音声認識システム用インタープリタの設計

14EC004 飯田頌平

2016 年 2 月 4 日

1　環境

本レポートは以下の環境を想定している。

コンピュータ	:	Raspberry Pi2
OS	:	Raspbian - Jessie
音声認識エンジン	:	Julius - 4..3.1

2　目的

Julius と Raspberry Pi を活用することにより、音声認識システムを搭載したオーディオを作ることが出来る。

しかし当初の音声認識オーディオは、構文解析の能力が非常に脆弱であり、ひとつの単語を認識することは得意であっても、人間が一般的に話す言葉を解釈することは難しかった。

そのためよりユーザに近い次元での対話が出来るようにするために、Julius から受け取った語句を解析するためのインタープリタを設計することにしたのである。

3　Julius について

Julius は京都大学と名古屋工業大学等が開発した音声認識エンジンである。

私が作成した音声認識オーディオでは、Julius のモジュールを使用して音素の解析を行っているため、モジュールの入出力に対応したインタープリタを設計する必要がある。

概略を述べると、Julius の入力にはオーディオデバイスからの音声入力が直接あてがわれ、出力には音素解析と辞書照合を経た単語が TCP のパケットから得られる。

この辞書については、文章を登録することができ、それによって単語の認識速度を上げるという工夫が取られている。故にある面では文章の解釈ができるインタープリタとしての役割を持っているとも取れる。

しかし送られてくるパケットの中身は単語であり、音素から単語への変換を行う際に登録された文章から文脈上自然な単語を選択することは出来ても、クライアント（パケットを受け取る側）がユーザの文章を解釈するという話には結びつかないのである。

Julius から送られてくる単語の羅列を、如何にアプリケーションが意味を認識できる文章として解釈するのか。それが Julius モジュールを使いインタープリタを作成する上での命題となる。

4　主題に立脚する

人間が言葉をかわすとき、どのようにして相手の言葉を解釈するのか。シェルのように決められた法則で言葉を受け取るのであれば解釈も容易いが、人間の話す言葉というものは単語の数も順番もとりとめがなく、無数に近い選択肢の中から当たりをつける必要がある。

自然と言葉の類推を行う脳の働きが解明されていない以上、既存のアルゴリズムを用いて文の解釈を行わなければならない。

そこで私が着目したのが単語である。単語に立脚して類推が行われているのではないか。人が文を解

釈するとき、まず主題となる単語を連想し、その主題からまた次の連想が行われるのではないか。

この仮定を基にすると、実に都合よくインタープリタを設計することができる。一度主題が連想されてしまえば、残りの文の構成要素の選択肢を一挙に絞ることができるためだ。

例を挙げると、まず会話の中から「コーヒー」という単語が主題として連想されたとしよう。すると人は次に「飲む」「淹れる」「買う」といった動作や、「紅茶」「パン」「喫茶店」といった関連する物の名前などを自然と連想するものだ。そのため会話の中で不十分にしか聞き取れなかった箇所があったとしても、相手の伝えたかった言葉を類推することができるのである。このとき多くの人は「国会議員」「爆発する」などのコーヒーとの関連度が低い語句を連想することはないだろう。故に主題と関係のある語句だけを、文の要素として選択肢に残すことができる。

ここまでの説明からして、私の過程に基づいたインタープリタには、

- 主題となる単語
- 主題に関連する要素（動作など）
- 主題と関連要素を結びつけるための属性

を内包する対応表が必要となることがわかるだろう。主題に関連する要素には、動作の他に肯定・否定・疑問・提案・命令などといった、文の性質を決定しうる語句も含まれている。肯定文か否定文かを取り違えてしまえばまったく逆の意味が解釈されてしまうためだ。

5　文章の判別方法

文章を判別する手段として、主題を軸にインタープリタを設計することは前の章で述べた。それでは如何にして主題を探せばよいのか。Julius のサーバから送られてくる語句の羅列の中から、如何にして主題を判別するのか。

5.1　語句の順番

英語は主語、述語、目的語と品詞の順番が明確な言語である。この場合は主題に相当する単語を見つけやすく、シェルのコマンドのように概ね先頭の語句に着目すればよい。

一方日本語は、主語と述語の位置がどこにあるのか、一概に言えない言語である。どのようにして主題となるべき単語を探し当てるのだろうか。

私はこの問題に対して、あえて順番というものを考えない方針を取ることにした。前口上や倒置法をいくら使われても問題のないシステムを作ろうとするなら、順番に固執して主題を判定する必要はないと判断したためだ。

