } h_i^0 = E_i, y_i = h_i^L \tag{3.2}$$

The *transformer_encoder* denotes the encoder of Transformer and A is the number of self-attention heads in one layer of encoder. The h_i^l is hidden state of the lth layer. BERT has two model sizes: BERT-base (L = 12, H = 768, A = 12, total parameters = 110 M) and BERT-large (L = 24, H = 1024, A = 16, total parameters = 340 M).

3.1.2 RoBERTa

The performance of BERT is not excellent because of significantly undertrained when pre-training. In order to develop it, four modifications were applied: 1) training the model longer with bigger batches and over more data; 2) using only masked language modelling (MLM) pre-training task; 3) training on longer sentence and 4) changing the masking pattern dynamically when pre-training. RoBERTa is trained following the BERT-large with L = 24, H = 1024, A = 16, total parameters = 255 M.

3.1.3 ALBERT

Increasing the model size can always improve performance on downstream tasks. However, it also needs large GPU/TPU memory and longer training time. ALBERT is a kind of lite BERT due to reducing some parameters. It incorporates factorized embedding parameterization and cross-layer parameter sharing to scale the pre-trained models. Furthermore, ALBERT uses sentence-order prediction (SOP) task in pre-training instead of next sentence prediction (NSP). ALBERT has three model sizes: ALBERT-large (total parameters = 18 M), ALBERT-xlarge (total parameters = 59 M) and ALBERT-xxlarge (total parameters = 233 M).

3.2 CRF Tagging Model

In NER task, the dependencies across output labels are strong. The NER model can joint use CRF to capture dependency between any two labels. The input of CRF is the hidden state sequence of BERT's last layer $h^L = (h_1{}^L, h_2{}^L, \ldots, h_n{}^L)$, the corresponding prediction output sequence of is $y = (y_1, y_2, \ldots, y_n)$.

Consider $P \in R^{n \times k}$ to be the matrix of scores output from BERT networks, where k is the number of labels, and $P_{i,j}$ means the corresponding score of the jth label of the ith token of input sequence.

Define the score of CRF as:

$$S(h, y) = \sum_{i=0}^{N} A_{y_i, y_{i+1}} + \sum_{i=1}^{N} P_{i, y_i} \tag{3.3}$$

where A is a matrix of transition scores, $A_{i,j}$ represents the score of the transition from label i to label j. The start and end of label sequence y are "*start*" label and "*end*" label, the number of labels should be $k + 2$ and the size of square matrix A becomes $k + 2$.

A softmax over all possible tag sequences yields a probability for the sequence y:

$$p(y|h) = \frac{e^{S(h,y)}}{\sum_{\tilde{y} \in Y_h} e^{S(h,\tilde{y})}} \tag{3.4}$$

where Y_h denotes all possible labels of input h. Maximizing the log-probability of the correct label sequence during training. While decoding, the model predicts the correct output sequence by obtaining the maximize score as:

$$y^* = \arg\max_{\tilde{y} \in Y_h} s(h, \tilde{y}) \tag{3.5}$$

The viterbi algorithm is used to compute the optimal output sequence y^* in (3.5) during decoding.

4 Experiments

In this section, we deal two works: 1) comparing the performance of BERT-CRF, ALBERT-CRF and RoBERTa-CRF; 2) applying probing task to find which layer obtains the best performance of three models in entity extraction task.

4.1 Datasets

In practical, the whole procedure of ATC operation is: 1) if any flight wants to leave the stand, the pilot needs to apply for push-back, after agreeing, 2) the flight enters the taxiway and then taxing to waiting point and 3) holds for take-off clearance. When ATC controller giving clearance, 4) flight enters runway and 5) takes off, and then ATC controller will command the flight to 6) depart from the airport. During these stages, ATC controller may adjust the 7) height and 8) speed of the fight. The flight 9) contacts the approach when it arrives the target airport, then it 10) lands the runway and taxis into the stand.

