必要事項を記入し，卒業論文と一緒に提出すること

卒業論文受領証　学生保管

学籍番号	C	0	1	1	2	3	3	6
氏　　名	寺田 佳輔							

	受領印
受領印を受けた後，本票を受け取り 大切に保管してください.	

卒業論文受領証　事務局保管

学籍番号	C	0	1	1	2	3	3	6	受領印
氏　　名	寺田 佳輔								
指導教員	田胡 和哉								
論文題目	人工知能ハブの 開発・利用および評価								

┌(┌^ o ^)┐ キレイに切り取ってね ┌(^ o ^┐)┐

2015
年 度

人工知能ハブの開発と評価

寺田 佳輔

田胡
研究室

[卒 業 論 文]

人工知能ハブの開発と評価

(指 導 教 員)　　田胡 和哉

コンピュータサイエンス学部 田胡研究室

学籍番号　C0112336

寺田 佳輔

[2015 年度]

東 京 工 科 大 学

卒 業 論 文

論 文 題 目

人工知能ハブの開発と評価

指 導 教 員	田胡 和哉	

提 出 日　2016 年 01 月 18 日

提 出 者

学　部	コンピュータサイエンス 学 部
学籍番号	C0112336
氏　名	寺田 佳輔

2015 年度　卒　業　論　文　概　　要

論 文 題 目

人工知能ハブの開発と評価

コンピュータ サイエンス学部	氏 名	寺田 佳輔	指 導 教 員	田胡 和哉
学籍番号 C0112336				

【概　要】

ルいず

目次

図目次

表目次

第 1 章

現状

1.1 近年の機械学習

1.1.1 人工無能

1.1.2 人工知能

1.1.3 neural network

1.1.4 DeepLearning

1.2 一般的な人工知能開発フレームワーク

1.3 一般的な人工知能の返答の流れ

第 2 章

提案

2.1 開発した人工知能の活用

今回提案するのは先ほど説明した一般的な人工知能フレームワークを用いて開発を行った人工知能を活用するためのフレームワークである．

2.1.1 知能の開発をサポートする既存フレームワーク

既存の人工知能フレームワークは，人工知能自体を作成することをサポートしている．具体例で言うと Chainer は Preferred Networks が開発したニューラルネットワークを実装するためのライブラリであり，実際の開発をすることの手助けをしている．

2.2 開発した知能を試す環境

今回提案するのは，フレームワークを用いて開発したアルゴリズムや，独自のアルゴリズムを考え，作成したプログラムを実際に動かし試す環境である．

通常，人工知能のアルゴリズムを試したいと考えた場合，そのプログラムに対して入力を与える入力の部分とその処理結果を出力する出力の部分を作成する必要がある．

作成した知能の出力結果がただ単に文字で入力して，文字で出力されれば良い場合は，準備をするのはほぼ手間が不要であるが，キャラクターとの会話などで試したい場合，非常に入出力の部分を作成するのに時間がかかってしまう．

そこで，今回は作成したアルゴリズムをすぐに試す環境を提供するフレームワークを提案する．

2.3 人工知能利用フレームワークの提案

人工知能利用フレームワークは会話や動作などの返答アルゴリズムを作成した際に，それらの作成したプログラムをフレームワーク上に適当に配置することで，状況や話題に応じて適切な作成した返答アルゴリズムが選択され Unity 上のキャラクターと会話を楽しむことができる，人工知能を利用すること

