深層学習を用いた Web API 仕様文書の生成方法の提案と評価

M2018SE011　永井 利幸
指導教員　青山 幹雄

1 研究背景

Web API の情報を集約した様々なキュレーションサイト[4]が公開されている．そのサイトではそれぞれで異なったテンプレートを使用し，情報を提供している．しかし，異なったテンプレートは，Web API の仕様の把握や比較が困難である．仕様書のフォーマットの統一化のため，OpenAPI Initiative[7]が Open API を提唱している．

本稿では，機械学習を用いて Web API の説明文書を要約し，OpenAPI で定義できる仕様を抽出することによる Web API の仕様文書の生成方法を提案する．

2 研究課題

本稿では，研究背景を踏まえ以下の 3 点を研究課題として設定する．

(1) Attention を用いた機械学習による Web API 記述から要約した仕様記述の生成方法
(2) 要約仕様記述から OpenAPI 形式の仕様記述生成方法
(3) 実際の Web API に提案方法を適用し，有効性と妥当性の評価

3 関連研究

3.1 Attention を用いた機械学習

機械学習とは，大量のデータからルールや判断基準を抽出し新たなデータについて推論を行う技術である．機械学習の実装を実現するためのフレームワークが提供されている．Preferred Networks の Chainer[3]や Google の TensorFlow[2]などがある．機械学習における Attention とは，入力情報全体ではなく，その一部のみを特に注目したベクトルをデコーダで使用する仕組みのことである．機械学習を行う際に検索クエリに一致するキーを索引し，対応する値を取り出す操作を行う．機械翻訳における Attention は，デコードする際に入力のどの部分に着目すべきかを判断するのに用いる．文書要約では，Attention の値を元に文章の抽出を行う．

3.2 Web API 説明文書の要約方法

文書要約とは，膨大な情報から要点をまとめた短い文章に変換する技術である．文書要約には，抽出型と抽象型がある．抽出型では，抽出するべき文章に対してはラベル1と2，それ以外にはラベル0を出力するようにLSTM(Long Short-Term Memory)を用いてモデルを学習させるNeural Summarization by Extracting Sentences and Words[1]などがある．抽象型では，エンコーダ(LSTM)で文章を潜在表現に圧縮しデコーダ(LSTM)で圧縮された表現から文を生成するEncoder-Decoderモデルがある．

機械学習を用いて自動的に仕様書を生成する方法が提案されている[9]．階層的クラスタリングを用いてURL，パステンプレート，HTTPメソッドを自動的に抽出し，ベースURLやWeb APIのエンドポイントを含んだ仕様書を生成する．116のWeb APIに対して実験を行い，Web API仕様書と公的な仕様書で多くの矛盾があることを発見した．

3.3 OpenAPI

OpenAPI[5]とは，REST API の記述形式である．Swagger 2.0[6]を拡張して定義されている．記述形式は，YAML と JSON の 2 種類から選択できる．API，サーバ，パス，コンポーネント，セキュリティに関する情報を記述でき，それぞれオブジェクトとして定義される．API コンシューマが API プロバイダにリクエストを送る際，バックグラウンドで仕様書生成が行われている．API コンシューマが参照するのは，Swagger の書式で記述した仕様書である Swagger Spec を基に生成されたドキュメントである．Swagger Codegen, Swagger UI, Swagger Editor, Swagger Core などのツールによって OpenAPI 形式の仕様書が生成されている．

4 アプローチ

従来の方法では静的に定義しているため，Web API を公開した後に変更が発生した場合，仕様書を変更して Web API を公開し直す必要がある．このことから，仕様の変更を動的に仕様書に反映できるプロセスを提案する(図 1).

