

言語モデルを用いた Ansible のプレイブックファイル作成時における ヒューマンエラー発生防止システムの開発

Development of a human error prevention system when creating Ansible playbook files using a language model

川口 真護† 水谷后宏‡§ 井口 信和‡§
Mamoru Kawaguchi Kimihiro Mizutani Nobukazu Iguchi

1. 序論

ネットワークエンジニアの業務の一つに、ネットワークの運用保守(以下、運用保守)がある。この業務は夜間など一般的な業務時間外での作業が多いため、ヒューマンエラーが多く起こるおそれがある[1]。また、ネットワークの運用管理の複雑化などの課題がある[2]。これら課題の解決策の一つとして、運用保守の自動化や効率化などがあり、この自動化等を実現するためには、人工知能の技術などを用いることが不可欠である[3]。本稿では、運用保守作業の一つであるネットワーク機器の設定変更作業に焦点をあてる。

ネットワーク機器の設定変更作業では、作業手順書の作成、ネットワーク機器の設定変更、ネットワークの動作確認を実施する。これらの作業を自動化する方法として、構成管理ツールを用いた方法があげられる。このツールの一つとして、2014 年頃から最も注目されている Ansible を例にとり[4]。Ansible を利用すると、ネットワーク機器の設定変更の自動化が可能となり、ヒューマンエラーの低減が期待できる。しかし、Ansible を用いた場合でも、実行する処理の流れが記載されたプレイブックファイルの作成をする時に、人の手が介在するため、ヒューマンエラーが起こる場合がある。

そこで我々は、これまでに、運用保守の自動化を目的に Ansible のプレイブックファイル作成時におけるヒューマンエラーの発生防止を目標とし、言語モデルを用いた Ansible のプレイブックファイル作成時におけるヒューマンエラー発生防止システム(以下、本システム)の検討を行ってきた[5]。本システムでは、Web ブラウザ上のコード補完機能が搭載されたエディタを用いて Ansible のファイル作成を実施する。

本稿では、本システムのコード補完機能の実装をした。コード補完機能は、本システムの利用者が Web ブラウザ上に入力したコードから言語モデルを用いて次のコードを予測する機能である。この機能を用いることで、Ansible のプレイブックファイル作成時の手作業を削減する事ができ、綴間違いなどのヒューマンエラーの発生を防止できる。また、作成した言語モデルの推薦精度の評価を実施した。

2. 設定変更作業

本節では、Ansible の概要とプレイの構成を示す。プレイ

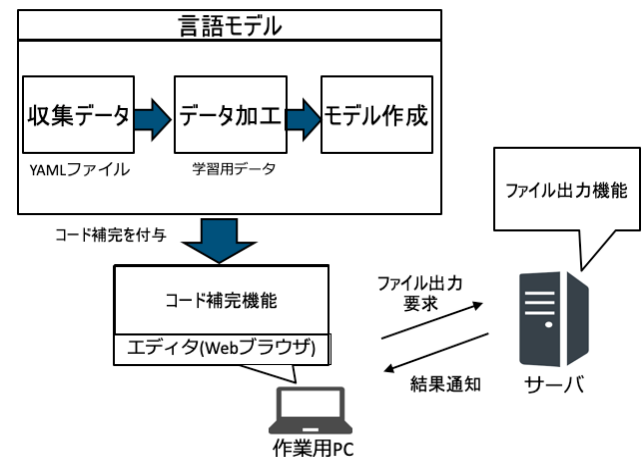


図1 本システムの構成

とは、プレイブックファイルを構成する処理の纏まりである。また、本研究が対象としている作業範囲についても述べる。

2.1. Ansible の概要 [6]

Ansible は、処理の指示を出すコントロールノードと処理の対象となるターゲットノードから構成されている。コントロールノードに Ansible をインストールし、インベントリファイルとプレイブックファイルを用意する。その後、実行コマンドを実行する事でターゲットノードに自動的にネットワークの設定を施すことができる。インベントリファイルは、ターゲットノードの接続情報を記載したファイルである。プレイブックファイルは、ターゲットノード側で実行する処理の流れを記載したファイルである。また、プレイブックファイルはプレイと呼ばれる処理の纏まりをシーケンスとして並べたものである。