に焦点を当てたフレームワークである．

2.3.1 全体構成

この人工知能利用フレームワークの全体の構成を次の**図 2.1** に示す．

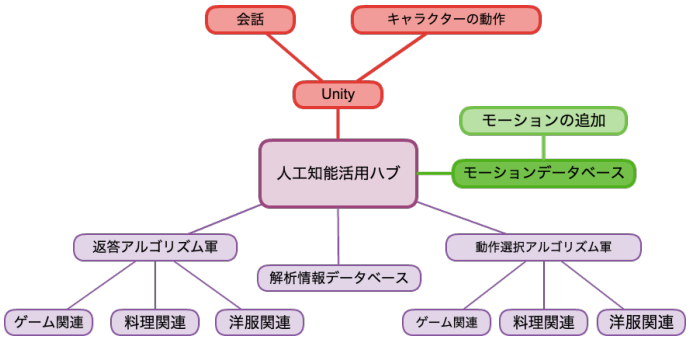

図 2.1　全体の構成図

この人工知能ハブは大きく分けて 3 つの要素で構成され，大きな流れで説明をすると Unity でユーザーが入力した内容をもとに人工知能活用ハブがその入力内容を受け取る．そして人工知能活用ハブの中で返答する内容が作り出され，モーションデータベースから適切な動作を選択し Unity へ動作と返答内容を出力する．この流れによってユーザーはキャラクターとの会話を行うことができる．

2.3.2 アルゴリズムのみを簡単に追加可能な知能ハブ

それではまずはじめに私が開発する，アルゴリズムのみを簡単に追加することができる人工知能活用ハブを提案します．

このハブでは作成した会話の返答，もしくはキャラクターの動作を選択するアルゴリズムを簡単に追加し話題によって追加したアルゴリズムの中から適切なアルゴリズムを用いて返答を行えるようにしている．

例えばゲーム関連の返答アルゴリズムを作り，試したいと考えた場合は，そのアルゴリズムを実装したプログラムをあらかじめ準備されている抽象クラスを用いて素早く作成し，**図 2.1** の返答アルゴリズム軍のプログラムに作成したプログラムを登録するだけで，ゲームの話題が来た時にそのアルゴリズムでキャラクターが返答するシステムを作ることができる．

同様にゲーム関連のキャラクターの動作を選択するアルゴリズムを作る場合は，そのアルゴリズムを抽象クラスを用いて素早く実装し，**図 2.1** の動作選択アルゴリズム軍の中に作成したプログラムを登録するだけで，ゲーム関連の会話をしている最中は，そのアルゴリズムを用いて動作を決定する仕組みを作ることができる．

はじめは，これらのアルゴリズムや人工知能が一つだけ実装されており，何も追加知能がない場合は

そのデフォルトアルゴリズムが選択されるようになっていますが，料理の話題に特化した話題解析アルゴリズムやゲームの話題に特化した感情解析のアルゴリズムが実装されることでより正確な解析が可能になるだけではなく，返答する際もゲーム専用の返答アルゴリズムなどがあることでより円滑なコミュニケーションが可能になると提案します.

様々なアルゴリズムが必要になることを考え，複数人で開発を行った際にも解析情報のデータベースによる共有などにより，よりスムーズに連携を行うことができるほか.

人工知能ハブでは，すでに解析した感情情報などの情報は全てデータベースによって共有されているので，ユーザーの入力した情報の解析を行うプログラムの開発は行わずに，すでにある感情解析プログラムの解析結果を使ってユーザーの感情状態を考慮した「会話ボット」などの開発を行うことも可能です.

2.3.3 作成したアルゴリズムを Unity ですぐに試せる機構

この人工知能利用フレームワークの人工知能活用ハブに登録された知能は Unity 上でのキャラクターとの対話ですぐに試すことができる.

共同研究者の藤井さんによると，MMD モデルを利用しているため好きなキャラクターで動作させることが出来，また，この人工知能利用フレームワークのために開発した，リアルタイムに動作を保管しながら動かす技術により，よりリアルな円滑なコミュニケーションが可能になっているという.

このように作成した人工知能をすぐにキャラクターとの対話という形で実行することができるため，入出力をどのような設計にするかや，開発はどうするかに迷うことなく，独自の対話アルゴリズムや人工知能の開発に専念することが可能になり，より高精度な対話を実現できると提案する.

2.3.4 Unity が利用可能なモーションを追加する機構

この人工知能利用フレームワークでは現在会話と動作の 2 つの出力を実装している.

