今回のマンションクレンジングアルゴリズムの詳細

**Details of ‘mansion cleansing algorithm’**

マンションクレンジングアルゴリズム（以下「本アルゴリズム」と略記）の目的 **Aim of ‘mansion cleansing algorithm’ (hereinafter called ‘this algorithm’)**

・ネット上のマンション名は宣伝文句と一緒に記載されていることが多く、余計な単語を取り除いて、キレイなマンション名だけを取り出すこと

As building names on the web is attached with advertising phrases, the aim is to get rid of advertising phrases and get a clean building name.

For example

例**.**【値下げ！】弊社限定未公開物件！豊田駅徒歩８分 ライオンズガーデン明大前パラダイム

→ライオンズガーデン明大前パラダイム

本アルゴリズムの構成プロセス **Process of making this algorithm**

**STEP1.**トレーニングセットを作る

**STEP2**.機械にトレーニングセットを学習させて、機械自身がマンション名かどうかを判定できるようにする

**STEP3.**学習済みアルゴリズムをファイル形式で保存する（シリアライズする）　　　　 **STEP4**.汚い文字列を送って、webサーバがキレイなマンション名を返すアルゴリズムを書く

**STEP1.**Making training set.

**STEP2.**Make machine learn traning set and be able to classify building name or not by machine itself.

**STEP3.**Serializing.

**STEP4**.To make algorithm that sends a long word group and get a building name from web server.

各**STEP**の詳細 **Details of each steps**

**STEP1.**トレーニングセットを作る

＊トレーニングセットは文字列とラベルから構成される。

トレーニングセットをどう作るか **How to make training set**

トレーニングセットは長い文字列を、文字以外の記号(】、！など)で区切って、短い文字列に分割する。

この際分割には、split関数を使い、文字以外の記号は人為的に指定する。さらに手作業で短い文字列にラベルをつける。マンション名なら「1」をつけ、マンション名でなければ「0」をつける。

今回作った「トレーニングセット」は長い文字列1000個を分割して、約7000個の短い文字列とラベル(0,or1)から構成されるリストととした。

また「テストセット」ととして学習後のアルゴリズムの精度を判定するリストを作成した（テストセットは、同様の構成要素からなる約3000個の文字列のリスト）

Training set is consists of short word groups that is divided from a long word groups by ‘split function’ in Python, being a list of 7000 short word groups. Its label 0 means not building name and 1 means building name. Also I made test set which is a list of 3000 short word groups.

**STEP2.**機械にトレーニングセットを学習させて、機械自身がマンション名かどうかを判定できるようにする

本STEPは、機械が自動で文字列をマンション名か、そうでないかを分類できるように、機械に学習させるSTEPである。

主な方法としては「ナイーブベイズ分類器」というアルゴリズムを使う。ナイーブベイズ分類器により、STEP1のトレーニングセットを読み込ませ、マンション名であるものとマンション名でない文字列を機械に学習させる。学習後には、機械はマンション名であれば「1」を、そうでなければ「0」と自動的に分類する。

そして機械が正確に分類できるかどうかの精度を確かめる方法として、STEP1で作った「テストセット」を使う。テストセットの文字列を機械にマンション名かどうか判定させ、人の手で振ったラベル（0,1）とあってるかどうかで精度（正解率）をみる。

今回の学習済みのナイーブベイズ分類器の精度は92%であった。約3000個の文字列の中で、2760個の文字列を正解と判定できたことになる（実際には96%だが、精度を考慮して計算すると92%）。アルゴリズムの精度としてはかなりいい方だといえる。

This step is to make machine learn the trains set, using ‘Naive Bayes classifier’ algorithm. Its machine performance is about 92%, which shows that machine can classify 2760 short word groups correctly in 3000 short word groups.

＊アルゴリズムの精度を上げるために **Ways for improving the machine performance**

今回のアルゴリズムの精度を向上させるための方法としては以下の方法が考えられる。

・トレーニングセットの文字列をさらに増やす。

→いろんなマンション名や、そうでない単語を、より多く含んだ文字列を含んだトレーニングセットを作ること。それにより、より精度の高いアルゴリズムを作ることができる。

現状としては、学習してない単語のみで構成される文字列は、本アルゴリズムでは判定できない。

The ways is probably to make new training set that consists of various building names and other phrases(not building name). This training set will be high quality as training set, resulting in improvement of machine performance. Currently, this algorithm can not classify the short word groups that consist of only unknown words in our training set.