ProgrammableWeb 上の説明文書を抽出し，OpenAPI の形式に変換する．ProgrammableWeb では，独自の仕様記述フォーマットを使用している．ProgrammableWeb での Web API の挙動を説明した文書を要約し，OpenAPI で定義可能な仕様を抽出することで，仕様書を生成する．要約を行うことで，Web API の仕様に対する理解が促進される．

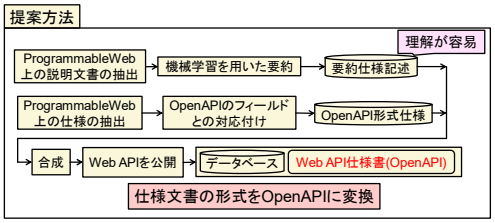

図 1 仕様変更を動的に反映可能なプロセス

5 深層学習を用いた Web API の仕様文書の生成方法

5.1 Web API 仕様文書生成における提案プロセス

図 1 のプロセスを例題に基づき詳細化した Web API 仕様文書生成における提案プロセスを図 2 に示す．

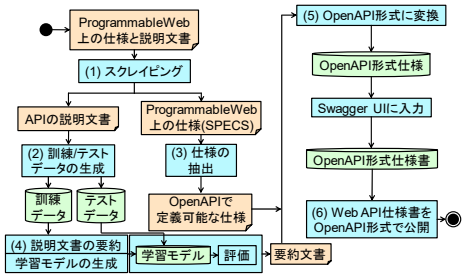

図 2 提案プロセス

(1) Web API 記述文書のスクレイピング

ProgrammableWeb 上に登録されている Web API の説明文書と仕様を合わせたテキストデータを取得する．スクレイピングを行った際に同時に抽出されたHTMLタグなどは取り除き，データ整形を行う．説明文書と仕様は，それぞれ別のテキストファイルとして分割する．

(2) 訓練データとテストデータの生成

モデルの学習と評価に用いる訓練データとテストデータを生成する．ProgrammableWeb 上でのカテゴリ内で Web API の登録数に偏りがあるため，カテゴリ毎で訓練データとテストデータの生成を行う．

(3) Web API 仕様記述の抽出

OpenAPI で定義可能な仕様を取得する．ProgrammableWeb で用いられている仕様記述フォーマットを表 1, OpenAPI で定義可能な仕様を表 2 にそれぞれ示す．ProgrammableWeb で定義されている仕様をそのまま OpenAPI でも定義可能なものと不可能なものが存在する．ProgrammableWeb 上の仕様の OpenAPI のフィールドへの変換を表 3 に示す．ProgrammableWeb で定義できる仕様は Web API のメタデータに関する情報が多いため，OpenAPI の info フィールドに記述する．ProgrammableWeb の定義が欠落している場合は，情報が存在しないことを示す．パスが存在していない場合は，paths の deprecated フィールドを true と記述する．その他の場合は，info の description フィールドに記述する．

(4) Web API 仕様記述の要約

ProgrammableWeb 上の Web API の挙動やレスポンスの形式を示した説明文書を要約する．機械学習を用いて重要と思われる文章を抽出し，info の description フィールドに記述する．

(5) Web API 仕様記述の OpenAPI 形式への変換

要約された Web API の説明文書と OpenAPI で定義できる仕様を合成し，OpenAPI 形式に変換する．Swagger Editor を用いて動的に表示されるプレビューを確認し，編集する．

(6) Web API 仕様書を OpenAPI 形式で公開

Swagger Editor を用いてサーバを生成する．YAML ファイルがアップロードされ，OpenAPI 形式で公開される．SwaggerUI を用いて生成した Web API 仕様文書の内容を検証する．

5.2 訓練データとテストデータの生成

モデルの学習と評価に用いる訓練データとテストデータを生成する．ProgrammableWeb 上では多くのカテゴリがあり，カテゴリ毎で説明文書に記述される内容は異なる．本研究では，訓練データとテストデータを一つのカテゴリから抽出した場合と 5 つのカテゴリから抽出した場合の 2 つを比較して実験を行う．訓練データは要約を行うためのモデル生成，精度や訓練誤差の評価に用いる．テストデータは未知のデータに対してモデルを用いた要約，精度や汎化誤差の評価に用いる．精度と汎化誤差の評価により，学習が進んでいるか判断が可能である．