ここで返答パターンは文字列で生成され，Unity で実行されるので無限のバリエーションで返答することができるが，動作（モーション）においては，現状その場で動作を生成することが難しい.

そのため，あらかじめモーションデータを作る必要があるが，そのモーション（動作）を定義するファイルを生成することは一般的には難しい，そこで共同開発の鈴木が Kinect で動作を定義し，データベースに保存，人工知能活用ハブと通信可能なプログラムを開発した.

この機構があることによって，人工知能活用ハブの中で新しい動きのパターンを追加したいとなった時にもすぐに Kinect を用いて動作ファイルを生成し，データベースに登録することで使えるようなる.

第 3 章

設計

知能ハブの構成を 3 段階に分けてと，Unity のキャラクターとの連携，解析や出力をするアルゴリズムを選定する時に利用している GoogkeAPI の流れで，人工知能利用フレームワークの構成を解説します．
最初に人工知能ハブの全体的な解析から出力の流れを示す図を以下の　　に提示します

3.1　入力された情報を解析する機構

まず初めに Unity 上のキャラクターへユーザが発言し，知能ハブが受け取った情報を解析する際の機構について解説したいと思います．

以下**図 3.1** に入力された情報が解析され，解析結果がデータベースに格納されるまでの構成を示す．

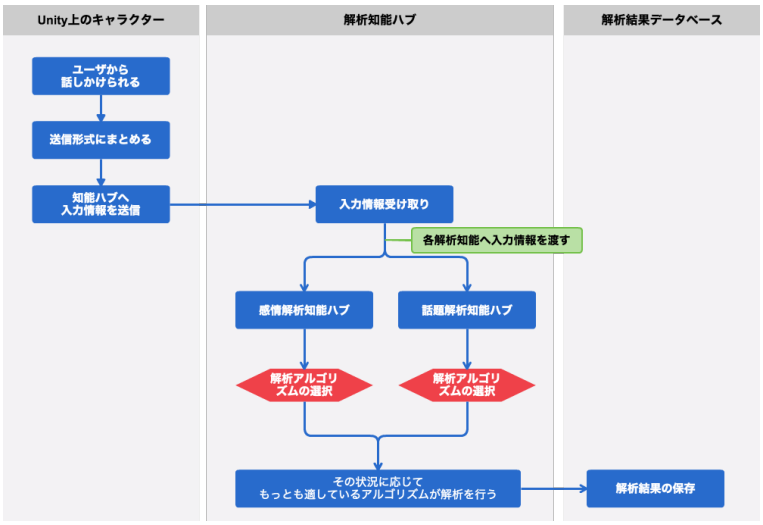

図 3.1　解析が行われるまでの図

図 3.1 の大まかな流れを説明すると，初めにユーザーが Unity 上で動作しているキャラクターに話しかけると，その内容を Unity が聞き取り，知能ハブへと送信します．送信した情報は知能ハブで受け取られ，その入力された情報は，それぞれ解析したい情報ごとに作られているハブへと渡されます．

図 3.1 の場合，感情を解析する感情解析知能ハブと話題を解析する話題解析知能ハブがあるため，入力された情報はこの 2 つの解析ハブへと送信されます．

情報が送信されると各解析ハブは入力された情報をもとに，登録されている各解析アルゴリズムの中からもっとも適切な解析アルゴリズムを選択し，実際の解析を行わせます．

実際に解析された情報は知能ハブ全体で共有されているデータベースへ保存することで，様々な場所から利用することができるようになります．

それでは以下の章で解析知能ハブの中のそれぞれの機能について説明したいと思います．

3.1.1 解析する情報別にアルゴリズムを保持する機能

図 3.1 を見て分かる通り，入力された情報は各解析する情報ごとに入力データを渡していきます．そして，各解析知能は入力された情報からその物事を解析するためのアルゴリズムを複数持っており，それを具体例を用いて表した図を以下の**図 3.2** に示します．