**STEP3.**学習済みアルゴリズムをファイル形式で保存する（シリアライズする）

STEP2はanaconda上で学習させた。学習済みのナイーブベイズ分類器をファイル形式で保存（シリアライズ）し、他のサーバ上でも使えるようにする。

今回はシリアライズした学習済みのナイーブベイズ分類器をHTTPサーバ上で呼び出し、使えるようにすることを目的とする。該当部分のコードのみ記載する

Aim of serializing this algorithm is to reuse it on other server. Its cord is written with Python and shown as below.

該当コード The cord

joblib.dump(glf, bays\_model-2.csv')  #シリアライズ、”glf”は学習済みのナイーブベイズ分類器のこと

MODEL\_FILE='bays\_model-2.csv'

clf = joblib.load(MODEL\_FILE)  #HTTPサーバ上で、シリアライズしたアルゴリズムを読み出し、使用する

**STEP4.**汚い文字列を送って、**web**サーバがキレイなマンション名を返す「マンションクレンジングアルゴリズム」を作る

STEP4では、HTTPクライアントで長い文字列を送信し、webサーバ上からキレイなマンション名を受け取るアルゴリズムを作成した。

本アルゴリズムはHTTPクライアントとHTTPサーバにより構成される。

HTTPクライアントにより、リクエストメッセージとして長い文字列を送る。そして、HTTPサーバによりwebがレスポンスメッセージとしてキレイなマンション名を返すようにした。

This step is to complete ‘mansion cleansing algorithm’, which is consists of HTTP client and HTTP server. With HTTP client, send a long word group as request message from client such as browser. With HTTP server, get a clean building name as response message from web server.

【**HTTP**サーバ】**HTTP server is consisted of bottle, JSON and Python.**

→bottle、JSONおよびpythonで構成される

【**HTTP**クライアント】**HTTP client is shown below.**

→**curl** [http://localhost:8080/](http://localhost:8080/test.php)Housmart -X POST -H "Content-Type: application/json"

-d '{"building\_name": "ライオンズガーデン明大○○○○"}'

実例 **an actual example**

1.リクエストメッセージとして「【値下げ！】弊社限定未公開物件！豊田駅徒歩８分 ライオンズガーデン明大前パラダイム」を送る

2.レスポンスメッセージとしてマンション名「ライオンズガーデン明大前パラダイム」を受け取る。

＊単語「パラダイム」はトレーニングセットに含まれていない単語である。トレーニングセットに含まれていない単語でもマンション名と判定できるか、を確かめるために単語「パラダイム」は意図的につけたもの。

request message→【値下げ！】弊社限定未公開物件！豊田駅徒歩８分 ライオンズガーデン明大前パラダイム

response message→ライオンズガーデン明大前パラダイム

The word ‘パラダイム’ in response message is unknown word in our training set. To prove that this algorithm can classify a shot word group including unknown word in our training set as building name, I attached the word ‘パラダイム’ to response message deliberately.

留意点 **Points you should know about this algorithm**

・本アルゴリズムの精度はSTEP2のトレーニングセットに大きく左右される。トレーニングセットに含まれていない単語のみで構成される文字列は判定できない。

・本アルゴリズムの汎用性を高める、つまりより多くのマンション名に対応できるようにするには、STEP1で作成したトレーニングセットの精度を向上させる必要がある。（STEP2の「＊アルゴリズムの精度を上げるために」を参照）

・トレーニングセットの精度から、本アルゴリズムの汎用性はかなり高いと判断する。つまりより多くのマンション名に対応できる。

・This algorithm’s performance depends on training set in STEP2, not being able to classify the sort word groups that include only unknown word in our training set.

・To improve this algorithm’s performance, make training set quality more high.

・I consider that this algorithm’s performance is very high. In other words, it will deal with various building names and other phases.