5.3 Web API の表現モデル

ProgrammableWeb 上での仕様と OpenAPI 上での仕様との対応付けを行う．表 3 を用いて OpenAPI 形式の仕様に変換する．ProgrammableWeb 上で仕様が定義されていない場合は，対応する OpenAPI 上での仕様を空欄とする．

表 1 ProgrammableWeb の仕様記述フォーマット

仕様	記載すべき事項
1. API Endpoint	API コールで用いられるベース URL
2. API Portal / Home Page	API のランディングページ
3. Primary Category	サービスと最も適合したカテゴリ
4. Secondary Categories	サービスを説明するのに助けとなるカテゴリ
5. API Provider	API を提供する企業
6. SSL Support	トラフィック保護のための SSL のサポート
7. Twitter URL	開発者チーム Twitter URL
8. Support Email Address	サポートのリクエストのための E メールアドレス
9. Authentication Model	エンドポイントによってサポートされている認証モデル
10. Version	API のバージョン番号
11. Terms Of Service URL	サービスの条件を記載した URL
12. Is the API Design/Description Non-Proprietary ?	API が非機密であるかどうか
13. Type	5 つの API タイプから選択
14. Scope	Aggregate, Microservice, Single purpose の 3 つから選択
15. Device Specific	デバイスに固有の API かどうか
16. Docs Home Page URL	API のドキュメントの URL
17. Architectural Style	API のアーキテクチャスタイル
18. Supported Request Formats	ペイロード(リクエスト)のフォーマットを一つ以上選択
19. Supported Response Formats	ペイロード(レスポンス)のフォーマットを一つ以上選択
20. Is This an Unofficial API?	非公式の API であるかどうか
21. Is This a Hypermedia API?	ハイパーメディアをサポートしているかどうか
22. Restricted Access (Requires Provider Approval)	制限されたアクセスかどうか(プロバイダの承認が必要)

表 2 OpenAPI で定義可能な仕様

フィールド名	必須	概要
openapi	○	バージョニングを記述
info	○	API のメタデータを記述
servers	×	API を提供するサーバを記述(配列で複数記述可能)
paths	○	API で利用可能なエンドポイントやメソッドを記述
components	○	API で使用するオブジェクトスキーマを記述
security	×	API 全体を通して使用可能なセキュリティ仕様を記述
tags	×	API で使用されるタグのリスト(タグは固有)
externalDocs	×	外部ドキュメントを記述

表 3 OpenAPI のフィールドへの変換

ProgrammableWeb の仕様(番号)	OpenAPI のフィールド
3, 4, 5, 7, 8, 10, 11, 13, 14, 15, 17, 20, 21, 22	info
2	servers
1	paths
18, 19	components
6, 9, 12,	security
該当なし	tags
16	externalDocs

5.4 Web API 説明文書の要約

5.4.1. CNN, Seq2Seq, Attention を用いた文書要約方法

ProgrammableWeb 上の説明文書を要約する．LSTM,

CNN(Convolutional Neural Network), Attention を用いて文書要約を行う．LSTM ブロックは，入力ゲート，忘却ゲート，出力ゲート，メモリセルで構成されている．本研究では，Web API の説明文書における長期記憶や短期記憶の依存関係を考慮できるため使用した．通常の文書要約では，文書間で文章量をほぼ同等にしたデータセットに対して行われる．Web API の説明文書は，Web API 毎で文章量や仕様を記述する順番が異なっているという特徴がある．またカテゴリは多岐にわたり，カテゴリ毎でドメイン知識が異なっている．提案方法では各文章の tf-idf を計算し，重要文を特定する．tf-idf を式(1)に示す．単語ごとの tf-idf を計算し，それらを加算してスコアを算出する．1 つの文書内でスコアが最も高かった文章に対してラベル 1, スコアが 2 番目，3 番目に高かった文章に対してラベル 2, それ以外の文章に対してラベル 0 を付与し学習を行う．Mapping, Weather, Mobile, Transportation, Social の 5 つのカテゴリをそれぞれ学習する場合と，5 つのカテゴリを一括して学習する場合の 2 パターンを行う．