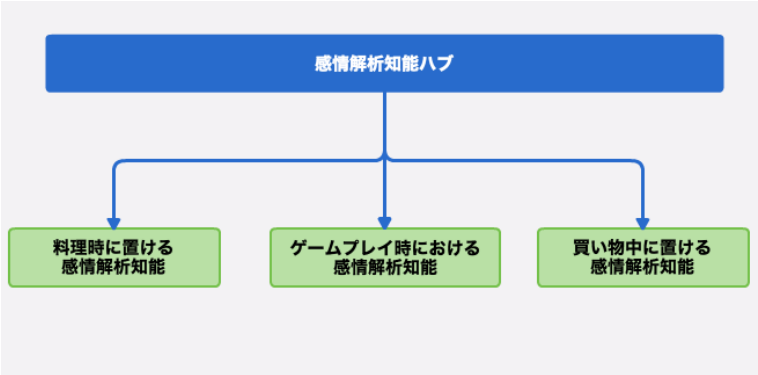

図 3.2　感情解析知能ハブとそれに付随する解析アルゴリズム

具体的に説明をすると，**図 3.2** の感情解析知能ハブは，感情を解析するための複数のアルゴリズムを持っていることになります．

各，感情解析知能ハブや話題解析知能ハブなどはアルゴリズム型の配列を持っており，その配列内に解析アルゴリズムの抽象クラスを実装したものを格納するだけで複数のアルゴリズムを保持し，適切なアルゴリズムが選択されるように設計されています．

また，解析する情報である感情や，話題といった種類はその他にも抽象クラスを実装し，解析知能ハブに登録することで追加することができます．

3.1.2　会話の話題別に解析するアルゴリズムを選ぶ機能

この人工知能ハブでは現在話している話題をもとに，どの解析アルゴリズムを選択するかを判定しています．
なので料理に関する話題をしているときは，料理関連の単語や会話に対応した解析アルゴリズムがあればそれが解析を行い，ない場合はその他の解析アルゴリズムの中でもっとも適した解析アルゴリズムが解析を行う設計になっています．
この話題を推定する際には google 検索の API を用いており，入力された内容を検索にかけてその結果から話題を推定しています．
こうすることで例えば，以下の**図 3.3**「クッパ[*1]が落ちた」という入力があったときに「クッパ　落ちた」で検索をした結果を取得します．

図 3.3　2015 年 12 月 20 日現在の「クッパ　落ちた」の Google 検索結果

図 3.3 の上位 5 件の検索結果を見るとゲームの話題であると判定され，ゲームに特化した感情解析を行うアルゴリズムが選択されます．

また，この時に「クッパ[*2]って美味しいよね」という入力があった場合，以下の**図 3.4** の「クッパ

[*1] クッパ：ゲーム「スーパーマリオブラザーズ」に登場する敵キャラクター
[*2] クッパ：クッパは韓国料理の一種。 スープとご飯を組み合わせた雑炊のような料理

美味しい」の検索上位 5 件を見て分かる通り，韓国料理のクッパの話題となるため料理に特化した感情解析を行うアルゴリズムが選択されます．

図 3.4　2015 年 12 月 20 日現在の「クッパ　美味しい」の Google 検索結果

このようにその時々に合わせて，適切な解析を行うアルゴリズムが選択されるような構造があり，これによって，より高精度な解析を行うことができます．

もし，このような機能がない場合，「クッパが落ちた」という文章は「落ちた」というキーワードから，たとえクッパという単語が料理名だと判明しても，ゲームの敵キャラクターとわからない限りはマイナスイメージな文と解析されると推測できます．

また，この両方を適切に解析できる，つまり現在の話題に限らず，感情を解析できるアルゴリズムを作成した場合は，そのアルゴリズムのみを感情解析知能ハブに登録することで確実にそのアルゴリズムが解析を行うように設定することが可能です．