Therefore, the practical data consists of 12 parts: 1) push-back (short for p-b), 2) taxi, 3) holding, 4) line up, 5) take-off (short for t-off), 6) departure (short for dept), 7) height adjust (short for h-adj), 8) speed adjust (short for s-adj), 9) approach (short for

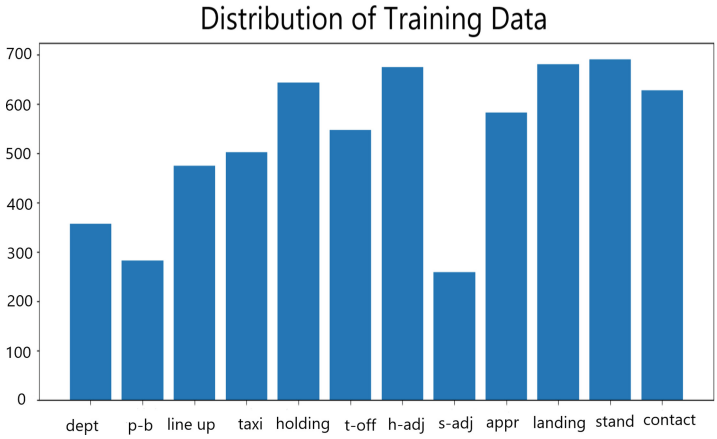

Fig. 2. Distribution of training data

appr), 10) landing, 11) into stand (short for stand), and 12) contact between controller and pilot (short for contact). The distribution of data is shown in Fig. 2.

The practical data includes more than 5000 ATC instructions with average length of 15 words. The longest instruction has 41 words and the shortest one has only 6 words. Experiments choose 90% to be training data and 10% to be testing data. There are 10 types of pre-defined entity: flight number, organization, action, status, location, height, speed, weather, time and other.

4.2 Performance

In this section, we use BERT-CRF, ALBERT-CRF and RoBERTa-CRF to extract the entities of ATC instructions. Table shows their parameters.

Table 1. Model parameters

Models	Total layers	Hidden size	Attn heads	Total parameters
BERT	12	768	12	110 M
ALBERT	12	768	12	59 M
RoBERTa	24	1024	16	255 M

As Table 1 shown, experiments choose BERT-base, ALBERT-base and RoBERTa-large in NER models and Adam to be optimizer. The experiment is done by applying one NVIDIA GeForce GTX 1060 GPU to train the models and the major parameters are shown following (Table 2).

Table 2. Parameter settings

Batch size	32
Max sentence length	45
Adam learning rate	0.01
Number of epochs	30
Dropout rate	0.1

Define the label of entity by using "BOI" format for data, where "B_" means beginning word of phrase, "I_" means the word belongs to phrase and "O_" means the word does not belong to any entity (Fig. 3).

As Fig. 3 shown, after 30th epochs, the error of three models converges below 3% . Among them, ALBERT obtains the best performance, which achieves less than 2% error rate. However, these results are based on full layers of each model, it is necessary to find which layer of model can achieve the best performance in order to further optimize the model.

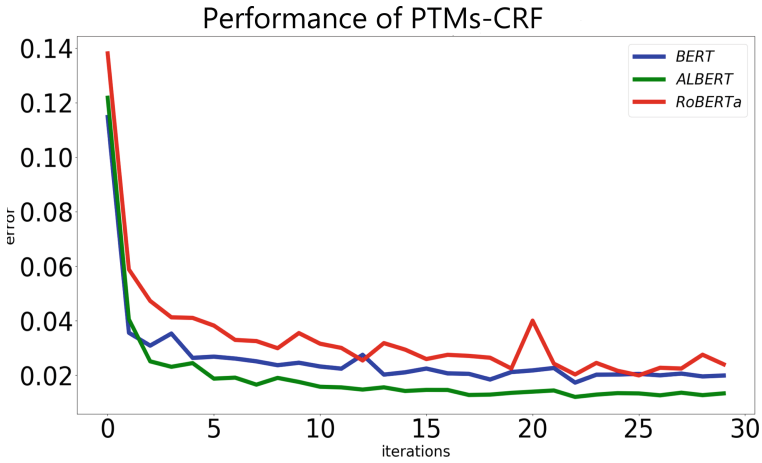

Fig. 3. Performance of BERTs-CRF models for entity extraction

4.3 Probing Task

In this section, probing task will investigate the effectiveness of features from different layers for extracting entities.