2.2. プレイの構成と作業範囲

Ansible のプレイの構成は以下の通りとなっている[7]。プレイの中では、ディレクティブとその値によって処理が定義されている。また、本研究では、これらのセクションのうち、最も記述量が多い Tasks セクションをコード補完の対象とする。

A) Targets セクション

ターゲットノードの接続に関する必要な情報を記述する箇所。

B) Vars セクション

† 近畿大学大学院総合理工学研究科, Graduate School of Science and Engineering Research, Kindai University

‡ 近畿大学情報学部, Faculty of Informatics, Kindai University

§ 近畿大学情報学研究所, Cyber Informatics Research Institute, Kindai University

タスクを効率化するための変数を記述する箇所

C) Tasks セクション

実行したい処理内容を記述する箇所。ディレクティブにモジュール名、その値に各モジュールのオプションを定義できる。

D) Handlers セクション

「notify」を指定したタスクが更新された場合に、実行したい処理内容を記述する箇所。notify に定義された名前と同じハンドラタスクが実行される。ハンドラの呼び出し方は、notify ディレクティブの値に実行したいハンドラ名を記述し、handlers セクションに同様のタスク名を定義することで実行される。

3. 関連研究

関連研究として、華山らの研究[8]がある。華山らの研究では、Docker と呼ばれるプラットフォームのコード補完に着目し、Dockerfile の作成を支援するシステムを開発している。

華山らの研究と本研究の共通項は、言語モデルを用いてコード補完を実現している点である。華山らの研究では、Docker に関する開発支援などの研究を推し進める事を目的としている。その一方で、本研究では、運用保守の自動化を目的に Ansible のプレイブックファイル作成時におけるヒューマンエラーの発生防止を目標としている。また、華山らの研究では、学習用データとテスト用データを区別せずに使用しているが、本研究では、学習用データとテスト用データをそれぞれ、Ansible のドキュメント[9]及び Ansible クックブック[10]内のコードとし、区別している。学習用データとテスト用データを区別した理由は、汎化性能を評価するためである。

また、他の関連研究として、御木らの研究[11]がある。御木らの研究では、大学のネットワークの利用者が起こしたヒューマンエラーを検知する手法を提案している。一方で、本研究では、言語モデルを用いた Ansible のプレイブックファイル作成時におけるヒューマンエラー発生防止システムを提案している。

4. 研究内容

本システムの構成を図1に示す。本システムはクライアントプログラムとサーバサイドプログラムによって構成される。クライアントプログラムは言語モデルによって付加されるコード補完機能を有している。また、Web ブラウザ上のエディタを用いて Ansible のプレイブックファイルの作成が可能である。サーバサイドプログラムはファイル出力機能を有しており、クライアントプログラムからのファイル出力要求に伴って、ファイルを出力する。次節以降に実装した機能について述べる。

4.1. システム要件

本研究の目標が Ansible のファイル作成時におけるヒューマンエラーの発生防止であるためシステム要件として次の2つを考案した。

要件 1: Ansible のファイル作成時の手作業を削減する

要件 2: 作成したコードを Ansible で運用可能な形にする

この二つの要件に対応するために、コード補完機能、ファイル出力機能を実装した。

4.2. コード補完

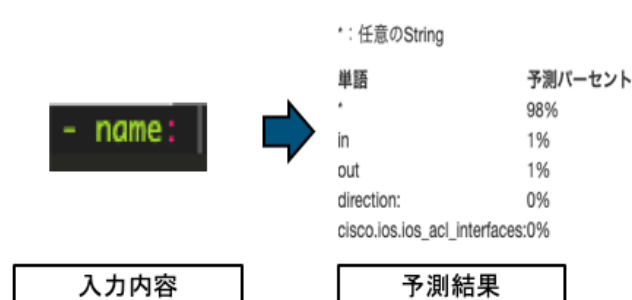


図2 予測結果

入力内容 (数値)	期待する内容 (数値)
-	name:
(0, 0, 1)	(3)
- name:	*
(0, 1, 3)	(2)
- name: *	direction:
(1, 3, 2)	(4)