5.4.2. Attention を用いた文章抽出

文書要約のアーキテクチャを図 3 に示す．文章エンコーダにおいて分散表現で入力された文章に対して CNN を適用する．文書エンコーダにおいて出力された文章ベクトルを順次 LSTM に入力することで文書を表現する隠れ層が生成される．文章抽出器において LSTM を用いて文章の抽出する．あるタイムステップにおける文章抽出器の隠れ層は，直前のタイムステップの隠れ層と値 p によって重み付けされた文章のベクトルによって決定される．p が文章を抽出するべきか判断する指標になる．p は文章に対する Attention である．p の値が大きいほど，その文章の重要度が高いと判断できる．p の値は文書エンコーダと文章抽出器の隠れ層を結合したベクトルを多層パーセプトロンに入力し，シグモイド関数に入力することで 0 から 1 の間の数値として得られる．学習の際に重要文に 1,2，その他の文に 0 を付与したラベルを使用する．重要文は tf-idf を計算し，スコアの高かった上位 3 文章とする．tf-idf は文章中に多数出現する，かつ他文章中に出現しない重要語を特定できる．重要語を多数含む文章を重要文と定義する．

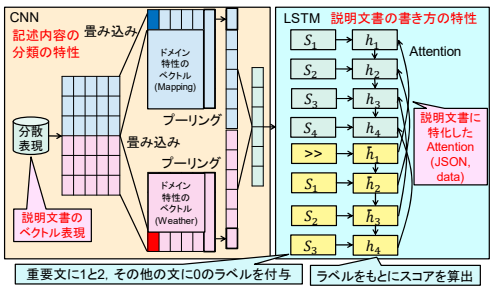

図 3 文書要約のアーキテクチャ

$$tf\ idf = tf \times idf \tag{1}$$

$$tf = \left(\frac{\text{文書}A\text{における単語}X\text{の出現頻度}}{\text{文章}A\text{における全単語の出現頻度の和}} \right) \tag{2}$$

$$idf = \log\left(\frac{\text{全文書数}}{\text{単語}X\text{を含む文書数}} \right) \tag{3}$$

5.5 学習モデルの生成

ハイパーパラメータ(最適化アルゴリズム，学習数(epoch)，バッチサイズ)を決定し学習モデルの生成を行う．最終的な要約結果の精度によってハイパーパラメータの設定を見直す．訓練誤差，汎化誤差，精度の評価に基づいて繰り返し学習を行い，最適なモデル設計を確定する．

5.6 学習の評価

生成したテストデータを用いて要約を行い，式(4)で表される ROUGE，式(5)で表される単語の網羅率，式(6)で表される圧縮率を用いて評価を行う．訓練誤差と汎化誤差は，TensorFlow[2]の評価ツールである TensorBoard を用いて可視化する．ROUGE による評価で値が 0 に近い場合や TensorBoard を用いた評価で誤差が発散している場合は，ハイパーパラメータの再設定や訓練データとテストデータの割合の変更を行う．