3.1.3　解析アルゴリズムを簡単に追加する機能

実際に解析を行うアルゴリズム自体を簡単に追加する機能について解説します．
この，実際に解析を行うアルゴリズムの実装はあらかじめ定義されている抽象クラスを実装することで完了し，その実装の手順もソースコードの行数に換算すると最短 3 行でアルゴリズムを追加することが可能です．

図 3.2 のゲームプレイ時における感情解析知能を追加したい場合は，抽象クラスを実装後，感情解析知能ハブにある抽象クラス型を保持する配列に対して，作成した抽象クラスを拡張したプログラムを入れることで実装を行うことができます．

話題を解析する知能ハブが親の抽象クラスを実装したもので，それに付随する解析アルゴリズムは子の抽象クラスを実装したものという構図になり，その関係性を以下の**図 3.5** に示します．

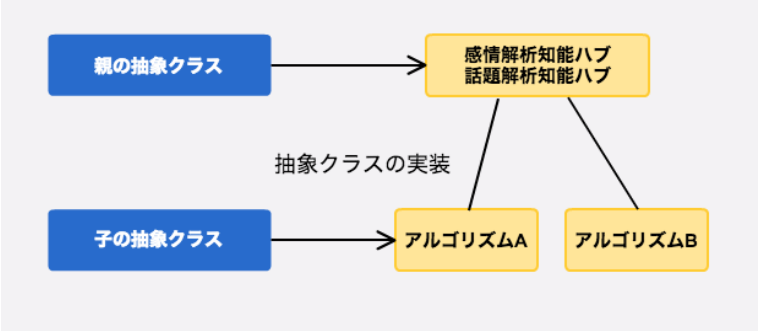

図 3.5　抽象クラスの関係と抽象クラスの実装例

図 3.5 の通り，親の抽象クラスだけでなく，子の抽象クラスに関しても簡単に実装を行い，追加する機構がある．

3.2　解析結果を保存する機構

この人工知能ハブには解析を行った情報やその他の様々な情報を保存するためのデータベースクラスが実装されている．

そして，このデータベースクラスはすべてのクラスで共有で利用できるように，すべての解析知能や出力を作成する知能の抽象クラスに含まれている．

3.2.1　解析情報を保存する機能

このデータベースは，解析した情報を保存する機能がある．
しかし，情報を保存するにあたり，解析を行うアルゴリズムを作る人によって解析結果の形式が異なることが予想できるため，どのようなオブジェクトでも保存が可能なように object 型を利用している．

実際に保存を行う場合は各解析アルゴリズム内でデータベースオブジェクトのメソッドに対して保存したい内容を引数で渡すだけで保存を行うことができる．

保存する際に付けられる名前は明確性と同一名のデータが存在しないように，その解析アルゴリズムのプログラム名＋データの形式という形で保存する．

例えば Mode-Topic-Game というゲーム話題解析知能が文字列で話題を保存したい場合はそのアルゴリズムの中で，解析が終わった時にデータベースオブジェクトの保存を行うメソッドに対して値を渡す．

そして，その保存した情報に対して Mode-Topic-Game-String という名前をつけることで明確性と同一名のデータが存在しないようにしている．

3.2.2　解析情報を取得する機能

その次に解析した情報を出力内容を作成するアルゴリズムの中から呼び出す方法について解説します．
実際に解析を行う際にはデータベースオブジェクトの情報取得メソッドに対して先ほどの解析情報の保存で説明した，欲しい情報の名前を指定することでその情報を取得することができる．

3.3　解析情報を元に出力内容を作成する機構

解析された情報を元に Unity のヤラクターに送信する出力内容を作成する工程についてユーザーが Unity 上のキャラクターとの会話をする例を表した**図 3.6** を用いて説明します.