表1 データ加工例

コード補完とは、コードの入力中に次のコードを予測し、補完候補を表示する機能である。コード補完は2種類存在している[12]。1つ目は、伝統的なコード補完である。伝統的なコード補完では、可能性のある補完候補を全て表示する。そのため、開発者がその非常に長い候補から入力値を選ぶ必要がある。このような課題を解決するために提案されたのが、2つ目のインテリジェントコード補完である。インテリジェントコード補完は、文脈を考慮し、言語モデルを用いて補完候補を絞ることで、伝統的なコード補完よりも表示する候補が少なくなり、開発者の生産性を向上できる。また、N-gram, Best Matching Neighbor(BMN)や Recurrent Neural Network(RNN)などは、インテリジェントコード補完システムを構築する際に、高い性能を発揮してきた。そこで、本システムでのコード補完機能の実装には、RNNを用いた手法を採用している。

4.3. コード補完機能

本機能は、Ansible のプレイブックファイル作成時のコード補完を行う機能である。本システムの利用者はWebブラウザ上のエディタに任意のコードを記述する。言語モデルは、入力されたコードの文脈を考慮し、確率順に上位5件の次のコードの候補を表示する。本機能によって推論された結果を図2に示す。本機能により、Ansible のプレイブックファイル作成時の手作業を削減する事ができる。また、言語モデルを作成するために、データを収集し、そのデータをRNNで学習させた。以下に、本機能を実装した手順を説明する。

4.3.1. データ収集

Ansible のドキュメントから Cisco モジュールである cisco.ios.ios_acl_interfaces モジュールのファイルを収

集した。このファイルの単語数は 173 個である。

4.3.2. データ加工

収集した Ansible のファイルから RNN の学習に不要なコードのコメント部分を削除した。name や ip-address などのパラメータ部分は Ansible を使用する人の環境によって変化するため、*という特殊文字に置換した。また、各文字列をトークンごとに分け、辞書を作成した。そして、トークンごとに数値化を行い、学習データに基づき、入力内容と期待する出力内容をセットとし、データを加工した。その時に、全データが固定長となるように 0 埋めを行った。表 1 に言語モデルの入力形状が 3 単語の場合のデータの加工例を示している。また、入力形状を 3 単語とした場合、データセットの組み合わせは、513 通りとなった。

4.3.3. 言語モデルの作成

本システムでは、コードを予測するために言語モデルを使用する。言語モデルの作成には、Long-Short-Term Memory (LSTM) [13] を使用した。LSTM とは、長期的な依存関係を学習することが出来る RNN の一種である。このニューラルネットワークの入力形状や epoch 数については、実験の項目で述べている。

4.3.4. 言語モデルの展開

作成した言語モデルは、TensorFlow.js を用いて Web ブラウザ上のエディタに展開し、クライアントプログラムにコード補完機能を付与した。

4.4. ファイル出力機能

本機能は、Ansible のファイルを出力する機能である。本システムの利用者は、Web ブラウザ上のエディタを用いて、Ansible のファイルを作成し、保存ボタンを押下する。サーバは、作成された Ansible のファイルを利用者のローカル環境上に保存する。本機能により、利用者が作成したコードをファイルへ変換できる。

5. 実験

実験では、言語モデルの推薦精度の評価を実施した。テストデータセットとして、Ansible クックブック [14] の cisco.ios.ios_acl_interfaces モジュールを使用している箇所のコードを使用した。

まず、テストデータセットのコードの任意の位置での正解単語を選択する。次に正解単語の手前までの内容を言語モデルの入力形状の範囲内で言語モデルに入力し、予測候補を出力する。この試行を実験回数 1 回とし、複数回実施する。最後に、予測候補と正解単語を比較し、Top-k 精度、MRR の値を計算する。今回の実験では、Top-1、Top-5 精度、MRR の値を計算している。Top-k 精度は、上位 k 位の予測候補のうちの正解単語数をテストデータの標本数の合計で割った値である。MRR は、予測候補のうち正解単語の位置の逆数をテストデータの標本数の回数求め、足していき、その結果をテストデータの標本数で割った値である。