In entity extraction task of ATC instructions, it shows that the middle layers of models obtain the best performance. The 6th layer of BERT achieves 0.72% testing error, the 5th layer of ALBERT achieves 1.32% testing error and the 7th layer of RoBERTa achieves 1.15% testing error (Table 3).

Table 3. Error rate of models

Models	Num of layer	Testing error
BERT	6	0.72%
ALBERT	5	1.32%
RoBERTa	7	1.15%

There are two findings: 1) entity extraction task is more depend on syntactic feature of ATC instructions, and 2) ATC instruction is a kind of short text and it has simple semantic feature, so the models depend less on semantics feature.

5 Conclusions

This paper proposes PTMs-CRF model for extracting the information of ATC instructions and conducts experiments to investigate different versions and layers of BERT for entity extraction task. There are some experiments findings: 1) BERT achieves excellent performance, 2) the middle layers of BERT outperform other layers, and 3) The ATC

instruction has simple semantic feature so the model depends more on syntactic feature of instructions.

Acknowledgements. The authors are grateful to CAAC North China Regional Administration for providing data. They also thank the anonymous reviewers for their critical and constructive review of the manuscript. This study was supported by the Fund of the China National Key R&D Program (No. 2020YFB1600100).

References

1. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvist. Investig. **30**(1), 3–26 (2007). https://doi.org/10.1075/li.30.1.03nad
2. Li, J., Sun, A.X., Han, J.L., Li, C.L.: A survey on deep learning for named entity recognition. IEEE Trans. Knowl. Data Eng. **29**, 1–1 (2020)
3. Bikel, D., Miller, S., Schwartz, R., Weischedel, R.: Nymble: a high-performance learning name-finder. In: Proc. Conference on Applied Natural Language Processing (1997)
4. McCallum, A., Li, W.: Early results for named entity recognition with conditional random fields, features induction and web-enhanced lexicons. In: Proc. Conference on Computational Natural Language Learning (2003)
5. Hammerton, J.: Named entity recognition with long short-term memory. In: HLT-NAACL 2003, vol.: 4, pp. 172–175 (2003)
6. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Natural architectures for named entity recognition. In: arXiv preprint arXiv: 1603.01360 (2016)
7. Cui, L., Zhang, Y.: Hierarchically-refined label attention network for sequence labelling. In: arXiv preprint arXiv: 1908.08676 (2019)
8. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
9. Chiu, J., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. In: arXiv preprint arXiv: 1511.08308 (2015)
10. Ma, X., Hovy, E.: End-to-end sequence labelling via bi-directional lastm-cnns-crf. In: arXiv preprint arXiv: 1603.01354 (2016)
11. Qiu, S.P., Sun, T.X., Xu, Y.G., Shao, Y.F., Dai, N., Huang, X.J.: Pre-trained models for natural language processing: a survey. Sci. China Technol. Sci. **63**(10), 1872–1897 (2020). https://doi.org/10.1007/s11431-020-1647-3
12. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. In: URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf (2018)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)
14. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**(Feb), 1137–1155 (2003)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NeurIPS (2013)
16. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: EMNLP (2014)
17. Peters, M.E., et al.: Deep contextualized word representations. In: NAACL-HLT (2018)
18. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: ACL, pp. 328–339 (2018)

19. Liu, Y.H., et al.: RoBERTa: a robustly optimized BERT pretraining approach. In: arXiv preprint arXiv:1907.11692 (2019)
20. Lan, Z.Z., Chen, M.D., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: a lite-BERT for self-supervised learning of language representations. In: arXiv preprint arXiv: 1909.11942 (2019)
21. Yang, Z.L., Dai, Z.H., Yang, Y.M., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: XLNet: generalize autoregressive pretraining for language understanding. In: NeurIPS, pp. 5754–5764 (2019)
22. Jawahar, G., Sagot, B., Seddah, D.: What does BERT learn about the structure of language? In: ACL, pp. 3651–3657 (2019)
23. Wallace, E., Wang, Y.Z., Li, S.J., Singh, S., Gardner, M.: Do NLP models know numbers? probing numeracy in embeddings. In: Proc of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 5307–5315 (2019)