6 プロトタイプの実装

6.1 プロトタイプのアーキテクチャ

提案方法を評価するために実装したプロトタイプのアーキテクチャを図 4 に示す．

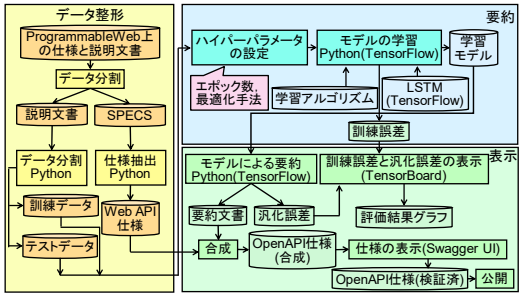

図 4 プロトタイプのアーキテクチャ

(1) データ整形

ProgrammableWeb から取得したオリジナルデータを説明文書と SPECS に分割し，説明文書から訓練データとテストデータ，SPECS から OpenAPI で定義可能な仕様を抽出する処理を Python で実装した．

(2) 要約

モデルの学習処理を Python と TensorFlow で実装した．

(3) 表示

訓練誤差と汎化誤差による評価結果の可視化を TensorBoard によって行った．仕様文書の表示を SwaggerUI で行った．

6.2 実装環境と実装結果

プロトタイプの実装環境を表 4 に示す．

表 4 ソフトウェアコンポーネント

コンポーネント	コンポーネント名	版
OS	Ubuntu	18.04
実装言語	Python	3.6.4
深層学習フレームワーク	TensorFlow	1.3.0
評価結果の可視化	TensorBoard	0.1.8
スクレイピングツール	Beautiful Soup	4.2.0
Web API 記述	OpenAPI	3.0.2

7 実データへの適用による評価

7.1 適用の目的と評価方法

実際の ProgrammableWeb 上の説明文書に対してプロトタ

イプを適用し，提案方法の有効性と妥当性を評価する．

7.2 適用対象

ProgrammableWeb の Web API に対して提案方法を適用した．文章数の違いによる結果への影響を最小限にするため，抽出する Web API の数は約 300 に統一した．表 5 に適用対象のデータ数を示す．

表 5 適用対象のデータ数

カテゴリ	Mapping	Weather	Mobile	Transportation	Social
最新更新日時	2019/8/6	2019/8/7	2019/8/15	2019/8/17	2019/7/15
Web API 登録数	1,030	299	1,063	509	1,547
文章数	1,126	1,151	1,148	1,137	1,143
単語数	19,689	20,614	19,461	19,165	19,141

7.3 評価結果

ProgrammableWeb の Weather カテゴリに属する Web API(Magic Seaweed Forecast API)に対して要約を行った結果を図 5 に示す．4 文(88 単語)の文書が 2 文(47 単語)に短縮された．削除された文章は同じ内容を繰り返している文や，Web API ではなくサービスに関する情報を含んでいた．評価指標として訓練誤差と汎化誤差，式(4)に示す ROUGE-1(N=1)，式(5)に示す単語の網羅率，式(6)に示す圧縮率を用いる．ROUGE-1，単語の網羅率，圧縮率は 0 から 1 までの範囲をとる．ROUGE-1 は，参照要約中の N グラム数に対する参照要約とシステム要約間で一致する N グラム数の割合をスコアとする．元の文書と抽出された文章を比較し，共通する単語が多いほど高い値をとる．単語の網羅率は，参照要約とシステム要約で一致した単語数を参照要約の単語数で割った値である．圧縮率は，参照要約の単語数からシステム要約の単語数を引いた値を参照要約の単語数で割った値である．訓練誤差の評価結果を図 6，汎化誤差の評価結果を図 7，ROUGE-1，単語の網羅率，圧縮率の評価結果を図 8 に示す．全てのカテゴリを学習した場合の評価は，Weather カテゴリの Web API を用いて行った．

$$\boldsymbol{ROUGE-N}(C,\mathcal{R}) = \left(\frac{Count_{match}(gram_N)(C,\mathcal{R})}{\#\ of\ N\text{-}grams\ \in\ \mathcal{R}}\right) \quad (4)$$

$$\textit{単語の網羅率} = \frac{\textit{説明文書と要約で一致した単語数}}{\textit{説明文書の単語数}} \quad (5)$$

$$\textit{圧縮率} = \frac{\textit{説明文書の単語数} - \textit{要約の単語数}}{\textit{説明文書の単語数}} \quad (6)$$