5.1. 実験結果・考察

実験結果は表 2 の通りとなった。実験の結果より、入力形状を増やすことで Top1 精度、MRR の数値が上昇していることが分かる。また、入力形状が 5 単語の時、Top1 精度は 80%、MRR は 0.875 という高い推薦精度を有することを確認

入力形状	Epoch 数	実験回数	Top1	Top5	MRR
3 単語	100	10	60%	100%	0.766
5 単語	100	10	80%	100%	0.875

表 2 実験結果

した。このことから、ネットワーク運用管理の自動化と開発者のヒューマンエラーの発生防止に寄与できると考えられる。

6. 結論・今後の予定

本研究では、運用保守の自動化を目的に Ansible のプレイブックファイル作成時におけるヒューマンエラーの発生防止を目標とし、言語モデルを用いた Ansible のプレイブックファイル作成時におけるヒューマンエラー発生防止システムを開発した。本システムを用いることで、ヒューマンエラーの発生防止及び開発者の生産性向上につながると考えられる。

今後の予定として、コード補完機能に先読みアルゴリズムを追加しようと考えている。先読みアルゴリズムとは、入力した内容に基づき、予測される候補の次の候補まで表示するアルゴリズムである。入力内容から言語モデルが予測するコードの候補リストを再帰的に言語モデルに入力することによって、このアルゴリズムを実装しようと考えている。先読みアルゴリズムを実装することにより、開発者の生産性がより向上することを期待している。また、学習データを増やし、他のモジュールに対するコード補完も可能になるようにしたいと考えている。

加えて、追加実験も実施したいと考えている。言語モデルの入力形状、epoch 数を変化させていき、精度がどの様に変化するかを調査したいと考えている。

謝辞

運用管理に関するアドバイスを頂きました（株）サイバーリンクス 松山様、那須様に感謝致します。

参考文献

- [1] 有藤平八郎：ヒューマンエラーと睡眠，産業医学，Vol.35，No.4,p.255(1995).
- [2] 情報通信技術委員会:ネットワーク分野の課題とその解決に向けた TTC の取り組み，入手先 <https://www.soumu.go.jp/main_content/000578678.pdf>(参照 2021-12-2).
- [3] 前掲：ネットワーク分野の課題とその解決に向けた TTC の取り組み，入手先 <https://www.soumu.go.jp/main_content/000578678.pdf>(参照 2021-12-2).
- [4] GitHub Stars，入手先 <<https://ja.exploratory.io/viz/PQd2wHB2BZ/GitHub-Stars-SaltStack-Ansible-Terraform-Chef-Puppet-and-Pulumi-kxA6lHC6OF>>(参照 2022-7-5)
- [5] 川口真護，水谷后宏，井口信和：言語モデルを用いた Ansible のファイル作成時におけるヒューマンエラー発生防止システムの検討，情報処理学会第 84 回全国大会講演論文集(2022)

- [6] 北山 晋吾,佐藤 学,塚本 正隆,畠中 幸司,横地 晃 :
Ansible 実践ガイド第 3 版
- [7] 前掲 : Ansible 実践ガイド第 3 版
- [8] 華山魁生, 柏本真佑, 楠本真二:言語モデルに基づく
Dockerfile コード補完システムの提案, 電子情報通信
学会技術研究報告信学技報, Vol. 120, No.193, pp19-
24(2020).
- [9] Ansible Documentation, 入手先
<<https://docs.ansible.com/ansible/latest/>>(参照 2022-6-30)
- [10] 大嶋 健容,三枝 浩太,宮崎 啓史,横地 晃 : Ansible クック
ブック
- [11] 御木孝亮, 富澤真樹: 人為ミスによるネットワーク障
害 検知方法の設計, 情報処理学会 研究報告情報シス
テム と社会環境(IS), Vol.2005, No.115(2005-IS-094),
pp.15- 21(2005)
- [12] 前掲 :言語モデルに基づく Dockerfile コード補完シス
テムの提案, 電子情報通信学会技術研究報告信学技
報, Vol. 120, No.193, pp19-24(2020).
- [13] Greff, Klaus, et al. "LSTM: A search space odyssey." IEEE
transactions on neural networks and learning systems 28.10
(2016): 2222-2232.
- [14] 前掲 : Ansible クックブック