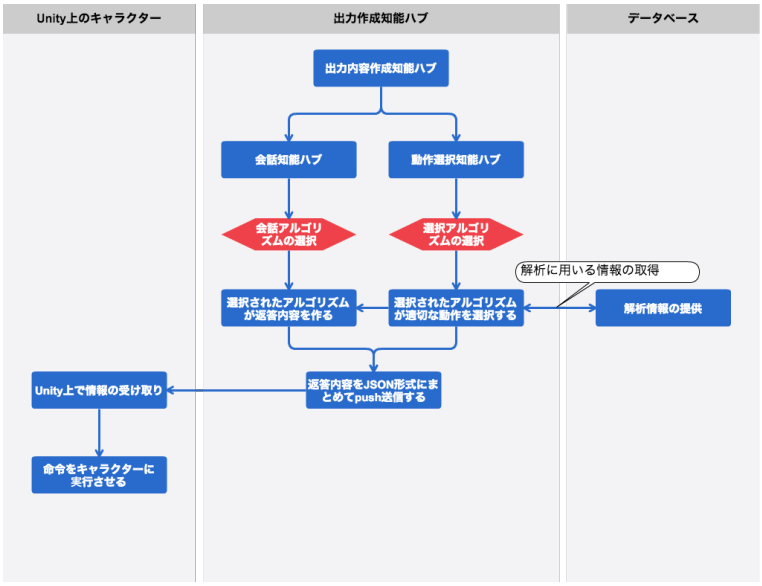

図 3.6　出力情報を作成するまでの流れ

図 3.6 を見て分かる通り出力作成知能ハブにも出力したい情報ごとにアルゴリズムを保持する機構があり，今回の**図 3.6** の場合は会話を行うための会話知能ハブとキャラクターの動作を選択するための動作選択知能ハブの 2 つがある.

それぞのハブでは入力された情報を元に，解析知能ハブの時と同じように最適なアルゴリズムを選択します，選択されたアルゴリズムはそれぞれデータベースにある，利用したい情報を取得し，返答内容や動作を決定します.

それぞれのアルゴリズム処理結果は Unity への命令形式である JSON 形式にまとめられ，websocket 通信で Unity へと送信され，Unity が命令を解釈，キャラクターが動作するという流れになります.

3.3.1　返答を行うタイミング

人工知能ハブが返答を行うことができるタイミングは 2 種類あります.
1 つめは相手から入力があった場合の返答，2 つ目が自ら発言する場合に返答する場合です.

相手から入力があった場合の返答は Unity から情報が送信されてきた時に出力を作成する知能ハブを呼び出すことで出力内容を作成し，返答を行っています.

自ら発言を行う場合は実装を行ってあるタイマーを利用して発言を行います．特定の時間や変数の値になった時に発言を行うように設定することが可能であり，出力内容作成知能ハブの自発的に発言する出力内容を作成するメソッドをそのタイミングで呼び出すことが可能になっています．

感情値や時間経過，状況の変化のあった時に websocket を用いて通信を Unity へ送信できるため，自発的に発言しているように見せることが可能です．

また，自ら発言する場合と，返答を行う場合でアルゴリズムが異なることが多いため，**図 3.5** のアルゴリズムを実装するための子の抽象クラスには返答する際のアルゴリズムと自ら発言する際のアルゴリズムを書くメソッドが用意されています．

3.3.2 会話の話題別に返答アルゴリズムを保持する機能

返答を行う際も，解析を行うときと同様に話題別に返答アルゴリズムを保持しています．

また，返答アルゴリズムを保持する仕組みに関しても同じく，返答アルゴリズム型の配列を持っており，その配列内に返答アルゴリズムの抽象クラスを実装したものを格納するだけで複数のアルゴリズムを保持し，適切なアルゴリズムが選択される部分についても同じ仕組みで動いています．

3.3.3 会話の話題別に返答アルゴリズムを選ぶ機能

出力情報ごとにアルゴリズムを保持しているため，解析知能ハブの時と同じように，その時々に合わせて最適なアルゴリズムが解析を行うようになっています．

3.4　作成した知能を Unity で試す機構

作成したアルゴリズムをすぐに実行し，試す環境として今回は統合開発環境を内蔵し、複数のプラットホームに対応するゲームエンジンである Unity を採用しました.