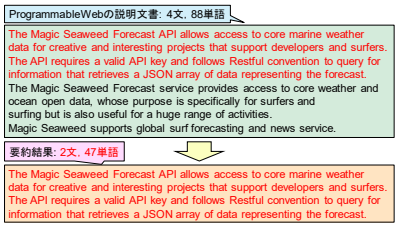

図 5 要約の実行結果

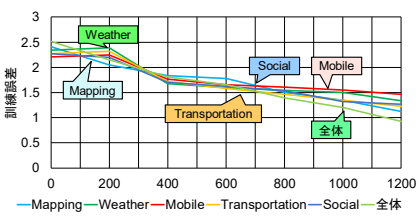

図 6 訓練誤差の評価結果

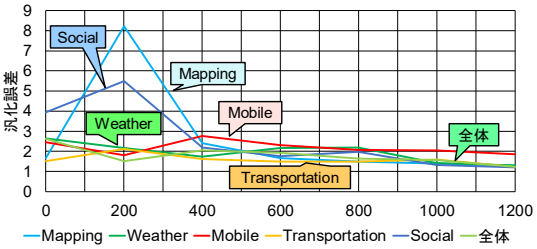

図 7 汎化誤差の評価結果

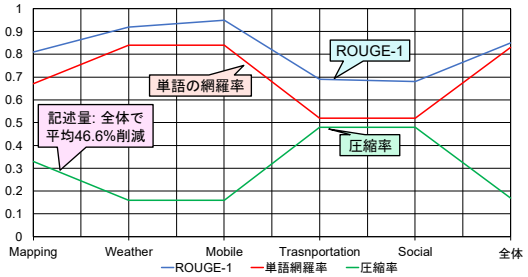

図 8 ROUGE-1，単語の網羅率，圧縮率の評価結果

8 考察

8.1 評価に基づく考察

図 8 の評価結果から利用者にとって不要である情報を削除でき，重要な情報を保持したまま要約を実行できた．習得容易性[8]が向上し，少ない情報量で仕様を把握できる．

8.2 関連研究との比較

関連研究[9]では機械学習を用いて仕様書に記述された URL などを自動的に抽出しているが，説明文書の要約はされていない．したがって，仕様文書の記述量が増大する．

提案方法では仕様記述の抽出と説明文書の要約が可能である．

9 今後の課題

(1) Attention を活かした分析の検討
(2) 他キュレーションサイトへの適用

10 まとめ

機械学習を用いて説明文書を要約し，OpenAPI で定義できる仕様を抽出することによる Web API 仕様文書の生成方法を提案した．文書要約により Web API 利用者の仕様理解の促進が可能になる．提案方法は異なる仕様記述フォーマットの統一への一アプローチとして期待できる．

参考文献

[1] J. Cheng, et al., Neural Summarization by Extracting Sentences and Words, Proc. of ACL'16, Jul. 2016, pp. 484-494.
[2] Google Inc., TensorFlow, https://www.tensorflow.org/.
[3] Preferred Networks, Chainer, https://chainer.org/.
[4] ProgrammableWeb, https://www.programmableweb.com/.
[5] SmartBear Software, Open API Specification, https://swagger.io/specification/.
[6] SmartBear Software, Swagger, https://swagger.io/.
[7] The Linux Foundation, Open API Initiative, https://www.openapis.org/.
[8] 山本 里枝子 他, Web API の習得容易性と相互運用性，およびその定量評価方法の提案と適用評価，情報処理学会論文誌, Vol. 60, No. 10, Oct. 2019, pp. 1896-1914.
[9] J. Yang, et al., Towards Extracting Web API Specifications from Documentation, Proc. of MSR '18, ACM, May 2018, pp. 454-464.