無論助詞や助動詞といった付属語のような、一部の前後関係を要する語句に関しては斟酌するが、こと主題の場所という問題に対しては、順番を考慮しないことにする。

5.2　人が主題を探すプロセス

人が他人と会話を行う際には、必ず相手に投げ返す言葉を探している。その言葉は往々にして主題（subject）にまつわるものであり、人が会話の中で主題を探すプロセスをモデル化することがインタープリタの設計に有用であると言えるだろう。

会話とは言葉のキャッチボールである。つまりターン制でボールを投げ合い、補り合うのである。しかしボールが投げられればターンの交代を判別できるキャッチボールと違い、会話においてターンの交代を判別することは難しい。曖昧なのだ。

それでも人はある程度の間を感じたとき、返事を返す手番であると概ね判断するだろう。そこで間というものを判断基準と置く。ここでは単純に、間は入力音声のレベルを観測することによって検知されるものだとする。振幅検知による間の判定について、Julius では終端記号として実装されているので、それを用いて間を検出する。

間によって会話のターンを判断できるようになっ

たため、文章の開始点と終了点もまた判断できるようになった。インタープリタが一文という単位で語句を扱えるようになったのだ。

この一文という単位の中で、主題を探すことにする。文章の中にある語句をすべて検索し、主題にふさわしい語句を見つけ出すのである。

一般的に主題として扱われる可能性が高い語句は、主題テーブルに格納しておくことにする。その中の主題データに該当する語句が一文中に存在するか否かを探索し、一意の値が発見できたときには、それを主題と見做す。もし複数の主題データが発見されたときは、優先順位を決めればよい。主題データが発見されなかった場合には、主題のはっきりした会話を促すよう処理すればよい。

主題テーブルの実装については、以下の要素をフィールドとして持たせることにする。

1. 主題データ
2. クラス
3. 優先度
4. 矛盾処理

主題データは主題としてふさわしい語句である。クラスは主題の分類を指す。クラスについての詳細は次章で述べる。優先度は複数の主題が認められときの優先順位を数値型で決めておく。優先度は 1 以上の一意の整数であり、数値が若いほうが優先され、等しい場合には矛盾処理の記述に従う。優先度は以下のように分類される。

1 ～ 9	重要なシステム操作を行う命令
10 ～ 99	一般的なシステム操作を行う命令
100 ～ 999	単純な会話の材料

矛盾処理は優先度が等しい主題についての処理を指す。また、矛盾処理として以下の項目を定める。

- AGAIN
- FORCE
- IGNORE

AGAIN は重複した主題について、どちらを優先するべきか聞き返す。FORCE は強制的に主題として認識させる。その性質上、FORCE を設定する主題は限定されねばならない。IGNORE は FORCE の逆で、強制的に主題として認識させない。

5.3 クラス

主題はクラス（class）に結び付けられる。主題以外の語句もまたクラスに結び付けられており、主題に対応した非主題語句を呼び出す際に用いられる。

予め主題が特定できたのであれば、そのクラスに応じて語句の候補を絞り込むことができるようになり、高速かつ適切に意味解析ができるようになるのである。

クラスは N 次の子クラスを持つ。クラス毎に必要となる意味の正確性が異なるために、その度合いに応じて子クラスは異なる。また、子クラスは更なる子クラスを持つことができる。

主題が特定されたあとは 1 次クラスが呼び出され、主題を判定した場合と同じ手順で 1 次クラスの判定が行われる。そして必要に応じて N ＋ 1 次のクラスが順次呼ばれ、文章の特定が終了する。

クラステーブルは以下のフィールドを持つ。

1. 語句データ
2. 親クラス
3. 子クラス
4. 優先度
5. 矛盾処理

語句データ、優先度、矛盾処理については主題の場合と同じ定義がされている。親子クラスの関係については前述のとおりである。1 次クラスならば親クラスに NULL、末端クラスならば子クラスに NULL と入れる。またクラスはクラスごとにディレクトリが分けられた状態で保存されている。