このゲームエンジンを用いることでウェブブラウザ上で動作するキャラクターを簡単に作成することができ，ブラウザ上で動作するため，様々なプラットフォームで試すことができます.

また，ブラウザを搭載していないデバイスの場合でも Unity 自体が複数のプラットフォームに対応しているため，様々な人が開発したアルゴリズムをすぐに試すことができます.

3.4.1　UnityWebPlayer での出力について

今回作成した人工知能利用フレームワークでは UnityWebPlayer を用いてブラウザ上でキャラクターとのコミュニケーションを取れるように設計しました.

図 3.7　Unity Web Player によるキャラクターの表示画面

図 3.7 のようにブラウザを搭載している PC や mac などのデバイスならばキャラクターを表示することが出来，作成したアルゴリズムをすぐに試すことが可能です.

3.4.2　Unity との連携に利用する WebSocket

Unity との通信には Web Socket を用いています.

web socket とはウェブサーバーとウェブブラウザとの間の通信のために規定を予定している双方向通信用の技術規格であり，それを採用した理由としてあげられるのが任意のタイミングでの push 通知

が可能な点です.

push 通知が可能になることによって，人工知能利用フレームワークから好きなタイミングで命令を送信し，Unity 上のキャラクターを動作させることができるようになるだけではなく，Unity 側のプログラムとしても命令がきた時にだけキャラクターを動作させ，ユーザーから入力があった時だけサーバへ入力情報を送信すればよいので処理が軽減されるという利点もあります.

3.4.3 Unity への送信フォーマットと作成

図 3.6 の「返答内容を JSON 形式にまとめて push 送信する」という部分の解説を行います.
この Unity への送信は汎用性の高い JSON 形式を用いて送信を行っています.

3.4.4 Unity からの受信フォーマット

3.5 アルゴリズムを選定する際に用いる GoogleAPI

3.5.1 GoogleAPI について

3.5.2 GoogleAPI の有効性

第 4 章

実装

4.1 開発環境

4.1.1 Java の利用

4.1.2 Maven フレームワーク

4.2 解析部分の実装

4.2.1 解析分野別にアルゴリズムを保持する機構

4.2.2 会話の話題別に解析するアルゴリズムを選択する機構

4.2.3 解析アルゴリズムを 3 行で簡単に追加する機構

4.2.4 現在実装している解析アルゴリズム

4.3 データベースの実装

4.3.1 全ての解析情報を保存する機構

4.3.2 解析した情報を取得する機構

4.4 返答情報を作成する機構

4.4.1 出力分野別にアルゴリズムを保持する機構

4.4.2 会話の話題別に出力を作成するアルゴリズムを選択する機構

4.4.3 返答アルゴリズムを 3 行で簡単に追加する機構

4.4.4 現在実装している出力アルゴリズム

4.5 Unity との通信の実装

4.5.1 Unity からの入力情報の受信

4.5.2 Unity への命令の送信

4.6 追加したモーションの利用

4.6.1 動作選択アルゴリズムの実装

4.7 GoogleAPI と形態素解析を用いた頻出単語表の作成する機構

第 5 章

実行結果

5.1 Unity の出力画面の図

5.2 実際の会話

5.3 アルゴリズムを追加した後の会話

第 6 章

結論

6.1 結論

6.1.1 アルゴリズムの追加による出力の変化

6.1.2 Google を用いた会話の話題推定の精度

6.1.3 簡単にアルゴリズムを追加できたか

謝辞

何か色々と感謝する.

参考文献

[1] ゼロの使い魔制作委員会：“ゼロの使い魔公式ウェブサイト” http://www.zero-tsukaima.com/zero/index.html (2012/12/28).

[2] 奥村晴彦 著：“LaTeX 2_ε 美文書作成入門 改訂第 3 版” (技術評論社 2004, 403pp).